ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

BÁO CÁO CUỐI KỲ

THỊ GIÁC MÁY TÍNH

PHÁT HIỆN VIÊM PHỔI

SỬ DỤNG MẠNG NƠ-RON TÍCH CHẬP (CNN)

Sinh viên thực hiện: Vũ Bá Anh

Nguyễn Tuấn Anh

Hoàng Lê Tuấn

Chuyên ngành: Khoa học máy tính & Thông tin

Lớp: K67A3

MỤC LỤC

LỜI MỞ ĐẦU

Viêm phổi là một bệnh lý phổ biến và có thể gây nguy hiểm đến tính mạng nếu không được chẩn đoán và điều trị kịp thời. Việc phát hiện viêm phổi thông qua các phương pháp hình ảnh, đặc biệt là chụp X-quang ngực, đóng vai trò quan trọng trong việc xác định mức độ và loại viêm phổi, từ đó đưa ra phương án điều trị thích hợp. Việc phân biệt giữa viêm phổi do vi khuẩn và viêm phổi do virus thường dựa vào những dấu hiệu khác biệt trong hình ảnh X-quang ngực. Ví dụ, viêm phổi do vi khuẩn thường thể hiện qua sự khu trú của sự thâm nhiễm trong một thùy phổi, trong khi viêm phổi do virus có xu hướng thể hiện hình ảnh mờ dần và lan tỏa ở cả hai phổi.

Mô hình sẽ áp dụng Mạng Nơ-ron Tích chập (CNN) – một phương pháp học sâu (Deep Learning) tiên tiến – để tự động phân loại hình ảnh X-quang ngực và phát hiện viêm phổi. CNN là một trong những kiến trúc mạng nơ-ron mạnh mẽ nhất hiện nay trong việc xử lý và phân tích hình ảnh, giúp cải thiện độ chính xác và tốc độ chẩn đoán. CNN có khả năng tự động học các đặc trưng từ hình ảnh mà không cần phải thiết lập các bước trích xuất đặc trưng thủ công, điều này làm cho nó trở thành lựa chọn lý tưởng trong các bài toán phân loại hình ảnh y tế.

Việc áp dụng CNN trong phân loại hình ảnh X-quang sẽ giúp hệ thống AI tự động phát hiện viêm phổi từ những hình ảnh X-quang ngực, cung cấp một công cụ hỗ trợ đắc lực cho các bác sĩ trong việc chẩn đoán bệnh, đồng thời giảm thiểu sai sót và tăng tốc độ đưa ra kết luận lâm sàng.

Nội dung chính của báo cáo gồm 3 chương:

Chương 1: Cơ sở lý thuyết, giới thiệu về CNN

Nội dung bao gồm các khái niệm cơ bản về lớp tích chập, lớp pooling, chức năng kích hoạt, và cách CNN hoạt động qua quá trình tự động học các đặc trưng quan trọng từ dữ liệu hình ảnh.

Chương 2: Thiết kế hệ thống

Trong chương này, sẽ mô tả chi tiết quy trình thiết kế hệ thống dự đoán viêm phổi sử dụng CNN. Nội dung bao gồm việc lựa chọn kiến trúc CNN phù hợp, thiết lập các tham số huấn luyện, và các bước xử lý dữ liệu đầu vào nhằm tối ưu hóa hiệu suất của mô hình.

Chương 3: Thực thi, đánh giá kết quả

Chương cuối cùng trình bày quá trình thực thi hệ thống trên tập dữ liệu thực tế, cùng với các phương pháp đánh giá hiệu suất của mô hình. Kết quả đạt được sẽ được phân tích để đánh giá độ chính xác, hiệu quả của CNN trong bài toán dự đoán viêm phổi.

CHƯƠNG 1: GIỚI THIỆU VỀ CNN

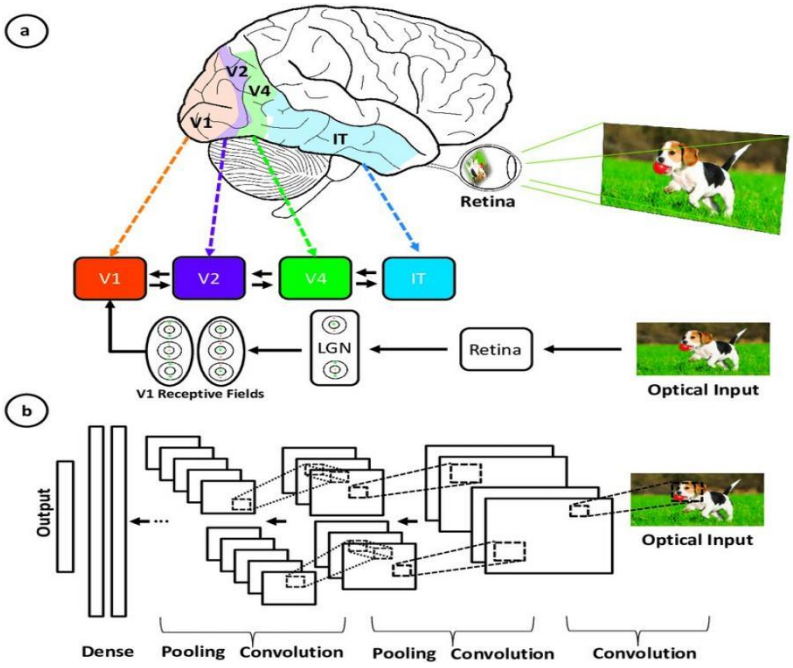
1.1 Tổng quan về mạng nơ-ron tích chập (CNN)

1.1.1 Mạng nơ-ron tích chập (CNN) là gì?

CNN là một loại mạng nơ-ron sâu (Deep Neural Network) được thiết kế đặc biệt để xử lý các dữ liệu có cấu trúc không gian, điển hình là hình ảnh. CNN được sử dụng rộng rãi trong các bài toán nhận dạng đối tượng, phân loại hình ảnh, phát hiện đối tượng, và phân đoạn ảnh, nhờ vào khả năng tự động trích xuất các đặc trưng từ dữ liệu mà không cần phải can thiệp vào cấu trúc dữ liệu.

1.1.2 Sự tương đồng của CNN với hệ thống thị giác của con người

Mạng nơ-ron tích chập (CNN) được lấy cảm hứng từ cách mà hệ thống thị giác của con người hoạt động. Mạng CNN mô phỏng các quy trình nhân diện đối tượng của thị giác con người thông qua các lớp tích chập (convolutional layers), lớp giảm kích thước (pooling layers) và lớp kết nối đầy đủ (fully connected layers). Dưới đây là 1 số điểm tương đồng:



Ảnh 1 Minh họa sự tương ứng giữa CNN đến vỏ não thị giác người

1. Kiến trúc phân cấp:

Cả hệ thống thị giác của con người và CNN đều có một cấu trúc phân cấp, trong đó các đặc trưng đơn giản được trích xuất ở các lớp đầu tiên và các đặc trưng phức tạp hơn được xây dựng dần dần ở các lớp sâu hơn. Trong CNN, các lớp tích chập đầu

tiên học được các đặc trưng cơ bản như cạnh và kết cấu, trong khi các lớp sau học các đặc trưng phức tạp hơn, như hình dạng và đối tượng.

2. Nhiều bản đồ đặc trưng (Multiple Feature Maps):

Trong quá trình xử lý hình ảnh của não bộ, nhiều bản đồ đặc khác nhau được trích xuất từ các tín hiệu đầu vào. CNN mô phỏng điều này thông qua việc sử dụng nhiều bộ lọc khác nhau trong mỗi lớp tích chập. Mỗi bộ lọc học một loại đặc trưng khác nhau.

3. Phi tuyến tính:

Các tế bào thần kinh trong vỏ não thị giác thể hiện các tính chất phản ứng phi tuyến tính, nghĩa là chúng có thể phản ứng mạnh với một số loại tín hiệu trong khi phản ứng yếu hoặc không phản ứng với các tín hiệu khác.

Trong CNN, tính phi tuyến tính này được đạt được thông qua các hàm kích hoạt như ReLU sau mỗi lớp tích chập. ReLU giúp mạng có thể học được các mối quan hệ phi tuyến tính giữa các đặc trưng, làm cho mạng trở nên mạnh mẽ hơn trong việc nhận diện các mẫu phức tạp.

1.2 Convolutional Layers

1.2.1 Khái niệm

Đây là khối xây dựng cơ bản quan trọng đầu tiên của khối CNN. Đúng như tên gọi, nhiệm vụ toán học chính của khối này ở đây là thực hiện các phép tích chập. Tức là áp dụng các 1 hàm cửa sổ trượt trên ma trận các điểm ảnh trong 1 hình ảnh. Hàm trượt này gọi là **kernel** hoặc **filter**, hai thuật ngữ này có thể thay thế cho nhau.

1.2.2 Kernel

Kernel là các ma trận số nhỏ. Các bộ lọc này di chuyển trên hình ảnh, thực hiện phép nhân từng phần tử với phần hình ảnh mà chúng bao phủ, trích xuất các đặc điểm như cạnh, kết cấu và hình dạng.

Trong lớp tích chập, nhiều bộ lọc cùng kích thước sẽ được áp dụng, và mỗi bộ lọc có nhiệm vụ nhận diện một mẫu hình cụ thể từ ảnh, chẳng hạn như độ cong của các chữ số, các cạnh, hay toàn bộ hình dạng của đối tượng.

