ĐẠI HỌC QUỐC GIA TPHCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

ĐỒ ÁN 1

XÁC ĐỊNH UNG THƯ DA QUA TẬP DỮ LIỆU HÌNH ẢNH

Giảng viên hướng dẫn : **Huỳnh Tuấn Anh**
Sinh viên thực hiện : **Lê Hoàng Thịnh**

Ho Chi Minh city, 25th December 2021

LỜI CẢM ƠN

Trân trọng gửi lời cảm ơn đến thầy Huỳnh Tuấn Anh, giảng viên khoa công nghệ phần mềm đã tạo điều kiện và cơ hội giúp đỡ em trong quá trình phát triển và hoàn thiện đồ án môn học nghiên cứu này. Trong thời gian qua, nhờ sự hướng dẫn từ thầy, em đã có được những kiến thức quan trọng và nhất định để hoàn thiện báo cáo một cách hoàn chỉnh nhất. Tuy nhiên, trong quá trình tìm hiểu và hoàn thiện đồ án, với thời gian, kiến thức và kinh nghiệm còn hạn chế nên khó có thể tránh được những sai sót, em mong nhận được những ý kiến đóng góp từ thầy để tích lũy thêm những bài học, kinh nghiệm để hoàn thành tốt hơn với đồ án tiếp theo. Em xin chân thành cảm ơn.

Lê Hoàng Thịnh

Table of contents

1. MỞ ĐẦU

1.1. Lý do chọn đề tài

Với sự phát triển mạnh mẽ của thời đại khoa học công nghệ, bên cạnh những ứng dụng được tạo ra cho mục đích đời sống hằng ngày, sức mạnh tính toán của máy tính còn được ứng dụng rộng rãi trong nhiều lĩnh vực nghiên cứu như nhận dạng ảnh, dự báo thời tiết, chẩn đoán y khoa …

Một trong những thay đổi mang tính thời đại là sự ứng dụng của phương pháp pháp học máy và phương pháp học sâu để áp dụng trong lĩnh vực thị giác máy tính.

Nhận thấy được tầm quan trọng và tính phổ biến của các phương pháp học tri thức trong thực tế, kèm với mong muốn được hỗ trợ cũng như giúp đỡ các chuyên gia y tế trong việc chẩn đoán bệnh, em đã quyết định chọn đề tài nghiên cứu về xác định ung thư da qua tập hình ảnh.

1.2. Mô tả bài toán

Bài toán bắt đầu bằng việc chuyên gia y tế sẽ chụp tấm ảnh trên da của bệnh nhân, dựa trên hình ảnh và các sắc tố da… để nhận dạng được loại ung thư này là lành tính hay ác tính. Vì lĩnh vực nghiên cứu liên quan đến y khoa nên yếu tố chính xác được đưa lên hàng đầu trước yếu tố về thời gian.

1.3. Mục tiêu nghiên cứu

Mục tiêu nghiên cứu của đồ án là khái quát về Machine Learning, sau đó là Deep Learning và sử dụng mạng neural để huấn luyện mô hình nhận diện ung thư da.

2. Máy học (Machine Learning)

Machine Learning là một tập con của AI (Artificial Intelligence – Trí tuệ nhân tạo). Nói đơn giản thì Machine Learning là một lĩnh vực nhỏ của Khoa học máy tính, nó có khả năng tự học hỏi dựa trên dữ liệu đưa vào mà không cần phải lập trình cụ thể. Machine Learning làm cho máy tính có những khả năng nhận thức cơ bản của con người như nghe, nhìn, hiểu được ngôn ngữ, giải toán, lập trình, … và hỗ trợ con người trong việc xử lý một khối lượng lớn thông tin khổng lồ mà chúng ta phải đối mặt hằng ngày, hay còn gọi là Big Data (dữ liệu lớn)

Nhìn chung có nhiều phương thức học tri thức, nhưng ta sẽ tập trung vào ba phương thức chính:

2.1. Học có giám sát (Supervised Learning)

Học có giám sát (Supervised Learning) là mô hình học máy sử dụng các thuật toán để dự đoán kết quả đầu ra (output) dựa trên các dữ liệu đầu vào. Mô hình này được huấn luyện để dự đoán có độ chính xác cao từ các cặp dữ liệu được thu thập trước (features và label). Trong đó:

- Feature: Thường được ký hiệu là x. Đây là một dạng tensor có nhiều chiều. x = (x1,x2,...,xn) trong đó với mỗi x_i được gọi là một feature.
- Label: Thường được ký hiệu là y. Đây là kết quả của một tập thuộc tính x tương ứng. Đây còn là output của bài toán.
- Một cách toán học, học có giám sát là khi chúng ra có một tập hợp biến đầu vào $X = \{x_1, x_2, \dots x_n\}$ và một tập hợp nhãn tương ứng $Y = \{y_1, y_2, \dots y_n\}$. Trong đó x_i, y_i là các vector. Các cặp dữ liệu (x_i, y_i) ta đã biết trước được gọi là dữ liệu huấn luyện (training data)

$$y_i \approx f(x_i), \forall i = 1,2, \dots N$$

Mục đích của việc học không giám sát nói chung là ta tìm được hàm f sao cho với mỗi giá trị x_i ta tìm được giá trị y_i sao cho sai số so với thực tế là nhỏ nhất.

Ví dụ: trong nhận dạng chữ viết tay, ta có ảnh của hàng nghìn ví dụ của mỗi chữ số được viết bởi nhiều người khác nhau. Chúng ta đưa các bức ảnh này vào trong một thuật toán và chỉ cho nó biết mỗi bức ảnh tương ứng với chữ số nào. Sau khi thuật toán tạo ra một mô hình, tức một hàm số mà đầu vào là một bức ảnh và đầu ra là một chữ số, khi nhận được một bức ảnh mới mà mô hình **chưa nhìn thấy.**

Nhìn chung học có giám sát được chia ra thành hai bài toán khác nhau dựa trên kết quả đầu vào và ra của chúng

2.2. Phân loại bài toán dựa trên phương thức học

2.2.1. Học có giám sát (Supervised Learning)

- Hồi quy (Regression)

Nếu *nhãn* không được chia thành các nhóm mà là một giá trị thực cụ thể. Ví dụ: một căn nhà rộng
Một bài toán được gọi là *hồi quy* nếu các *nhãn* của *dữ liệu đầu vào không được chia thành các nhóm hữu hạn mà là một giá trị thực cụ thể*. Ví dụ: dự đoán giá nhà dựa trên vị trí và diện tích là một bài toán hồi quy.