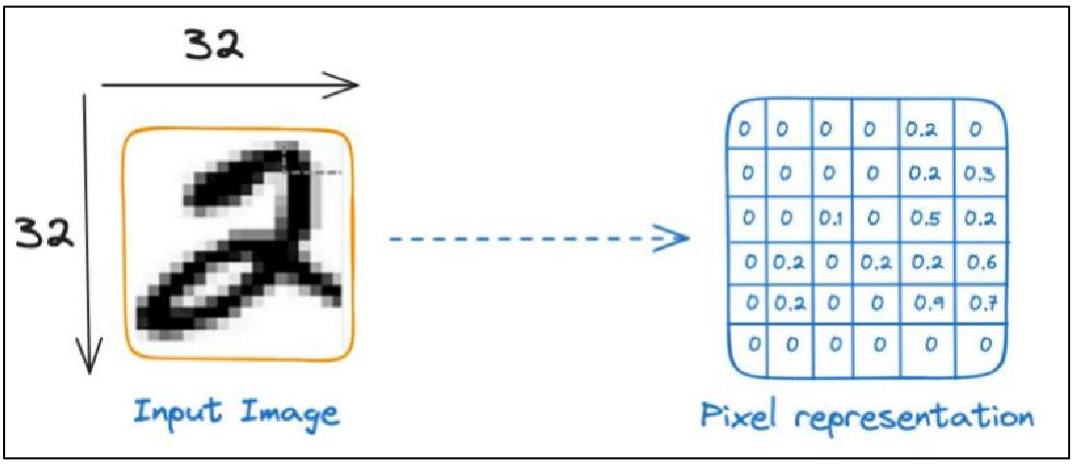
Nói đơn giản, trong lớp tích chập, ta sử dụng các ô lưới nhỏ (gọi là filter hoặc kernel) di chuyển qua từng vùng của ảnh. Mỗi ô lưới nhỏ giống như một chiếc kính

lúp, tìm kiếm các mẫu hình cụ thể trong ảnh, chẳng hạn như đường thẳng, đường cong, hoặc các hình dạng. Khi di chuyển qua ảnh, nó sẽ tạo ra một lưới mới, làm nổi bật những vị trí phát hiện được các mẫu hình này.

Ví dụ, một bộ lọc có thể giúp phát hiện các đường thẳng, bộ lọc khác thì tìm kiếm các đường cong, v.v. Bằng cách sử dụng nhiều bộ lọc khác nhau, CNN có thể thu thập đầy đủ các đặc trưng của những mẫu hình trong ảnh, giúp nhận diện tốt hơn toàn bộ nội dung của ảnh.

1.2.3 Cách phép tích chập thực hiện trên hình ảnh

Hình ảnh thực chất là lưới bao gồm các pixel. Các pixel này tượng trưng cho đơn vị nhỏ nhất trong hình ảnh và nắm giữ các thông tin về màu cũng như cường độ tại điểm pixel đó.



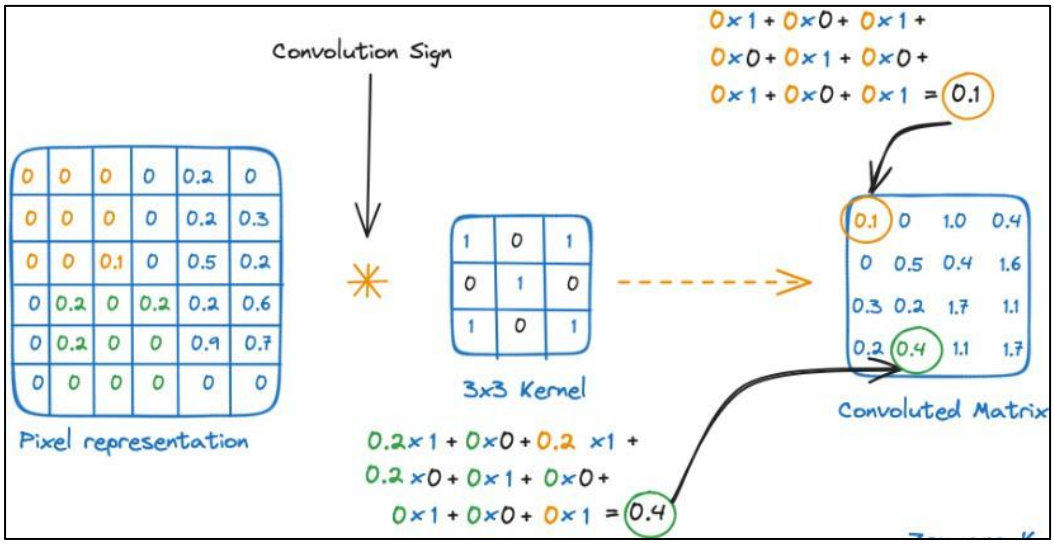
Ảnh 2 Minh họa sự biểu diễn pixel của ảnh

Với ma trận hình ảnh và kernel đầu vào, chúng ta sẽ thực hiện phép tích chập bằng cách phép toán tích vô hướng cụ thể như sau:

1. Bắt đầu đặt kernel ở góc bên trái của ma trận ảnh. Kernel sẽ bao phủ 1 vùng nhỏ trong ma trận ảnh tương đương với kích thước của kernel
2. Thực hiện phép nhân từng phần tử giữa các giá trị trong kernel và các giá trị pixel tương ứng đang bị bao phủ bởi kernel trong ảnh.
3. Cộng tất cả các giá trị sau phép nhân để tính tổng của các tích đó. Kết quả này là giá trị đại diện cho điểm ảnh hiện tại.
4. Gán giá trị tính được vào vị trí đầu tiên (góc trên bên trái) trong ma trận đầu ra. Đây sẽ là giá trị đầu tiên trong ma trận sau khi tích chập.

5. Dịch chuyển kernel theo kích thước của cửa sổ trượt. (thường là 1 hoặc nhiều hơn, tùy thuộc vào stride được chọn).
6. Lặp lại các bước từ 1 đến 5 cho đến khi kernel đã quét qua toàn bộ ảnh.

   Kết quả cuối cùng là một ma trận đầu ra tích chập biểu diễn các đặc trưng đã được trích xuất từ ảnh.

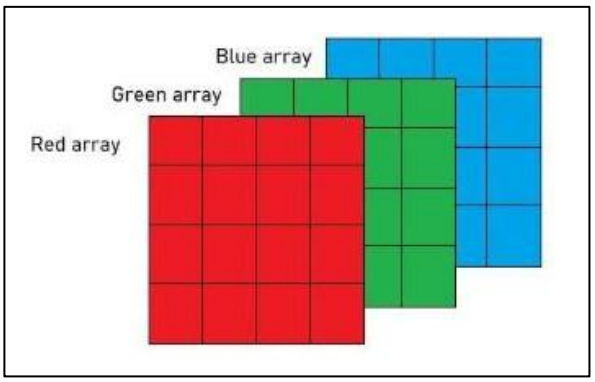


Ảnh 3 Biểu diễn phép tích chập

1.2.4 Channels, Stride, Padding

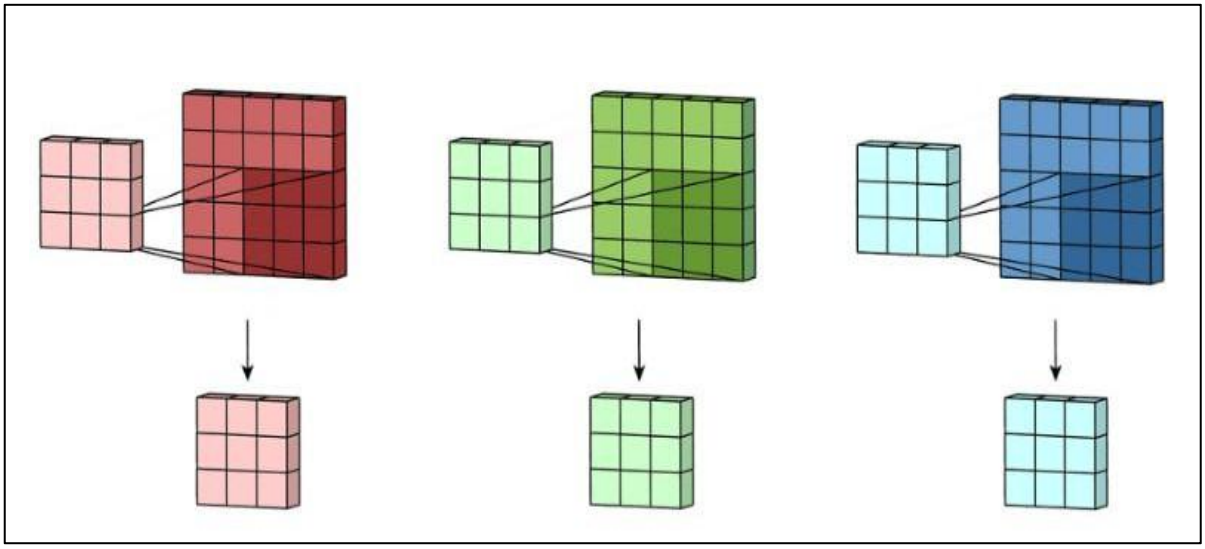
Channels:

Một hình ảnh thường được cấu tạo bởi 3 channel (RGB) và được biểu diễn dưới 3 ma trận khác nhau tương ứng, mỗi kênh màu tương ứng với một thành phần cường độ sáng của màu sắc đó trong từng điểm ảnh (pixel) của hình ảnh. Khi biểu diễn dưới dạng ma trận, mỗi kênh sẽ là một ma trận hai chiều, với kích thước tương ứng với chiều cao và chiều rộng của ảnh.



Ảnh 4 Biểu diễn ma trận ảnh RGB

Khi áp dụng kernel lên ảnh RGB, kernel cũng sẽ có 3 chiều để phù hợp với các kênh màu của ảnh đầu vào. Kernel sẽ quét qua từng kênh màu cùng lúc và thực hiện tích chập trên cả ba kênh, rồi kết hợp lại để tạo ra một ma trận đầu ra duy nhất. Kết quả là một bản đồ đặc trưng (feature map) đại diện cho các đặc trưng của cả ba kênh màu.



Ảnh 5 Biểu diễn tích chập trên từng kênh màu của ảnh

Stride:

Khi thực hiện phép tích chập, kernel sẽ di chuyển qua các điểm ảnh (pixel) của ảnh đầu vào, nhưng có nhiều cách khác nhau mà kernel có thể di chuyển.