- Phân loại (Classification)

Một bài toán được gọi là *classification* nếu các *nhãn* của *dữ liệu đầu vào* được chia thành một số hữu hạn nhóm. Ví dụ: Gmail xác định xem một email có phải là spam hay không; các hãng tín dụng xác định xem một khách hàng có khả năng thanh toán nợ hay không

2.2.2. Học không giám sát (Unsupervised Learning)

Đối với học không giám sát, chúng ta không biết được *giá trị đầu ra* hay *nhãn* mà chỉ có dữ liệu đầu vào. Thuật toán học không giám sát sẽ dựa vào cấu trúc của dữ liệu để thực hiện một công việc nào đó, ví dụ như phân nhóm (clustering) hoặc giảm số chiều của dữ liệu (dimension reduction) để thuận tiện trong việc lưu trữ và tính toán.

Ví dụ cho việc học không giám sát: tính năng phân loại theo chủ đề và sự kiện của Google là một bài toán học không giám sát.

2.2.3. Học bán giám sát

Học bán giám sát là mô hình học kết hợp từ học có giám sát và học không giám sát bằng việc gán nhãn cho một số dữ liệu và một số dữ liệu thì lại không được gán nhãn trong đó số lượng dữ liệu không được gán nhãn thường lớn hơn nhiều so với dữ liệu được gán nhãn. Các dữ liệu 8 này được kết hợp với nhau để tận dụng hiệu quả dữ liệu có được. Thông thường, điều này sẽ làm tăng tính chính xác của mô hình xây dựng.

2.2.4. Học tăng cường (Reinforcement Learning)

Reinforcement learning là các bài toán giúp cho một hệ thống tự động xác định hành vi dựa trên hoàn cảnh để đạt được lợi ích cao nhất (maximizing the performance). Hiện tại, Reinforcement learning chủ yếu được áp dụng vào Lý Thuyết Trò Chơi (Game Theory), các thuật toán cần xác định nước đi tiếp theo để đạt được điểm số cao nhất. Một ví dụ của học tăng cường là robot Alphago đã chiến thắng áp đảo kỳ thủ cờ vây hàng đầu thế giới.

2.3. Phân loại bài toán dựa trên thuật toán

Một số nhóm thuật toán tiêu biểu sẽ được đề cập sau đây sẽ được gom nhóm, ngoài ra sẽ còn nhiều thuật toán khác

2.3.1. Thuật toán hồi quy

- Linear Regression
- Logistic Regression
- Stepwise Regression

2.3.2. Thuật toán phân lớp

- Linear Classifier
- Support Vector Machine (SVM)
- Kernel SVM

2.3.3. Thuật toán mạng neural di truyền

- Perception
- Softmax Regression
- Multilayer Perceptron
- Back-propagation

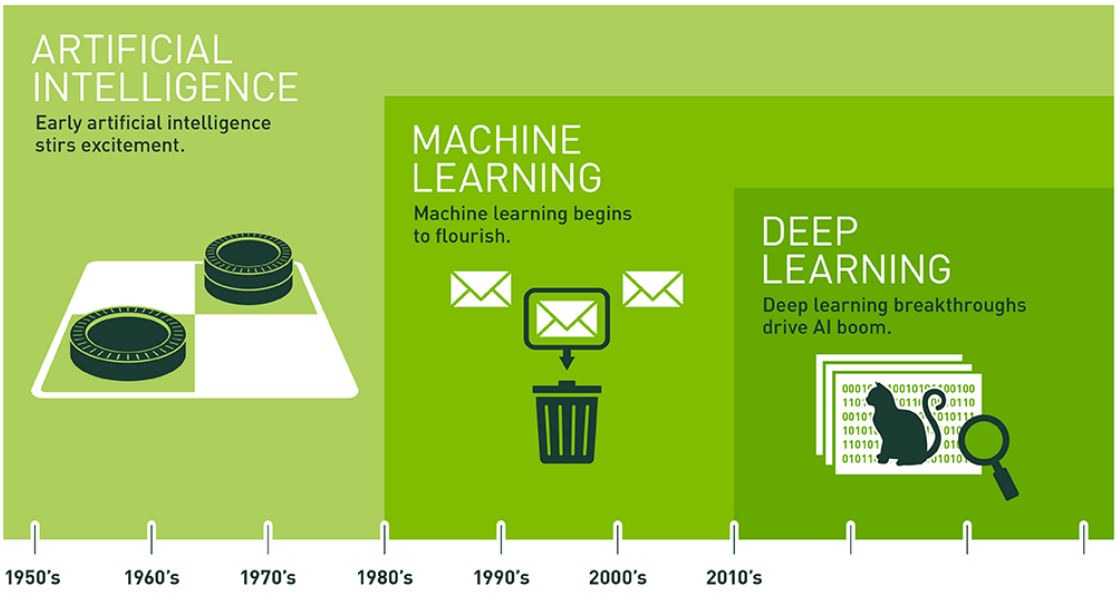
2.3.4. Thuật toán gom nhóm

- K-mean clustering
- K-medians

3. Học sâu (Deep Learning)

3.1. Giới thiệu về học sâu

Nhìn chung trí tuệ nhân tạo (AI), học máy (Machine Learning) và học sâu (Deep Learning) có mối liên hệ với nhau giống như hình được mô tả dưới



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Deep Learning là một tập hợp con của Machine Learning, có khả năng khác biệt ở một số khía cạnh quan trọng so với Machine Learning nông truyền thống, cho phép máy tính giải quyết một loạt các vấn đề phức tạp không thể giải quyết được.

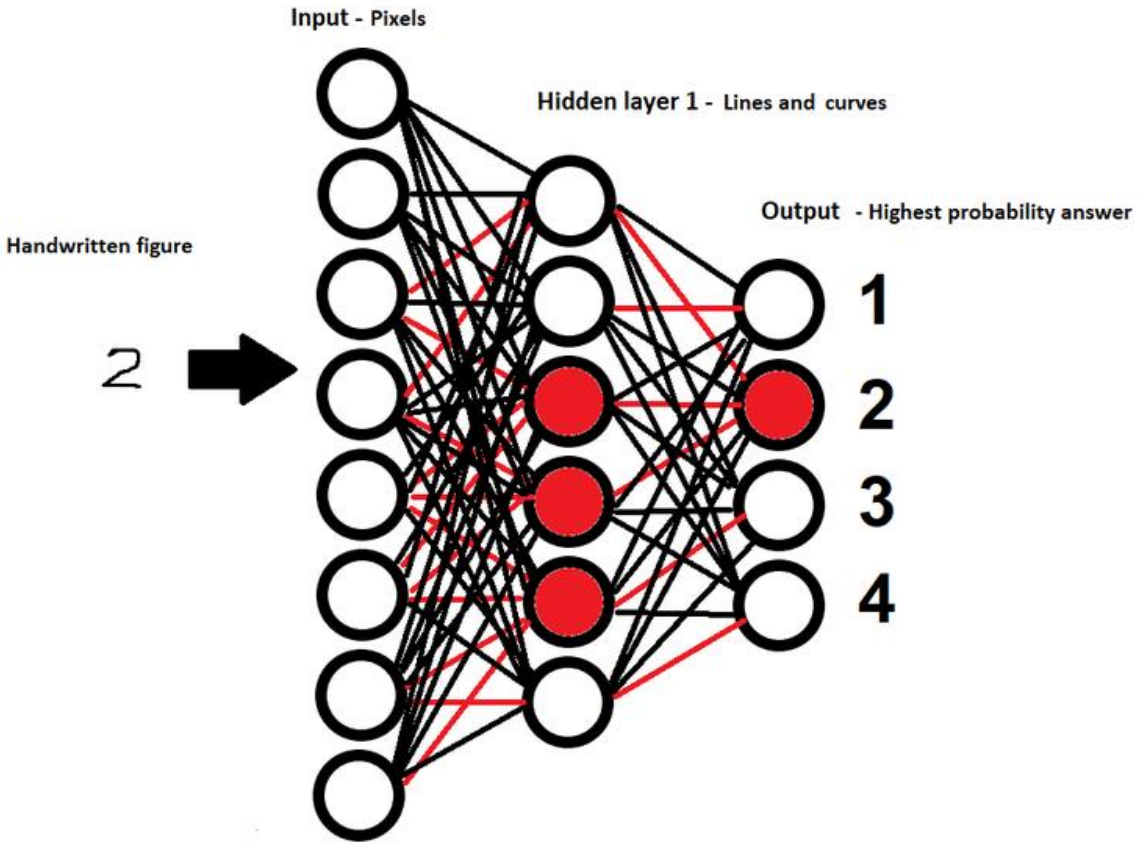
Đối với machine learning, bài toán được giải quyết dựa trên một số lượng tham số đầu vào có cấu trúc và dự đoán được kết quả đầu ra. Vấn đề là hàng loạt vấn đề trong thế giới thực không phù hợp với những mô hình đơn giản như vậy. Một ví dụ về một trong những vấn đề thực tế phức tạp này là nhận ra các số viết tay.

Để giải quyết vấn đề này, máy tính cần phải có khả năng đối phó với sự đa dạng lớn trong cách thức trình bày dữ liệu. Mỗi chữ số từ 0 đến 9 có thể được viết theo vô số cách: kích thước và hình dạng chính xác của mỗi chữ số viết tay có thể rất khác nhau tùy thuộc vào người viết và trong hoàn cảnh nào, góc chụp nào…

Đối phó với sự biến đổi của các features này và sự lộn xộn tương tác lớn hơn giữa chúng, tạo ra vô số khả năng chính là nơi học tập sâu và mạng lưới thần kinh sâu trở nên hữu ích.

Mạng lưới thần kinh là các mô hình toán học có cấu trúc được lấy cảm hứng từ các liên kết neural trong bộ não người.

Mỗi nơ-ron trong mạng nơ-ron là một hàm toán học lấy dữ liệu thông qua đầu vào, biến đổi dữ liệu đó thành dạng dễ điều chỉnh hơn và sau đó phun ra thông qua đầu ra. Bạn có thể nghĩ về các nơron trong một mạng lưới thần kinh như được sắp xếp theo lớp, như hình dưới đây.



Tất cả các mạng thần kinh đều có một lớp đầu vào, trong đó dữ liệu ban đầu được đưa vào và một lớp đầu ra, tạo ra dự đoán cuối cùng. Nhưng trong một mạng lưới thần kinh sâu, sẽ có nhiều "lớp tế bào" ẩn giữa các lớp đầu vào và đầu ra (gọi là Hidden Layer), mỗi lớp cho dữ liệu vào nhau. Do đó, thuật ngữ "Deep" trong "Deep Learning" và "mạng lưới thần kinh sâu", nó liên quan đến số lượng lớn các lớp ẩn – thường lớn hơn 3 – tại trung tâm của các mạng thần kinh này.

Trong sơ đồ trên, mỗi vòng tròn đại diện cho một neural trong mạng, với các neural được tổ chức thành các lớp thẳng đứng. Mỗi neural được liên kết với mọi nơ-ron ở lớp sau (đây còn được gọi là Dense layer), thể hiện thực tế là mỗi neural tạo ra một giá trị vào mỗi neural ở lớp tiếp theo. Màu sắc của các liên kết trong sơ đồ cũng khác nhau. Các màu khác nhau, đen và đỏ, thể hiện tầm quan trọng của các liên kết giữa các neural (trọng số). Các liên kết màu đỏ là những liên kết có ý nghĩa lớn hơn, có nghĩa là chúng sẽ khuếch đại giá trị khi nó đi qua giữa các lớp. Đổi lại, sự khuếch đại giá trị này có thể giúp kích hoạt tế bào thần kinh mà giá trị đang được đưa vào.

Như hình trên ta có thể thấy, đầu ra của một lớp là đầu vào của lớp tiếp theo trong mạng, với dữ liệu chảy qua mạng từ đầu vào đến đầu ra.