Stride (bước nhảy) là số pixel mà kernel di chuyển qua trên ảnh đầu vào trong mỗi lần. Tham số này quyết định tốc độ và cách thức kernel quét qua ảnh.

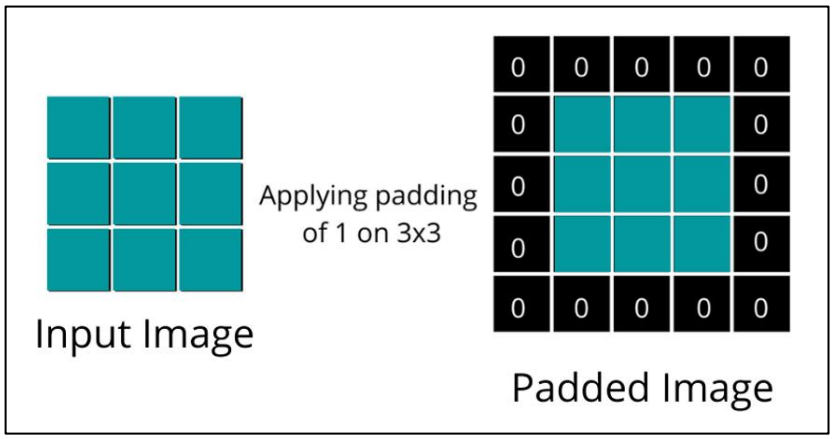
Sử dụng stride lớn hơn giúp giảm kích thước của ma trận đầu ra, từ đó giúp giảm thiểu yêu cầu tính toán kiểm soát tình trạng overfitting, nhưng đồng thời cũng có thể làm mất đi một số chi tiết nhỏ trong ảnh.

Padding:

Là việc thêm các pixel vào xung quanh viền của ảnh đầu vào, nhằm giải quyết vấn đề thông tin không được quét đủ ở các vùng biên của ảnh.

Khi chúng ta thực hiện phép tích chập, các pixel ở biên của ảnh thường chỉ được quét một vài lần, ít hơn so với các pixel ở trung tâm, điều này có thể dẫn đến mất mát thông tin tại vùng biên. Padding giúp giải quyết vấn đề này bằng cách thêm các pixel

giả (thường là giá trị 0) xung quanh viền của ảnh, làm tăng kích thước ảnh và đảm bảo rằng các đặc trưng quan trọng ở viền ảnh cũng được kernel quét đủ lần.



Ảnh 6 Ảnh padding

1.3 Pooling Layers

Mặc dù các lớp tích chập có thể làm giảm kích thước đầu ra, nhưng mục tiêu chính của chúng không phải là giảm số chiều mà là trích xuất các đặc trưng

Thực tế, trong phần lớn các trường hợp, ta không giảm kích thước dữ liệu.

Ngược lại, chúng ta tạo ra thêm các kênh đặc trưng mới, không tồn tại trước đó. Vì vậy, ngay cả khi kích thước không gian của bản đồ đặc trưng (feature map) có thể nhỏ hơn sau mỗi lớp tích chập, nhưng số lượng bản đồ đặc trưng lại nhiều hơn.

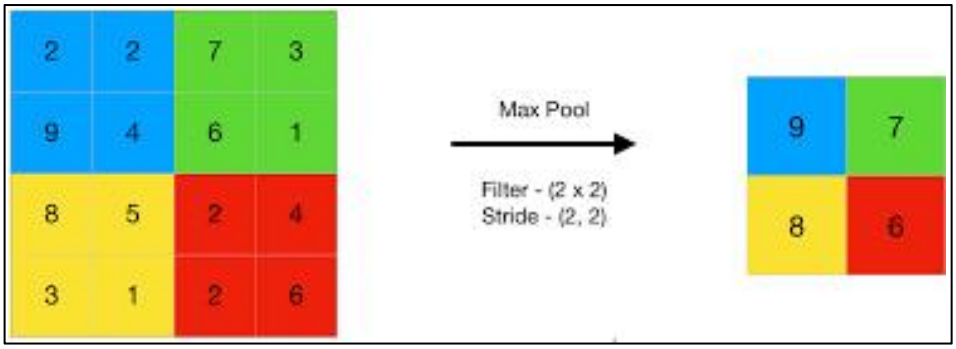
1.3.1 Khái niệm lớp Pooling

Lớp pooling có chức năng tóm tắt thông tin trong từng vùng nhỏ của bản đồ đặc trưng và giữ lại những đặc trưng quan trọng nhất, từ đó làm giảm kích thước không gian mà không làm mất nhiều thông tin cần thiết. Việc này giúp cho mô hình tập trung vào các đặc trưng quan trọng và ổn định hơn khi phát hiện các mẫu ở các vị trí khác nhau.

1.3.2 Các phương pháp Pooling thông dụng

1. Max Pooling:

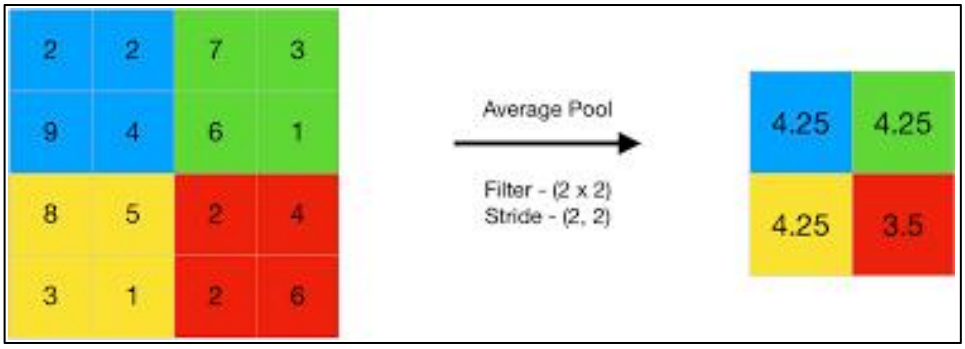
   Lấy giá trị lớn nhất từ mỗi vùng trong bản đồ đặc trưng. Phương pháp này giúp giữ lại những đặc trưng nổi bật nhất.



Ảnh 7 Max Pooling

2. Average Pooling:

Tính giá trị trung bình của các phần tử trong mỗi vùng. Phương pháp này ít được sử dụng hơn nhưng vẫn có thể hữu ích trong một số trường hợp, đặc biệt khi cần làm mịn bản đồ đặc trưng.



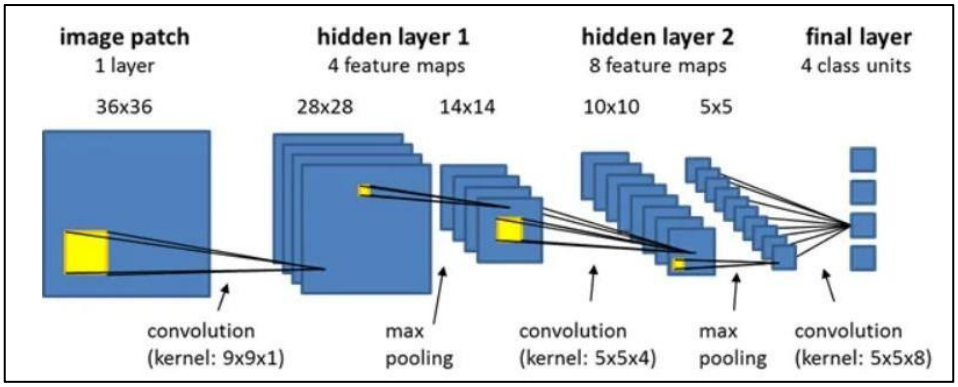
Ảnh 8 Average Pooling

1.3.3 Các đặc tính quan trọng

Không thay đổi số lượng kênh: Lớp pooling không giảm số lượng kênh. Mỗi phép pooling được thực hiện độc lập trên từng kênh, vì vậy số lượng kênh trước và sau lớp pooling vẫn giữ nguyên.

Giảm kích thước không gian: Pooling giảm chiều rộng và chiều cao của bản đồ đặc trưng, giúp mạng dễ dàng tính toán và tăng tính khái quát.

Dưới đây là khái quát cách hoạt động và tương tác giữa Convolutional Layer cùng với Pooling Layer trong mô hình CNN:



Ảnh 9 Minh họa Pooling Layer

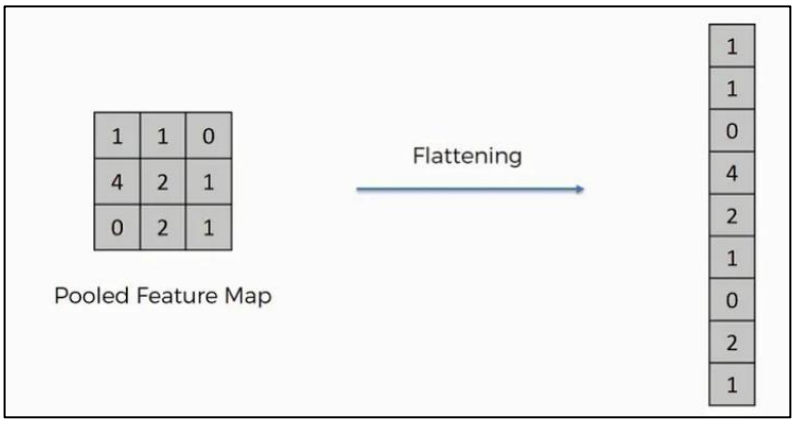
Như ví dụ trên đã thấy, Pooling làm giảm số chiều của dữ liệu đầu vào mà không làm giảm số lượng kênh.

Lớp Pooling là một thành phần quan trọng trong kiến trúc CNN, giúp giảm kích thước không gian của bản đồ đặc trưng mà không làm mất đi các đặc trưng quan trọng. Bằng cách áp dụng các kỹ thuật như Max Pooling hoặc Average Pooling, lớp pooling giúp giảm tải tính toán và kiểm soát hiện tượng quá khớp, làm cho mô hình trở nên hiệu quả hơn và linh hoạt hơn. Nhờ vậy, CNN có thể giải quyết tốt các bài toán nhận diện và phân loại hình ảnh phức tạp với độ chính xác cao hơn.