Nhiều lớp tế bào thần kinh này về cơ bản cung cấp một cách để mạng lưới thần kinh xây dựng một hệ thống phân cấp thô gồm các tính năng khác nhau tạo nên chữ số viết tay trong câu hỏi. Chẳng hạn, nếu đầu vào là một mảng các giá trị đại diện cho các pixel riêng lẻ trong hình ảnh của hình viết tay, lớp tiếp theo có thể kết hợp các pixel này thành các đường và hình dạng, lớp tiếp theo kết hợp các hình dạng đó thành các đặc điểm riêng biệt như các vòng lặp trong 8 hoặc tam giác trên trong 4, và như vậy. Bằng cách xây dựng một bức tranh về các tính năng này, các mạng thần kinh hiện đại có thể xác định – với độ chính xác rất cao – con số tương ứng với một chữ số viết tay. Tương tự, các loại mạng thần kinh sâu khác nhau có thể được đào tạo để nhận diện khuôn mặt trong hình ảnh hoặc để phiên âm lời nói bằng văn bản.

Ví dụ, trong trường hợp mô hình nhận dạng chữ số viết tay, các trọng số này có thể được sửa đổi để nhấn mạnh tầm quan trọng của một nhóm pixel cụ thể tạo thành một dòng hoặc một cặp các đường giao nhau tạo thành 7.

3.2. Các thành phần của mạng học sâu

Mô hình học được các liên kết giữa các neural rất quan trọng trong việc đưa ra dự đoán thành công trong quá trình đào tạo.

3.2.1. Hàm kích hoạt (activation function)

Ở mỗi bước trong quá trình đào tạo, mạng sẽ sử dụng một hàm toán học để xác định mức độ chính xác của dự đoán mới nhất của nó so với dự kiến (activation). Hàm này tạo ra một loạt các giá trị lỗi, do đó hệ thống có thể sử dụng để tính toán cách mô hình nên cập nhật giá trị của các trọng số được gắn vào mỗi liên kết, với mục đích cuối cùng là cải thiện độ chính xác của các dự đoán của mạng.

3.2.2. Hàm tối ưu hóa (optimizer)

Mức độ mà các giá trị trên sẽ được thay đổi được tính bởi một chức năng tối ưu hóa (optimizer), chẳng hạn như giảm độ dốc và những thay đổi đó được đẩy lùi trên toàn mạng vào cuối mỗi chu kỳ đào tạo trong một bước gọi là lan truyền ngược (back-propagation)

Trải qua nhiều, rất nhiều vòng huấn luyện (epochs) và với sự trợ giúp của việc điều chỉnh tham số thủ công không thường xuyên, mạng sẽ tiếp tục dự đoán tốt hơn cho đến khi nó đạt gần với độ chính xác cao nhất. Tại thời điểm này, ví dụ, khi các chữ số viết tay có thể được nhận ra với độ chính xác hơn 97%, mô hình Deep Learning có thể nói là đã được huấn luyện thành công.

3.3. Tóm tắt

Về cơ bản, Deep Learning cho phép Machine Learning giải quyết một loạt các vấn đề phức tạp mới – chẳng hạn như nhận dạng hình ảnh (Computer Vision), ngôn ngữ và lời nói (NLP) – bằng cách cho phép máy móc tìm hiểu cách các tính năng trong dữ liệu kết hợp thành các dạng trừu tượng ngày càng cao hơn. Ví dụ: trong nhận dạng khuôn mặt, cách các pixel trong hình ảnh tạo ra các đường và hình dạng, cách các đường và hình dạng đó tạo ra các đặc điểm khuôn mặt và cách các đặc điểm khuôn mặt này được sắp xếp thành một khuôn mặt.

3.3.1. Khi nào nên sử dụng phương pháp học sâu

Khi dữ liệu thu được phần lớn không có cấu trúc và có rất nhiều dữ liệu.

Các thuật toán Deep Learning có thể lấy dữ liệu lộn xộn và không có nhãn rộng rãi – chẳng hạn như video, hình ảnh, bản ghi âm thanh và văn bản – và áp đặt đủ thứ tự cho dữ liệu đó để đưa ra dự đoán hữu ích.

3.3.2. Khi nào không nên sử dụng phương pháp học sâu

Đối với các tác vụ không tầm thường, việc đào tạo một mạng lưới thần kinh sâu thường sẽ yêu cầu xử lý một lượng lớn dữ liệu bằng cách sử dụng các cụm GPU cao cấp trong nhiều, nhiều giờ.

4. Mạng Neural Tích Chập (Convolutional Neural Network)

4.1. Sơ lược

Với sự phát triển mạnh mẽ của các thuật toán học sâu ở những năm gần đây, Convolutional Neural Networks (CNNs) đã trở nên phổ biến và tạo ra những thay đổi mang tính "cách mạng" trong lĩnh vực thị giác máy tính nói chung và học tri thức nói chung. Một cách áp dụng điển hình của CNNs là để phân loại hình ảnh ví dụ như phân biệt giữa chó và mèo, phân biệt giữa các khối u của ung thư da…

4.2. Tại sao lại sử dụng mạng neural tích chập thay cho các mạng neural thông thường

4.2.1. Kích thước hình ảnh quá lớn

Thời đại công nghệ đang rất phát triển cả về phần mềm lẫn phần cứng, các thiết bị như camera ngày nay có thể chụp được những bức hình siêu nét với độ phân giải cực cao. Do đó khi huấn luyện hay ứng dụng một mô hình Thị giác máy tính, các Input có khả năng là những hình ảnh có kích thước rất lớn. Ví dụ, một bức ảnh có độ phân giải 1Mp sẽ có tới 1 triệu điểm ảnh, với 3 kênh màu thì số lượng dữ liệu cần xử lý ngay tại lớp đầu tiên sẽ là 3 triệu con số.

4.2.2. Sự thay đổi của vật thể dựa theo yếu tố bên ngoài

Ở một mô hình Neural Network thông thường, mỗi điểm ảnh sẽ kích hoạt các nơron khác nhau. Do đó việc thay đổi vị trí vật thể cần được xác định sẽ ảnh hưởng rất lớn đến Output, dẫn đến khả năng tối ưu kém, tốn nhiều tài nguyên để tối ưu thuật toán.

4.2.3. Cách mạng neural tích chập giải quyết vấn đề

CNNs sẽ đặt các điểm ảnh vào các bối cảnh mà tại đó chúng thể hiện được sự tương tác với các điểm ảnh liền kề như thế nào. Vì trong thực tế thì các điểm ảnh sẽ mang nhiều thông tin hữu ích hơn khi đặt nó kế những điểm ảnh khác. Từ đó ta sẽ rút trích được các đặc trưng mà ta có thể sử dụng được như các cạnh của vật thể.

4.3. Tầng tích chập (Convolutional Layer)

4.3.1. Phép tích chập

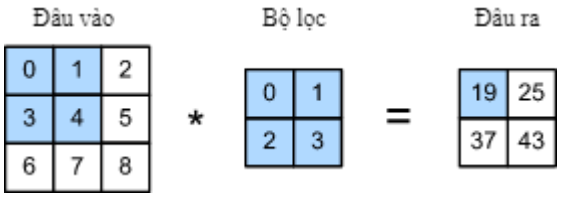
Như tên gọi của nó – CNNs (Mạng nơ-ron tích chập), phép tích chập là phép toán trọng yếu được sử dụng bởi thuật toán này. Tầng tích chập bao gồm các bộ lọc (filters) dưới dạng một ma trận 2 chiều. Ví dụ:

$$K = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Trước khi đến với định nghĩa của phép tích chập ta sẽ xem xét tới phép tương quan chéo. Với ảnh đầu vào I và một bộ lọc K kích thước $k1 \times k2$ toán tử tương quan chéo được định nghĩa như sau:

$$(I \otimes J)_{ij} = \sum_{m=0}^{k1-1} \sum_{n=0}^{k2-1} I(i+m, j+n) K(m,n)$$

Ví dụ:



Các phần được tô màu là phần tử đầu tiên của đầu ra cùng với các phần tử của mảng đầu vào và mảng bộ lọc được sử dụng trong phép toán:

$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$$

Mục tiêu thật sự của các tầng tích chập là để trích xuất ra các đặc tính cục bộ của hình ảnh (ví dụ như các cạnh, góc). Để có thể hiểu rõ hơn ta có thể sử dụng một bộ lọc Sobel dọc (vertical Sobel filter) như sau:

$$K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Sau khi áp dụng bộ lọc trên cho một hình ảnh ta có kết quả sau:

Các đặc tính cục bộ của hình ảnh sẽ được trích xuất đầy đủ hơn bởi nhiều bộ lọc khác nhau, vì rất có thể một bộ lọc sẽ trích xuất được một vài đặc trưng (hữu ích) của vật thể và một bộ lọc khác sẽ lấy thêm được vài đặc trưng nữa. Tuy nhiên, việc xác định một bộ lọc chính xác là gì dường như là điều bất khả thi. Tuy nhiên, như mọi mô hình Neural Network khác, các giá trị của bộ lọc có thể được học bằng những tập dữ liệu huấn luyện qua quá trình huấn luyện mô hình.

4.3.2. Tóm tắt

- Chúng ta có thể thiết kế bộ lọc để phát hiện các biên (cạnh) trong ảnh.
- Chúng ta có thể học các tham số của bộ lọc từ dữ liệu.

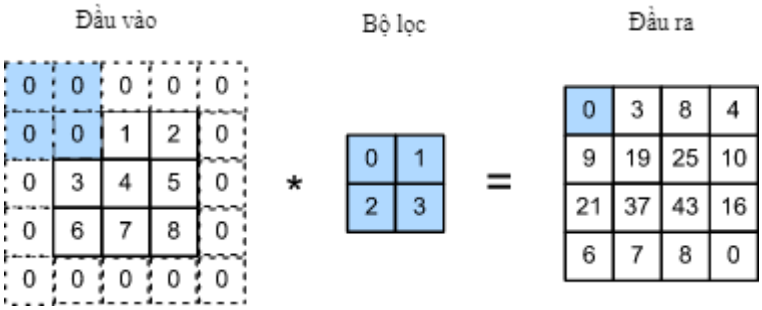
4.4. Đệm (Padding) và Sải bước (Stride)

4.4.1. Đệm (Padding)

Như đã biết, với kích thước đầu vào là nh × nw và kích thước bộ lọc là kh × kw thì kích thước của đầu ra sẽ là:

$$(nh - kh + 1) \times (nw - kw + 1)$$

Một điều dễ nhận thấy được rằng, phép tích chập có thể bỏ qua vài đặc trưng có thể rút trích từ những điểm ảnh nằm ở biên. Nếu bỏ qua và không giải quyết vấn đề này thì các điểm dữ liệu mất mát có thể tích lũy dần qua các tầng tích chập liên tiếp nhau. Ta có thể giải quyết các vấn đề này một cách đơn giản bằng việc chèn các điểm ảnh có giá trị là 0 vào các biên của ảnh, nhờ đó làm tăng kích thước sử dụng được của ảnh.



Phần tô đậm là các phần tử của mảng đầu vào và bộ lọc được sử dụng để tính phần tử đầu ra thứ nhất $0 \times 0 + 0 \times 1 + 0 \times 2 + 0 \times 3 = 0$

Nhìn chung nếu chúng ta chèn thêm tổng cộng ph hàng đệm (phân nửa ở phía trên và phân nửa ở phía dưới) và pw cột đệm (phân nửa bên trái và phân nửa bên phải) thì kích thước đầu ra sẽ là:

$$(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1)$$

Điều này có nghĩa là chiều cao và chiều rộng của đầu ra sẽ tăng thêm lần lượt là p_h và p_w. Trong nhiều trường hợp, ta sẽ muốn thiết lập $p_h = k_h - 1$ và $p_w = k_w - 1$ để đầu vào và đầu ra có cùng chiều dài và chiều rộng. Điều này sẽ giúp việc dự đoán kích thước đầu ra của mỗi tầng dễ dàng hơn khi ta xây dựng mạng.

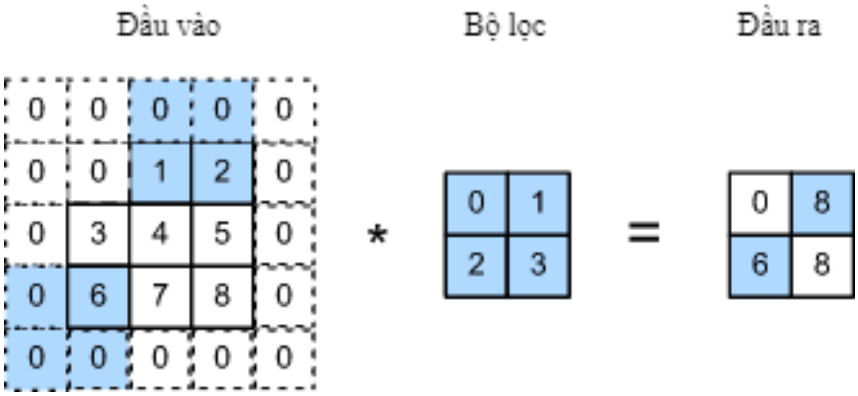
CNNs thường dùng các bộ lọc có chiều dài và chiều rộng là 1 số lẻ như 1, 3, 5, 7. Việc chọn bộ lọc có kích thước lẻ giúp chúng ta bảo toàn được các chiều không gian khi thêm cùng số hàng đệm cho cạnh trên và dưới, và thêm cùng số cột đệm cho cạnh trái phải.