1.4 Fully Connected Layers

1.4.1 Flattening Layers

Có nhiệm vụ chuyển đổi dữ liệu từ dạng ma trận đa chiều (2D hoặc 3D) thành một vector 1 chiều (1D). Cụ thể, lớp flattening "làm phẳng" bản đồ đặc trưng từ convolutional và pooling layers.



Ảnh 10 Flattening

Mặc dù đã bị đổi chiều dữ liệu, nhưng nó không hề làm thay đổi bất cứ điều gì tới thông tin dữ liệu.

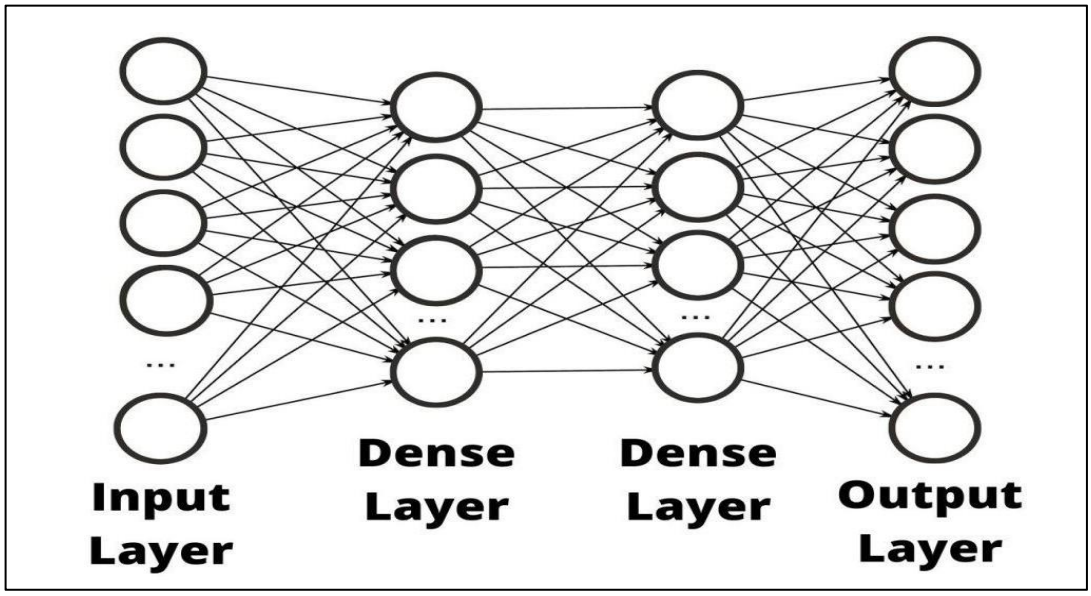
Tại sao lại cần Flattening Layer?

1. Chuyển đổi dữ liệu thành 1D: Dữ liệu từ các lớp tích chập và pooling thường là ma trận hoặc tensor nhiều chiều, và lớp fully connected chỉ hoạt động với dữ liệu 1 chiều. Lớp flattening giúp chuyển dữ liệu về dạng vector 1D để đảm bảo tính tương thích.
2. Tích hợp các đặc trưng: Flattening cho phép kết hợp tất cả các đặc trưng đã được trích xuất trong suốt quá trình tích chập. Khi dữ liệu ở dạng vector 1 chiều, mô hình có thể dễ dàng học cách kết hợp các đặc trưng này để đưa ra dự đoán chính xác hơn.

1.4.2 Dense Layers

Trong CNN, Dense layer là một lớp mà mỗi neuron của nó kết nối với tất cả các neuron trong lớp trước. Đây là lớp truyền thống của mạng nơ-ron, thường xuất hiện ở cuối của mô hình CNN để xử lý và tổng hợp các đặc trưng sau khi đã qua Convolutinal layers và Pooling layers.

Khi các đặc trưng này đến Dense layer, lớp này sẽ tổng hợp chúng để thực hiện các tác vụ phân loại hoặc hồi quy. Trong một mô hình CNN cho phân loại ảnh, lớp Dense cuối cùng thường có số lượng neuron bằng số lớp cần phân loại, với hàm kích hoạt softmax hoặc sigmoid để dự đoán xác suất cho từng lớp.



Ảnh 11 Dense Layer

Mỗi nơ-ron có các trọng số riêng (weights) gắn với từng đặc trưng đầu vào và có thêm một giá trị dịch (bias). Đầu ra của mỗi nơ-ron sẽ là:

$$y = activation(W.X+b)$$

W là vector trọng số

X là vector đầu vào

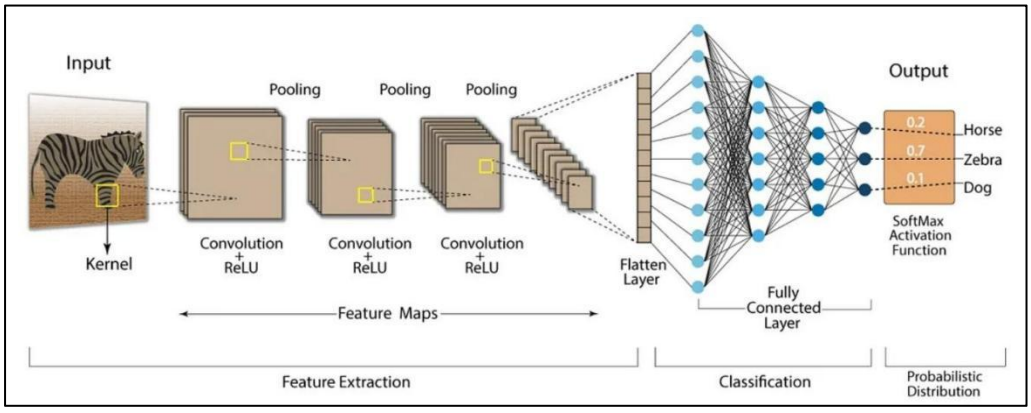
b là bias

activation là hàm kích hoạt, như ReLU, sigmoid, softmax, giúp xác định giá trị đầu ra cuối cùng.

Lớp dense giúp chuyển đổi từ các đặc trưng đã trích xuất thành dự đoán cuối cùng của mô hình, và là thành phần chính giúp mạng học cách tổng quát hóa từ các đặc trưng để phân loại hay dự đoán chính xác hơn.

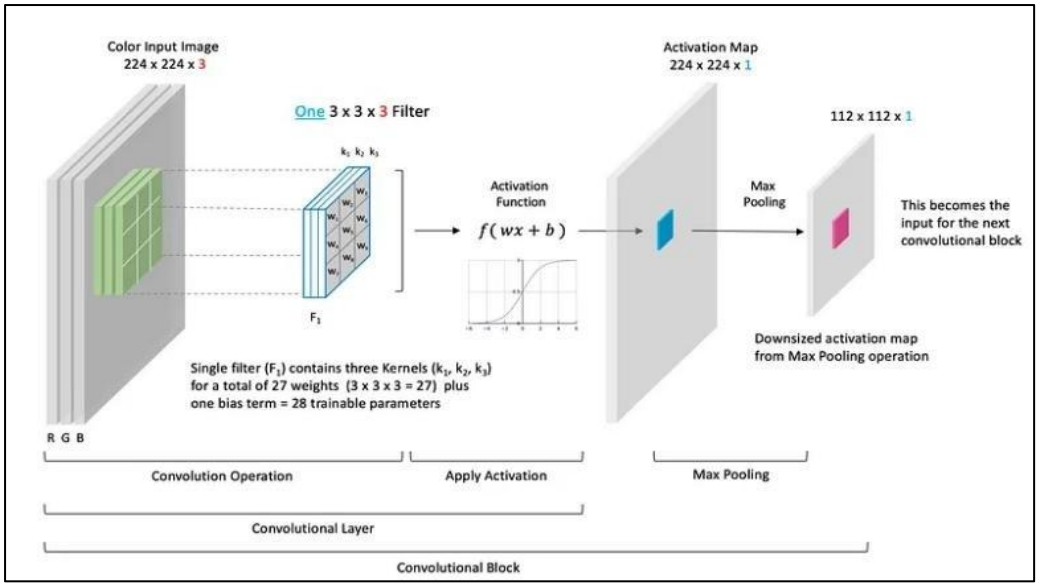
1.5 Activation Functions

Hàm kích hoạt là thành phần không thể thiếu trong mạng nơ-ron, vì nếu thiếu chúng, chúng ta sẽ chỉ tạo ra một mô hình tuyến tính, rất hạn chế khi cần học các mối quan hệ phức tạp trong dữ liệu.



Ảnh 12 Sơ đồ hoạt động của CNN

Trong phần trích xuất các đặc điểm (Feature Extraction), các hàm kích hoạt sẽ sử dụng ngay sau khi thực hiện tích chập.



Ảnh 13 Activation Function

Lớp pooling và flattening không áp dụng hàm kích hoạt vì mục tiêu của nó chỉ là giảm kích thước của đặc trưng đầu vào chứ không phải thêm phi tuyến.