4.4.2. Sải bước (Stride)

Phép tích chập sải bước (Stride Convolution) là một kỹ thuật cho phép ta giảm đáng kể kích thước ảnh khi đầu vào là một ảnh có độ phân giải quá lớn.

Ở phép 6 tích chập, thông thường ta sẽ bắt đầu di chuyển một bộ lọc từ góc trên bên trái từ vị trí điểm ảnh đầu tiên và đi tuần tự qua trái rồi xuống dưới với mỗi bước lặp cách nhau một

cột hoặc một hàng, lúc này sải bước là 1. Tuy nhiên như đã nói, khi ta muốn tăng tốc độ làm việc cũng như giảm kích thước ảnh, ta có thể bỏ qua một vài bước tích chập. Ta gọi số hàng và số cột di chuyển qua mỗi lần là sải bước (stride).



Phần tô đậm là các phần tử đầu ra, các phần tử đầu vào và bộ lọc được sử dụng để tính các đầu ra này: 0×0+0×1+1×2+2×3=8, 0×0+6×1+0×2+0×3=6.

Nhìn chung, khi sải bước theo chiều cao là s_h và sải bước theo chiều rộng là s_w , kích thước đầu ra là:

$$[(n_h - k_h + 1 + p_h + s_h)/s_h] \times [(n_w - k_w + 1 + p_w + s_w)/s_w]$$

- Phần đệm có thể tăng chiều cao vào chiều rộng của đầu ra. Nó thường được sử dụng để đầu ra có cùng kích thước với đầu vào.
- Sải bước có thể giảm độ phân giải của đầu ra, ví dụ giảm chiều cao và chiều rộng của đầu ra xuống 1/n chiều cao và chiều rộng của đầu vào (n là một số nguyên lớn hơn 1).
- Đệm và sải bước có thể được dùng để điều chỉnh kích thước chiều của dữ liệu một cách hiệu quả..

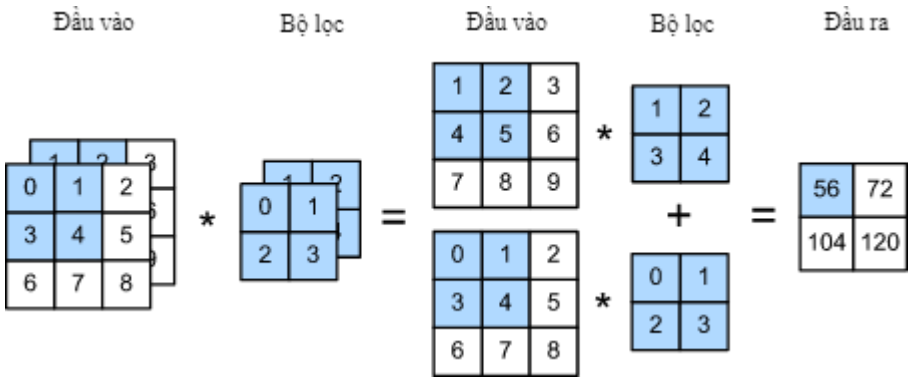
4.5. Đa kênh đầu vào và đa kênh đầu ra

4.5.1. Đa kênh đầu vào

Phép tích chập còn thể hiện tính tương quan chéo về kênh. Do đó khi ta có một Input có c_i kênh thì cũng cần một bộ lọc gồm c_i kênh tương ứng. Ví dụ ta có Input đơn giản là ảnh xám $i_h \times i_w$ có duy nhất một kênh màu thì bộ lọc chỉ đơn giản là một ma trận 2 chiều có độ lớn $k_w \times k_h$. Nhưng nếu Input là hình màu với 3 kênh màu RGB thì mỗi đầu vào sẽ có kích thước $3 \times i_w \times i_h$và bộ lọc tương ứng sẽ là $3 \times k_w \times k_h$. Vì đầu vào và bộ lọc đều có c_i kênh, ta có thể thực hiện phép tương quan chéo trên từng cặp mảng hai chiều của đầu vào và 7 bộ lọc cho mỗi kênh, rồi cộng kết quả của c_i kênh lại để tạo ra một mảng hai chiều. Đây là kết quả của phép tương quan chéo hai chiều giữa dữ liệu đầu vào đa kênh và kênh bộ lọc tích chập đa đầu vào. Với I là ảnh đầu vào, K là một tập các bộ lọc và số kênh đầu $C = 3$ ta có thể viết dưới dạng sau:

$$(I \otimes J)_{ij} = \sum_{m=0}^{k_h-1} \sum_{n=0}^{k_w-1} \sum_{c=0}^{C} I(i+m, j+n, c)K(m, n, c) + b$$

Dưới đây là hình minh hoạ cho phép tương quan chéo hai chiều với hai kênh đầu vào. Phần tô đậm là phần tử đầu ra đầu tiên cùng các phần tử của mảng đầu vào và bộ lọc được sử dụng trong phép tính đó:



Phần tô đậm là phần tử đầu ra đầu tiên cùng các phần tử của mảng đầu vào và bộ lọc được sử dụng trong phép tính đó:

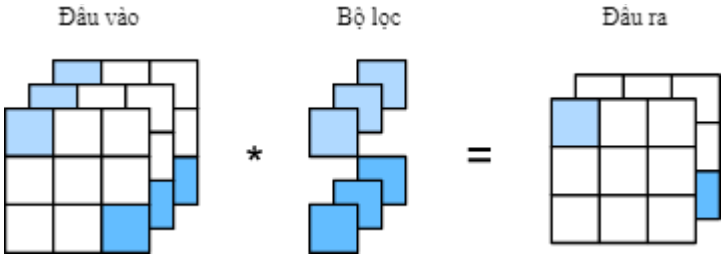
$$(1 \times 1 + 2 \times 2 + 4 \times 3 + 5 \times 4) + (0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3) = 56$$