Một số hàm kích hoạt phổ biến trong CNN bao gồm:

1. ReLU (Rectified Linear Unit): Hàm ReLU là hàm kích hoạt phổ biến nhất trong CNN với công thức sau:

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

   Hàm này giữ lại các giá trị dương và loại bỏ các giá trị âm (bằng cách đặt về 0). ReLU giúp giảm thiểu hiện tượng biến mất đạo hàm, giúp quá trình lan truyền ngược hiệu quả hơn.
2. Sigmoid: Hàm Sigmoid đưa đầu ra về khoảng (0, 1), giúp phù hợp cho các bài toán nhị phân:

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

   Tuy nhiên, sigmoid dễ gây ra hiện tượng bão hòa gradient, làm chậm quá trình học.
3. Sofmax: Thường được sử dụng trong lớp đầu ra cho các bài toán phân loại nhiều lớp. Hàm này đưa các đầu ra về xác suất, sao cho tổng xác suất bằng 1

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

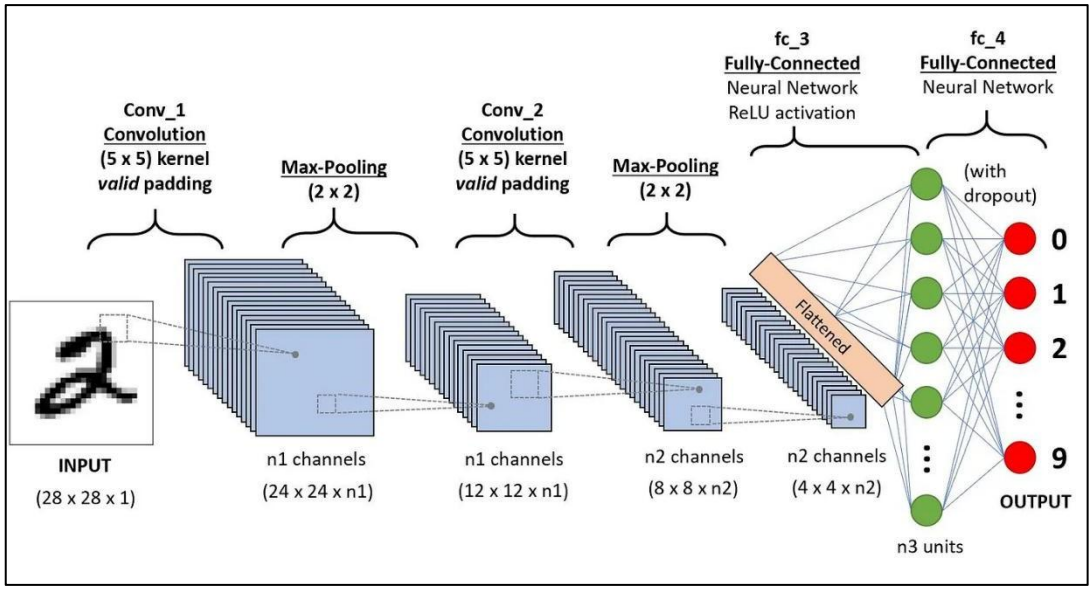
Softmax hữu ích khi muốn xác định lớp đầu ra có xác suất cao nhất cho một đối tượng đầu vào.

Trong CNN, các hàm kích hoạt thường được chọn dựa trên yêu cầu cụ thể của bài toán, với ReLU và Softmax là những lựa chọn phổ biến trong các lớp ẩn và lớp đầu ra tương ứng.

CHƯƠNG 2: THIẾT KẾ HỆ THỐNG MÔ HÌNH PHÁT HIỆN VIÊM PHỔI

2.1 Tổng quan về hệ thống

Hệ thống được thiết kế nhằm nhận diện và phát hiện viêm phổi sử dụng mạng nơ ron tích chập. Đây là một loại mạng nơ-ron chuyên biệt, thực hiện việc trích xuất đặc trưng từ hình ảnh trước khi tiến hành phân loại. Quá trình thực hiện được mô tả tổng quát như ảnh sau:



Ảnh 14: Mô phỏng tổng quát mô hình CNN

Hệ thống sẽ trải qua các bước chính sau đây:

1. **Tăng cường dữ liệu (Data Augmentation)**: Mặc dù hiện tại đã có sẵn 5,863 ảnh X-ray, mô hình sẽ áp dụng kỹ thuật tăng cường dữ liệu để mở rộng tập dữ liệu huấn luyện. Kỹ thuật này sẽ giúp tạo ra nhiều biến thể của hình ảnh gốc, từ đó tăng tính đa dạng cho tệp dữ liệu huấn luyện.

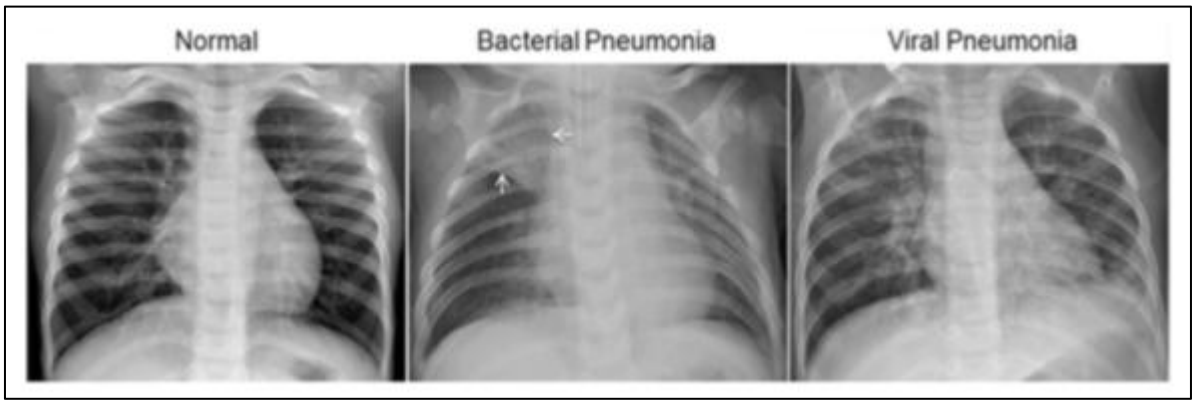
2. **Chuẩn hóa dữ liệu**: Bước này thực hiện chuẩn hóa và định dạng lại dữ liệu ảnh để chuẩn bị đầu vào cho mô hình CNN. Việc chuẩn hóa ảnh trước giúp mô hình học nhanh hơn, vì khi các giá trị pixel nằm trong khoảng [0, 1], gradient sẽ ổn định hơn trong quá trình tối ưu hóa. Ngoài ra, điều này giúp giảm ảnh hưởng của ánh sáng hay cường độ sáng của ảnh.
3. **Sử dụng mô hình CNN (Convolutional Neural Network)**: Phương pháp sử dụng CNN là lựa chọn phù hợp cho dữ liệu ảnh lớn nhờ khả năng xử lý ảnh vượt trội. CNN sẽ học các đặc trưng phức tạp trong hình ảnh và tối ưu hóa quá trình phát hiện viêm phổi qua ảnh.

2.2 Chuẩn bị dữ liệu

2.2.1 Dữ liệu hình ảnh

Trong mô hình này, chúng ta sẽ sử dụng các ảnh X-quang lồng ngực (hướng chụp trước-sau) được lấy từ dữ liệu hồi cứu của các bệnh nhân nhi từ 1 đến 5 tuổi tại Trung tâm Phụ nữ và Trẻ em Quảng Châu, Trung Quốc. Tất cả các ảnh X-quang được thực hiện trong quá trình chăm sóc lâm sàng định kỳ của bệnh nhân.

Bộ dữ liệu được tổ chức thành 3 thư mục riêng biệt: train (dữ liệu huấn luyện), test (dữ liệu kiểm tra) và val (dữ liệu kiểm định). Mỗi thư mục chứa các thư mục con tương ứng với từng loại ảnh (Pneumonia/Normal). Tổng cộng, bộ dữ liệu bao gồm 5.863 ảnh X-quang lồng ngực (định dạng JPEG) và được chia thành 2 loại: Pneumonia (viêm phổi) và Normal (bình thường).

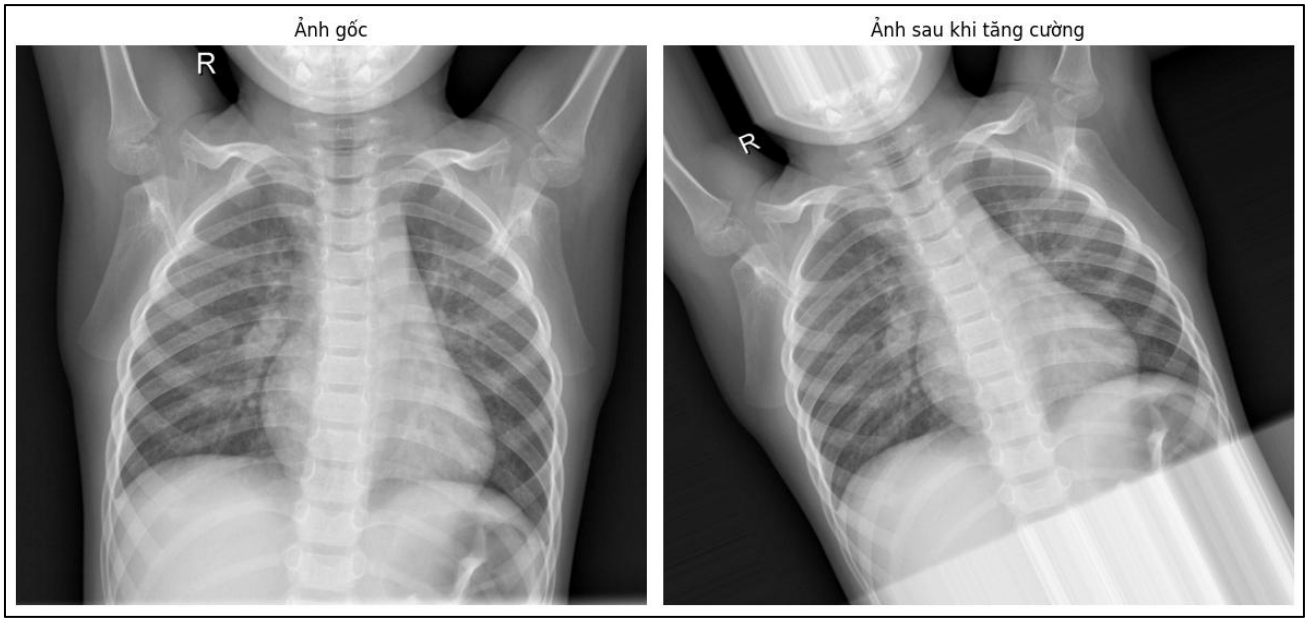


Ảnh 15 Tập dữ liệu viêm phổi

2.2.2 Dữ liệu tăng cường

Để tránh vấn đề overfitting, cần mở rộng dữ liệu huấn luyện. Điều này có thể thực hiện bằng cách tăng kích thước tập dữ liệu hiện tại thông qua các phép biến đổi.

Các phương pháp biến đổi dữ liệu theo cách làm thay đổi biểu diễn mảng nhưng vẫn giữ nguyên nhãn được gọi là kỹ thuật tăng cường dữ liệu (data augmentation).



Ảnh 16 Ảnh sau khi thực hiện biến đổi tăng cường

Kỹ thuật tăng cường dữ liệu làm đa dạng cho dữ liệu huấn luyện và giảm nguy cơ overfitting như sau:

1. Xoay ngẫu nhiên ảnh huấn luyện với góc xoay tối đa 30 độ.
2. Phóng to hoặc thu nhỏ ngẫu nhiên một số ảnh với mức zoom tối đa là 20%.
3. Dịch chuyển ngang ngẫu nhiên ảnh huấn luyện tối đa 10% chiều rộng của ảnh.
4. Dịch chuyển dọc ngẫu nhiên ảnh huấn luyện tối đa 10% chiều cao của ảnh.
5. Lật ngang ngẫu nhiên một số ảnh huấn luyện.

Sau khi áp dụng các kỹ thuật này, dữ liệu huấn luyện được mở rộng và đa dạng hóa, giúp cải thiện khả năng tổng quát hóa của mô hình. Khi mô hình đã sẵn sàng, tập dữ liệu huấn luyện được sử dụng để huấn luyện mô hình.

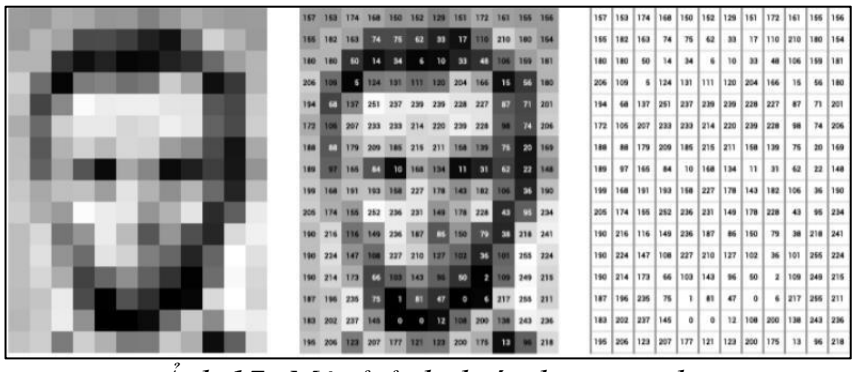
2.2.3 Chuẩn hóa dữ liệu

Ảnh được chuẩn hóa bằng cách chia toàn bộ giá trị pixel (ban đầu nằm trong khoảng [0, 255]) cho 255, đưa dữ liệu về khoảng [0, 1]. Điều này giúp giảm ảnh hưởng của sự khác biệt về cường độ ánh sáng giữa các ảnh và tăng tốc độ hội tụ của mô hình trong quá trình huấn luyện. Sau đó, dữ liệu được định dạng lại thành kích thước phù hợp

với đầu vào của CNN, với hình dạng (batch_size, height, width, channels), trong đó channels là 1 đối với ảnh xám. Đồng thời, nhãn (y_train, y_val, y_test) cũng được chuyển đổi thành mảng NumPy để dễ dàng sử dụng trong quá trình huấn luyện và đánh giá mô hình.

2.3 Xây dựng mã nguồn mô hình CNN

Một ảnh là một lưới các pixel. Mỗi pixel đại diện cho một đơn vị nhỏ nhất của hình ảnh và chứa thông tin về màu sắc hoặc độ sáng của điểm ảnh đó. Ở mô hình lần này, chúng ta còn thực hiện thêm việc chuyển đổi ảnh về ảnh xám, nên 1 ảnh sẽ được hình dung như sau:



Ảnh 17: Mô tả ảnh dưới dạng pixel

Do đó, việc thực hiện các bước như Convolutinal Layer, Pooling Layer hay Flattening sẽ được thực hiện gần như mô tả trong lý thuyết.

Trong Python, thư viện Keras cung cấp lớp Sequential để giúp chúng ta xây dựng mô hình bằng cách xếp chồng các lớp một cách tuần tự. Điều này rất phù hợp với việc xây dựng các mô hình CNN, vì các lớp trong CNN cũng được kết nối theo một cách tuần tự: từ lớp đầu vào cho đến lớp đầu ra.

1. <u>Lớp Conv2D:</u>

Trong mô hình CNN, lớp Conv2D giúp học các đặc trưng từ ảnh đầu vào.

Cấu trúc của lớp Conv2D trong mô hình được sử dụng như sau:

```
Conv2D(filters, kernel_size, activation, input_shape)
```

- filters: Số lượng bộ lọc được sử dụng trong lớp Conv2D. Mỗi bộ lọc học một đặc trưng riêng biệt từ ảnh đầu vào. Ví dụ, nếu filters=32, thì sẽ có 32 bộ lọc.
- kernel_size: Kích thước của bộ lọc (hay kernel). Thông thường, các kernel có

kích thước nhỏ như 3x3 hoặc 5x5 sẽ được sử dụng. Bộ lọc sẽ quét qua ảnh và thực hiện phép toán tích chập tại mỗi điểm.

- activation: Hàm kích hoạt được áp dụng cho kết quả của phép toán tích chập.

  Hàm kích hoạt phổ biến là ReLU.

- input_shape: Kích thước của ảnh đầu vào, ví dụ ảnh có kích thước 200x200 và là ảnh đơn sắc (1 kênh màu).

Mỗi bộ lọc (filter) sẽ được di chuyển qua ảnh và tính toán phép toán tích chập với các giá trị pixel của ảnh. Mỗi filter khác nhau sẽ có các tác dụng như phát hiện các đặc trưng như cạnh, kết cấu, đường viền, màu sắc, v.v.

Một số filter phổ biến được dùng trong mô hình CNN như:

- Bộ lọc phát hiện cạnh (Sobel)
- Bộ lọc phát hiện góc (Harris)
- Bộ lọc làm sắc nét

Ngoài ra còn 1 số bộ lọc khác Gaussian, Laplacian,…. các bộ lọc không cần được định nghĩa thủ công mà sẽ được học trong quá trình huấn luyện. Khi mạng học, các bộ lọc sẽ tự động điều chỉnh để phát hiện các đặc trưng phù hợp nhất với dữ liệu huấn luyện.

Mô hình sẽ trải qua 3 lần Conv2D, dưới đây là tóm tắt các lớp, cùng với kích thước đầu ra và số lượng tham số:

Tên Lớp	Output Shape	Số tham số	Mô tả
Conv2D (1)	(150, 150, 32)	320	32 bộ lọc, kernel 3×3.
Conv2D (2)	(75, 75, 64)	18,496	64 bộ lọc, kernel 3×3
Conv2D (3)	(38, 38, 64)	36,928	63 bộ lọc, kernel 3×3
Conv2D (4)	(19, 19, 128)	73,856	128 bộ lọc, kernel 3x3
Conv2D (5)	(10, 10, 256)	295,168	256 bộ lọc, kernel 3x3

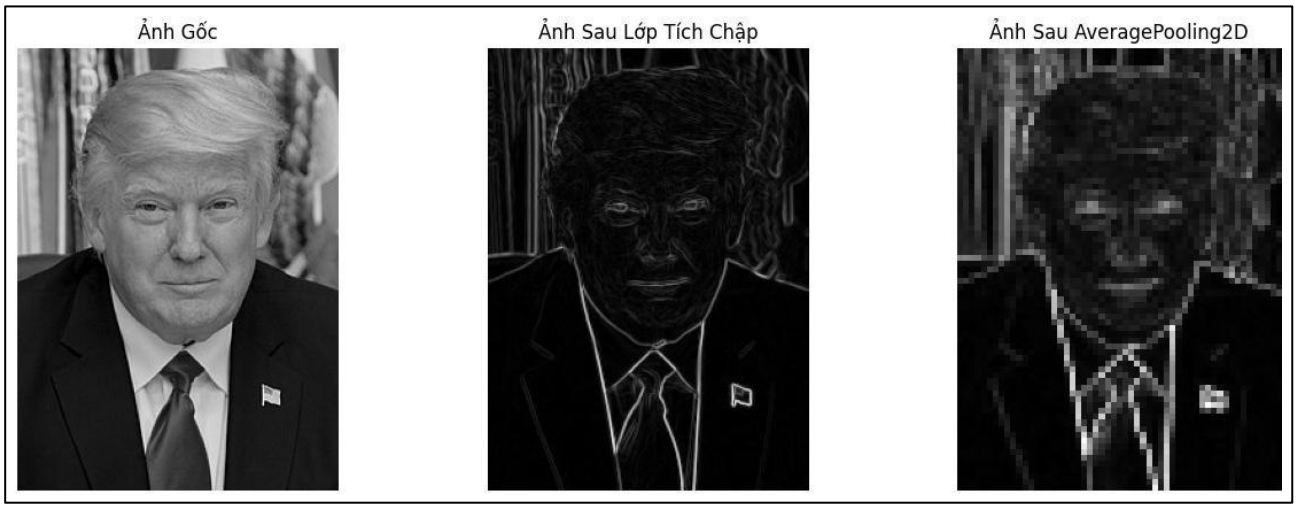
Trong mô hình huấn luyện, Các lớp Conv2D sử dụng bộ lọc với số lượng tăng dần (64, 128, 256) nhằm giúp mô hình học được các đặc trưng phức tạp hơn của ảnh khi đi qua các lớp sâu hơn. Tăng số lượng bộ lọc khi đi sâu vào mô hình giúp mô hình nhận diện được các đặc trưng đa dạng hơn và có khả năng phân loại chính xác

hơn.