4.5.2. Đa kênh đầu ra

Như đã biết từ trước, việc có nhiều kênh ở mỗi tầng thật sự quan trọng trong việc phát hiện cạnh của vật thể vì có thể coi mỗi kênh sẽ biểu diễn một tập đặc trưng riêng và có thể kết hợp với nhau để có thể tính toán một cách tối ưu. Xu hướng trong việc học trong mô hình mạng nơ-ron thường là tăng số lượng kênh và giảm độ phân giải. Vì vậy, việc phát hiện biên sẽ tối ưu hơn khi thực hiện trên đa kênh thay vì trên một kênh duy nhất. Đặt c_i và c_o lần lượt là số lượng kênh đầu vào và đầu ra, k_h và k_w lần lượt là chiều cao và chiều rộng của bộ lọc. Để có được một đầu ra với nhiều kênh, ta có thể tạo một mảng bộ lọc có kích thước $c_i \times k_h \times k_w$cho mỗi kênh đầu ra. Ta nối chúng lại dựa trên chiều kênh đầu ra đã biết, sao cho kích thước của bộ lọc tích chập là $c_o \times c_i \times k_h \times k_w$. Trong các phép tính tương quan chéo, kết quả trên mỗi kênh đầu ra được tính từ bộ lọc tích chập tương ứng với kênh đầu ra đó và lấy đầu vào từ tất cả các kênh trong mảng đầu vào.

4.5.3. Tầng tích chập 1x1

Suy cho cùng, phép tích chập dùng để trích xuất sự tương quan của những điểm ảnh lân cận nên các bộ lọc phải bao gồm những mảng 2 chiều và có kích thước lớn hơn 1 phần tử. Như vậy, phép tích chập 1 x 1 dường như không có ý nghĩa trong việc trích xuất những đặc trưng đó. Tuy nhiên, chúng lại là một phép toán phổ biến đôi khi được sử dụng để thiết kế các mạng sâu phức tạp. Thực chất phép tích chập 1 x 1 chỉ có tác động vào chiều kênh của đầu vào. Lưu ý rằng đầu vào và đầu ra có cùng chiều cao và chiều rộng. Mỗi phần tử trong đầu ra là một tổ hợp tuyến tính của các phần tử ở cùng một vị trí trong ảnh đầu vào. Bạn có thể xem tầng tích chập 1 × 1 như một tầng kết nối đầy đủ được áp dụng lên mỗi vị trí điểm ảnh đơn lẻ để chuyển đổi c_i giá trị đầu vào thành c_o giá trị đầu ra tương ứng. Bởi vì đây vẫn là một tầng tích chập nên các trọng số sẽ được chia sẻ giữa các vị trí điểm ảnh. Do đó, tầng tích chập 1 × 1 cần tới $c_o \times c_i$ trọng số (cộng thêm các hệ số điều chỉnh). Hình bên dưới minh họa phép tương quan chéo sử dụng bộ lọc tích chập 1 × 1



Phép tính tương quan chéo sử dụng bộ lọc tích chập 1 × 1 với 3 kênh đầu vào và 2 kênh đầu ra. Các đầu vào và các đầu ra có cùng chiều cao và chiều rộng.

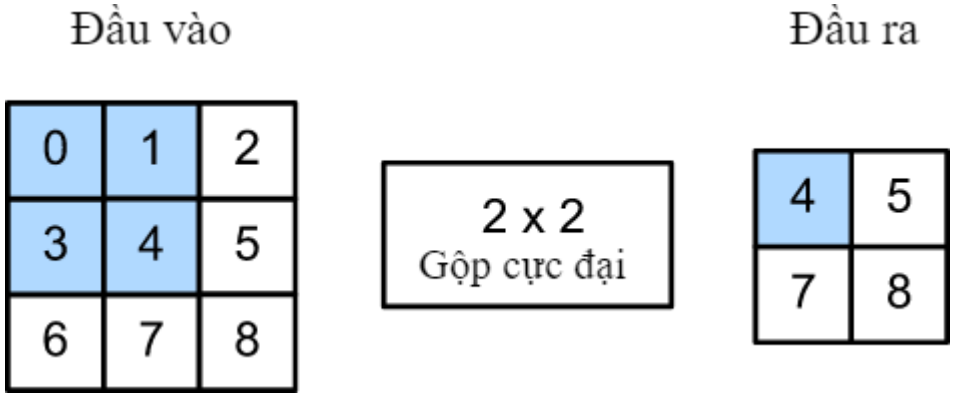
4.5.4. Tóm tắt

- Ta có thể sử dụng nhiều kênh để mở rộng các các giá trị của dữ liệu đầu vào trong phạm vi cửa sổ đó và cho ra một kết quả duy nhất. Ví dụ như phép gộp cực đại sẽ cho ra kết quả là giá trị lớn nhất của dữ liệu đầu vào trong phạm vi mà cửa sổ gộp đi qua.
- Tầng tích chập 1 × 1 khi được áp dụng lên từng điểm ảnh tương đương với tầng kết nối đầy đủ giữa các kênh.
- Tầng tích chập 1 × 1 thường được sử dụng để điều chỉnh số lượng kênh giữa các tầng của mạng và để kiểm soát độ phức tạp của mô hình.

4.6. Tầng gộp (Pooling)

4.6.1. Gộp (pooling)

Có thể nhận thấy rằng những điểm ảnh gần nhau sẽ có các giá trị tương đương nhau nên khi áp dụng bộ lọc thì kết quả của phép tích chập cũng có tính chất tương tự, các giá trị gần nhau cũng sẽ gần bằng nhau. Điều đó có thể dẫn đến việc dư thừa thông tin khi không phải mọi giá trị đều mang đến lợi ích trong việc học của mô hình, thậm chí còn làm cho mô hình phải sử dụng nhiều tài nguyên hơn. Nếu ta chỉ quan tâm đến các điểm dữ liệu thưa thớt thì vẫn có thể học được một biểu diễn toàn cục mà vẫn giữ nguyên toàn bộ lợi thế đến từ các tầng tích chập xử lý trung gian. Vấn đề trên sẽ được giải quyết tại tầng gộp bằng cách gộp các phần tử lân cận lại với nhau. Có rất nhiều cách để gộp (pooling), nhưng phổ biến nhất vẫn là gộp cực đại (**max pooling**) và ít dùng hơn là gộp trung bình (**average pooling**).