2. Lớp Pooling:

Mô hình huấn luyện sẽ thực hiện 2 kỹ thuật pooling chính: **AveragePooling2D** và **GlobalAveragePooling2D**. Đây là các lớp giúp giảm số lượng tham số và kích thước không gian của dữ liệu.

Có pool_size = (2,2), nghĩa là mỗi nhóm 4 pixel sẽ được thay thế bằng giá trị trung bình của chúng. Đây là ví dụ của ảnh sau khi đã trải qua lớp tích chập và Average Pooling:



Ảnh 18: Ảnh sau khi thực hiện Pooling

Sau mỗi lớp Conv2D, việc giảm kích thước qua pooling giúp giữ lại thông tin quan trọng nhất và loại bỏ các chi tiết dư thừa, chuẩn bị cho các tầng sâu hơn học các đặc trưng trừu tượng.

3. Lớp Dense:

Sau khi trích xuất các đặc trưng từ ảnh qua các lớp Conv2D và Pooling, lớp Dense với 132 nơ-ron được sử dụng để kết hợp các đặc trưng này lại và giảm dần số lượng nơ-ron. Lớp này giúp mô hình học các mối quan hệ phức tạp giữa các đặc trưng đã học và chuyển chúng thành các giá trị có thể phân loại.Cấu trúc của lớp Dense như sau:

- Input: Dữ liệu đầu vào được đưa vào lớp Dense dưới dạng một vector 1 chiều sau khi được flattening.
- Weights: Mỗi đầu vào sẽ có một trọng số (weight) liên kết với nó. Trọng số

này sẽ được điều chỉnh trong quá trình huấn luyện để tối ưu hóa mô hình.

- Biases: Mỗi nút trong lớp Dense có một giá trị bias (độ lệch) để điều chỉnh kết quả đầu ra.
- Sau đó hàm kích hoạt sẽ được áp dụng để tạo ra đầu ra cuối cùng: Output = Activation(W.X+b).

Trong quá trình huấn luyện, các trọng số và bias trong lớp Dense được điều chỉnh thông qua các thuật toán tối ưu hóa như Gradient Descent và thuật toán Backpropagation. Mục tiêu là giảm thiểu hàm mất mát (loss function) để tối ưu hóa mô hình, giúp dự đoán chính xác hơn.

Đầu ra của mô hình này sẽ là một giá trị duy nhất, đại diện cho xác suất của mẫu thuộc lớp "Pneumonia" hay “Normal” sau khi áp dụng hàm kích hoạt sigmoid. Ví dụ, trong mô hình dự đoán viêm phổi này, đầu ra có thể là một giá trị như sau: 0.85

p = 0.85: Xác suất mẫu thuộc lớp "Pneumonia" là 85%.

1 - p = 0.15: Xác suất mẫu thuộc lớp "Normal" là 15%.

Dựa vào giá trị xác suất này, mô hình sẽ phân loại mẫu vào lớp có xác suất cao nhất. Trong trường hợp này, nếu xác suất là 0.85, mẫu sẽ được phân loại vào lớp "Pneumonia".

2.4 Tổng kết mô hình

Tổng kết lại, mô hình CNN sẽ gồm các lớp sau:

1. Conv2D và MaxPooling: Các lớp tích chập (Conv2D) sử dụng kernel kích thước 3x3 để phát hiện các đặc trưng trong ảnh, kết hợp với padding 'same' giúp duy trì kích thước ảnh đầu vào sau mỗi lớp tích chập. MaxPooling (2x2) giúp giảm độ phân giải của ảnh, đồng thời giữ lại các đặc trưng quan trọng, giảm thiểu chi phí tính toán và làm giảm nguy cơ overfitting. Mô hình sử dụng ba lớp Conv2D với số lượng bộ lọc tăng dần (32, 64, 128, 256).

2. BatchNormalization: Sau mỗi lớp tích chập, BatchNormalization được sử dụng để chuẩn hóa đầu ra của các lớp, giúp cải thiện sự hội tụ của mô hình và làm giảm hiện tượng overfitting.

3. Dropout: Các lớp Dropout được thêm vào với tỷ lệ 0.1, 0.2 nhằm ngăn chặn overfitting bằng cách ngẫu nhiên "bỏ qua" một phần đơn vị trong mạng trong quá trình huấn luyện, giúp mô hình học được các đặc trưng mạnh mẽ hơn.

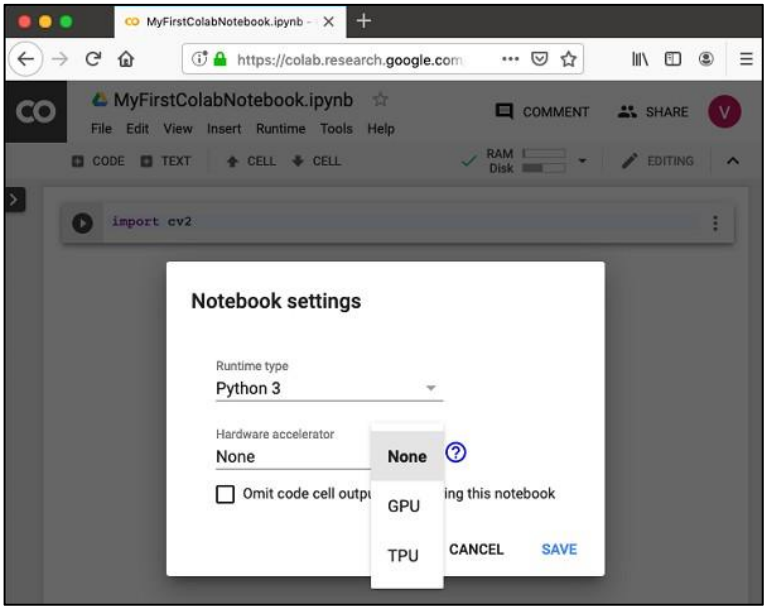
4. Flatten và Dense: Sau các lớp tích chập và pooling, lớp Flatten chuyển đổi dữ liệu đầu vào thành một vector một chiều. Lớp Dense với 128 đơn vị và hàm kích hoạt ReLU giúp học các đặc trưng cao cấp từ dữ liệu. Cuối cùng, lớp Dense với 1 đơn vị và hàm kích hoạt sigmoid cung cấp kết quả phân loại nhị phân.

5. Hàm mất mát: Mô hình sử dụng hàm mất mát binary_crossentropy, phù hợp với bài toán phân loại nhị phân.

CHƯƠNG 3: THỰC THI MÔ HÌNH

3.1 Yêu cầu thiết bị

Việc huấn luyện mô hình từ đầu đòi hỏi cấu hình máy tính cao, đặc biệt là phải có GPU. Ta có thể sử dụng Google Colab Pro nếu thiết bị không đáp ứng được nhu cầu. Ngoài ra dung lượng RAM 16GB hoặc cao hơn sẽ giúp tối ưu hóa quá trình xử lý và tránh quá tải.



Ảnh 19 GPU trên Google Colab

Sau đó ta có thể chạy đoạn lệnh sau để kiểm tra liệu thiết bị có đang chạy mô hình trên GPU hay không:

```
# Kiểm tra GPU

device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

Ảnh 20 Kiểm tra GPU

3.2 Tệp dữ liệu

Sẽ có 3 tệp dữ liệu đầu vào, trong đó:

- Tập huấn luyện (train): Tập dữ liệu này sẽ được sử dụng để huấn luyện mô hình. Mô hình sẽ học các đặc trưng từ dữ liệu trong tập huấn luyện này và điều chỉnh các tham số (weights) của mạng neural.

- Tập validation: Tập dữ liệu này sẽ được sử dụng trong quá trình huấn luyện để kiểm tra và đánh giá hiệu suất của mô hình sau mỗi epoch (vòng lặp huấn luyện). Tập này không tham gia vào việc cập nhật tham số của mô hình.

- Tập kiểm tra (test): Sau khi mô hình đã được huấn luyện và tinh chỉnh, tập kiểm tra sẽ được sử dụng để đánh giá hiệu suất cuối cùng của mô hình trên dữ liệu mà mô hình chưa thấy trước đó.

3.3 Thực hiện chương trình

Đầu tiên, chuẩn bị các thư viện cần thiết cho mô hình

```
import matplotlib.pyplot as plt
import seaborn as sns
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout , BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from keras.callbacks import ReduceLROnPlateau
import cv2
import os
```

Ảnh 20: Thư viện cần thiết

- Tensorflow: là thư viện mã nguồn mở của Google, giúp xây dựng và huấn luyện các mô hình học sâu.
- Sequential: là mô hình tuyến tính, trong đó các lớp được sắp xếp tuần tự và mỗi lớp sẽ nhận đầu vào từ lớp trước và truyền ra đầu ra cho lớp tiếp theo.

- Các mạng nơ-ron như Dense Layer, Conv2D(lớp tích chập), các lớp Pooling làm giảm kích thước không gian của ảnh.
- 1 số thư viện toán học và các thư viện hỗ trợ trong việc trực quan hóa dữ liệu.

Sau khi đã thực hiện việc import dữ liệu đã được chuẩn bị, sẽ thực hiện việc chuẩn hóa ảnh về khoảng [0,1].

```
# Chuẩn hóa dữ liệu
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255
x_test = np.array(x_test) / 255
```

Ảnh 21: Chuẩn hóa dữ liệu

Tiếp theo thực hiện việc tăng cường dữ liệu của tập train:

```
datagen = ImageDataGenerator(
        rotation_range=30,  # xoay ngẫu nhiên ảnh trong phạm vi (độ, từ 0 đến 180)
        zoom_range=0.2,  # zoom ngẫu nhiên ảnh
        width_shift_range=0.1,  # di chuyển ngẫu nhiên ảnh theo chiều ngang (tính theo tỷ lệ của chiều rộng tổng thể)
        height_shift_range=0.1,  # di chuyển ngẫu nhiên ảnh theo chiều dọc (tính theo tỷ lệ của chiều cao tổng thể)
        horizontal_flip=True)  # lật ngẫu nhiên ảnh theo chiều ngang

datagen.fit(x_train)
```

Ảnh 22: Tăng cường dữ liệu tập train

Thực hiện xây dựng mô hình CNN:

```
model = Sequential()
model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'
                 , input_shape = (150,150,1)))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.1))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(128 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(256 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Flatten())
model.add(Dense(units = 128 , activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(units = 1 , activation = 'sigmoid'))
model.compile(optimizer = "rmsprop" , loss = 'binary_crossentropy' , metrics = ['accuracy'])
model.summary()
```

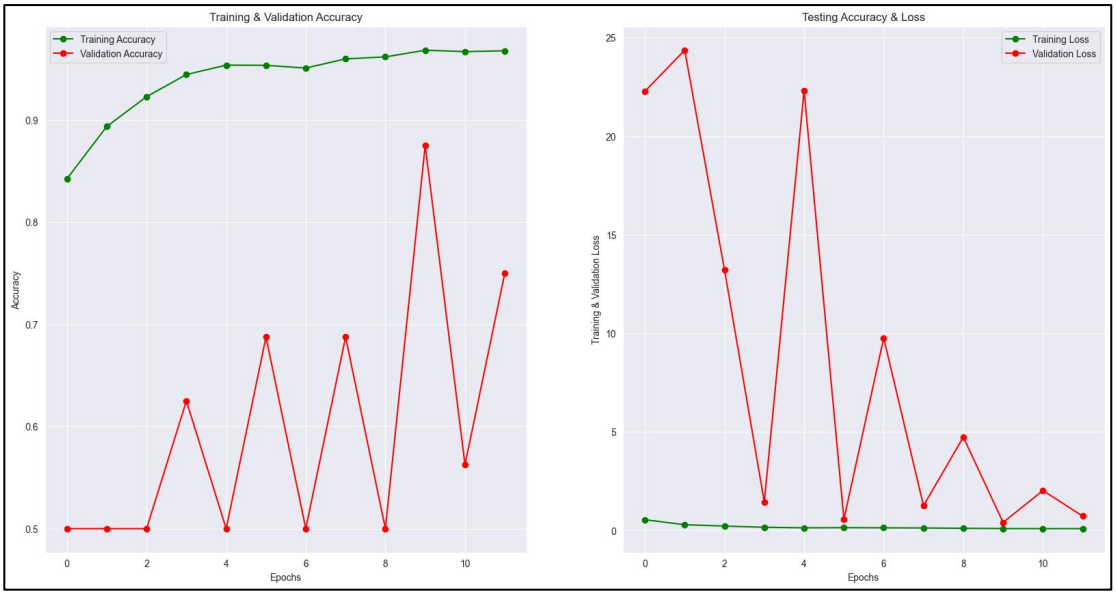
Ảnh 23: Mã nguồn mô hình CNN

3.4 Kết quả và đánh giá

Với mô hình như trên chúng ta sẽ thu được:

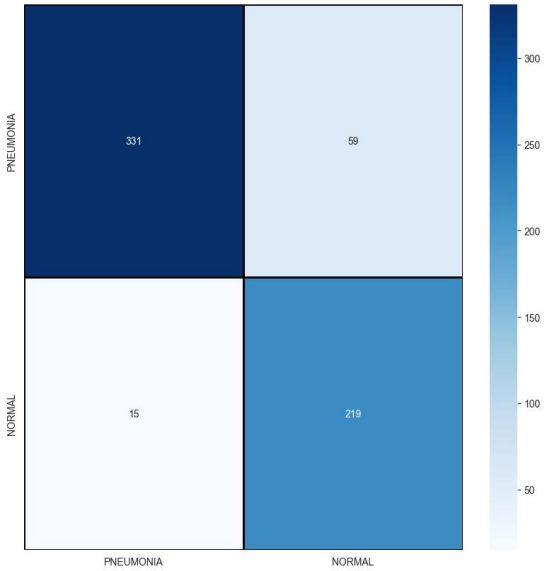
- Accuracy = 88.14%
- Loss = 0.33

Ta có thể sử dụng các thư viện để vẽ biểu đồ đánh giá Accuracy (độ chính xác) và Loss (mất mát) của mô hình trên từng epochs:



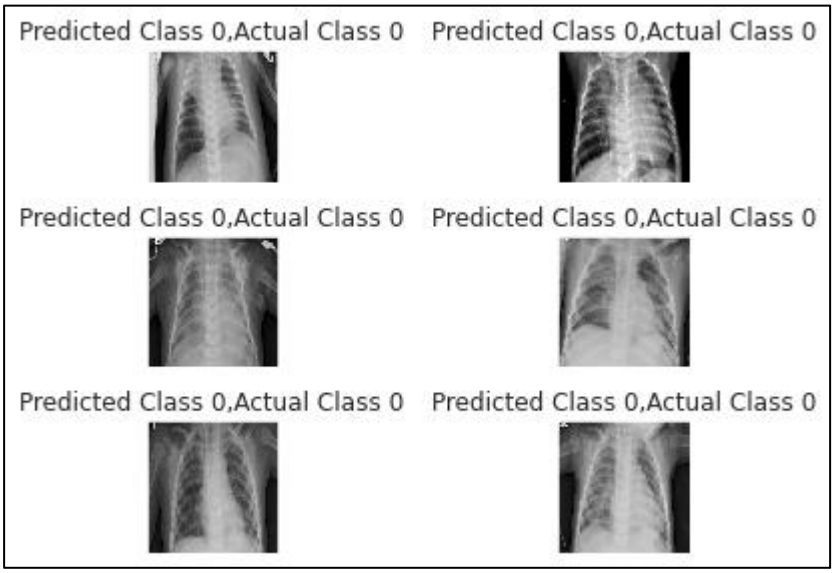
Ảnh 24: Biểu đồ quá trình huấn luyện

Confusion matrix của mô hình:



Ảnh 25: Confusion matrix

Tiến hành dự đoán với 1 số ảnh trong tập test, mô hình đã cho ra kết quả khá tốt:



Ảnh 26: Dự đoán ảnh trên tập test

Đánh giá tổng quan:

- Mô hình đã cho ra accuracy khá tốt trên tập train và validation, khi thực hiện dự đoán 1 số ảnh thực tế trên tập test, mô hình cũng cho ra kết quả tốt.
- Mô hình thể hiện khả năng nắm bắt đặc trưng tốt từ tập huấn luyện, minh chứng qua sự giảm đều đặn của train loss và sự gia tăng train accuracy trong quá trình huấn luyện.
- Loss của mô hình là 0.29357. Giá trị này cho thấy mức độ sai lệch giữa dự đoán của mô hình và giá trị thực tế. Một giá trị loss nhỏ cho thấy mô hình đang học được khá tốt, tức là sự khác biệt giữa dự đoán và giá trị thực tế là không quá lớn.
- Accuracy của mô hình là 92.63%. Đây là một chỉ số khá cao, cho thấy mô hình dự đoán đúng được khoảng 92.6% số lượng mẫu thử. Đây là một dấu hiệu tốt, cho thấy mô hình hoạt động khá hiệu quả trên bộ dữ liệu test.

LỜI KẾT

Trong báo cáo này, đã mô tả các bước thực hiện áp dụng mạng nơ-ron tích chập để để phát hiện viêm phổi. Viêm phổi là một căn bệnh nguy hiểm, và việc phát hiện sớm là rất quan trọng để điều trị kịp thời. Vì vậy, việc sử dụng công nghệ để hỗ trợ chẩn đoán bệnh là một hướng đi tiềm năng và có ích.

Qua quá trình thử nghiệm, mô hình CNN đã cho kết quả khá ấn tượng với độ chính xác cao trong việc phân loại hình ảnh phổi, từ đó giúp phát hiện viêm phổi với hiệu quả vượt trội. Các kỹ thuật tiền xử lý dữ liệu như chuẩn hóa ảnh, tăng cường dữ liệu (data augmentation) và batch normalization đã giúp cải thiện đáng kể hiệu suất mô hình, giảm thiểu tình trạng overfitting và đảm bảo tính tổng quát của mô hình khi áp dụng vào các dữ liệu thực tế.

Mô hình CNN đã triển khai đã cho thấy khả năng học và phân loại hình ảnh hiệu quả, với độ chính xác đạt được là 92.63%. Việc áp dụng các kỹ thuật tăng cường dữ liệu và chuẩn hóa ảnh đã giúp cải thiện độ chính xác và làm giảm hiện tượng overfitting, qua đó tạo ra một mô hình hiệu quả hơn khi đối mặt với dữ liệu mới.

Nhìn chung, dự án này không chỉ giúp chúng em hiểu rõ hơn về các thuật toán trong học sâu, mà còn làm rõ tầm quan trọng của việc ứng dụng AI trong lĩnh vực y tế. Chúng em tin rằng trong tương lai, công nghệ AI sẽ giúp đẩy nhanh việc phát triển các công cụ hỗ trợ y tế, làm tăng hiệu quả điều trị và chăm sóc sức khỏe cộng đồng.

Cuối cùng, chúng em xin gửi lời cảm ơn sâu sắc đến các thầy cô, anh chị trợ giảng đã hướng dẫn, tạo điều kiện cho chúng em hiểu biết thêm về môn học trong suốt quá trình giảng dạy. Chúng em hy vọng rằng những kết quả đạt được trong dự án sẽ có ích cho cộng đồng và có thể trở thành bước đệm để phát triển các ứng dụng AI trong y tế.

DANH MỤC TÀI LIỆU THAM KHẢO

1. An Introduction to Convolutional Neural Networks (CNNs). Nguồn:

https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns

2. Detection of pneumonia using convolutional neural networks and deep learning. Nguồn:

https://www.sciencedirect.com/science/article/pii/S0208521622000742

3. Medical Diagnosis with CNN& Transfer Learning. Nguồn:

https://www.kaggle.com/code/homayoonkhadivi/medical-diagnosis-with-cnn-transfer-learning

4. Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification. Nguồn:

https://data.mendeley.com/datasets/rscbjbr9sj/2