Gộp cực đại với cửa sổ có kích thước 2 × 2. Các phần tô đậm thể hiện phần tử đầu ra đầu tiên và phần tử đầu vào được dùng để tính toán: $max(0, 1, 3, 4) = 4$

Ta có thể thấy phép gộp thực chất là làm giảm kích thước đầu vào bằng cách chia chiều dài và chiều rộng cho chiều dài và chiều rộng của cửa sổ duyệt. Tương tự, khi gộp trung bình ta thay các giá trị đầu ra bằng trung bình của các giá trị ở vị trí đang duyệt. Để lý giải cho lý do mà người ta thường chọn phép gộp cực đại đó là các điểm ảnh có giá trị lớn nhất thường phản ánh rõ ràng nhất các đặc trưng cục bộ của một bức ảnh. Một điểm cuối cùng là đối với phép gộp ta không có bất kỳ tham số mô hình nào vì các phép toán đã được ta định sẵn.

4.6.2. Gộp với đầu vào vào đa kênh

Khác với tầng tích chập, tầng gộp sẽ áp dụng phép gộp lên từng kênh riêng biệt thay vì cộng từng phần tử tương ứng của các kênh lại với nhau như tầng tích chập. Do đó, số lượng kênh đầu ra cũng sẽ giống với số lượng kênh đầu vào.

4.6.3. Tóm tắt

- Tầng gộp không có tham số mô hình.
- Tầng gộp để giảm kích thước của ảnh và giảm các thông tin dư thừa và gây nhiễu.
- Số lượng kênh đầu ra của tầng gộp sẽ bằng số lượng kênh đầu vào tầng gộp đó.

4.7. Softmax

Về bản chất, CNNs là một thuật toán phân lớp. Điều này có nghĩa là khi ta có một bức ảnh là Input, ta mong nhận lại được kết quả rằng nó sẽ được phân vào lớp nào trong những lớp đã được định nghĩa sẵn trong mô hình. Ví dụ như bài toán phân lớp chó, mèo và gà, ta có sẵn một bức hình và cần biết được nó sẽ được phân vào lớp nào trong 3 lớp trên. Ta cần thêm một tầng liên kết đầy đủ và tính ra xác suất của ảnh sẽ rơi vào mỗi lớp là bao nhiêu, từ đó kết luận được nó sẽ thuộc lớp nào bằng cách lấy lớp có xác suất cao nhất. Và hàm kích hoạt ở tầng liên kết đầy đủ được lựa chọn là hàm Softmax.

4.7.1. Tổng quan

Hàm Softmax biến các giá trị thực tùy ý thành các xác suất. Với n số x1, x2 . . . xn:

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_i}}$$

Hàm Softmax rất hữu dụng trong việc phân loại hình ảnh. Giả dụ mạng nơ-ron của ta có đầu ra bao gồm 2 số thực, một số biểu diễn cho hình ảnh này là chó và số còn lại là cho mèo: $[-1,2]$

Animal	x	e^x	Probability
Dog	-1	0.368	0.047
Cat	2	7.39	0.953

Điều này có nghĩa là có đến 95.3% bức hình này là hình mèo. Với hàm Softmax ta có giải quyết bài toán phân lớp bằng xác suất thay vì chỉ là câu trả lời True/False (thực chất hàm Softmax là một trường hợp tổng quát của bài toán phân lớp nhị phân).

4.7.2. Hàm mất mát entropy chéo (Cross-entropy Loss)

Mục đích thật sự của chúng ta khi sử dụng hàm Softmax để làm hàm kích hoạt ở tầng cuối không chỉ đơn giản là để có thể phân được lớp có khả năng cao nhất. Bởi vì ta có thể xét các giá trị sau khi tính toán và lựa chọn lớp nào có giá trị cao nhất. Lý do thật sự là để ta có thể đánh giá được mức độ chính xác của thuật toán và tính được mức độ chắc chắn của dự đoán bằng cross-entropy loss. Hàm cross-entropy loss được định nghĩa như sau:

$$L = -ln(p_c)$$

Trong đó, c là lớp thực tế và pc là xác suất dự đoán cho lớp c. Như các hàm mất mất khác mất mát càng nhỏ là càng tốt. Ví dụ:

$$p_c = 0.8, L = -ln(0.8) = 0.223$$

Ta có thể thấy rằng xác suất càng cao thì độ sai lệch so với thực tế càng nhỏ.

4.8. Tổng kết

Trong chương này ta đã tìm hiểu sơ lược về mạng CNNs và các thành phần cũng như các phép toán hình thành nên nó. Đây là những cơ sở lý thuyết để ta tiến sâu hơn vào nghiên cứu và ứng dụng học sâu trong bài toán xử lý và nhận diện ung thư da qua tập hình ảnh.

5. Transfer learning và fine tuning

5.1. Định nghĩa

Transfer Learning là một kỹ thuật mà trong đó một mô hình đã được huấn luyện dựa trên một tác vụ cụ thể, khi đó một phần hay toàn bộ mô hình có thể được tái sử dụng dựa trên nhiệm vụ của mỗi layer trong mô hình đó. Một mô hình mới sử dụng một phần hay toàn bộ mô hình đã được huấn luyện (pretrained model) để học một target task cụ thể tuỳ vào mỗi nhiệm vụ của layer mà model mới có thể thêm vào.

5.2. Fine tuning

Là một phương pháp tinh chỉnh mô hình để nâng cao độ chính xác của mô hình được huấn luyện, các bước tiến hành ở đây như sau:

- Lấy mô hình đã được huấn luyện trước
- Ta sẽ tiến hành đóng băng (freezing) các lớp của mô hình đã được huấn luyện.
- Sau đó thêm vào một hoặc nhiều layer vào cuối của mô hình trên
- Tiến hành xây dựng mô hình và train với dữ liệu qua một số ít vòng epochs
- Tiến hành gỡ đóng băng các lớp trước của mô hình đã được huấn luyện
- Xây dựng mô hình và tiến hành huấn luyện lần cuối

6. Bài toán xác định ung thư da qua tập hình ảnh

6.1. Các thư viện sử dụng chính

6.1.1. Open CV

OpenCV viết tắt cho Open Source Computer Vision Library. OpenCV là thư viện nguồn mở hàng đầu cho thị giác máy tình và hiện có thêm tính năng tăng tốc GPU cho các hoạt động theo real-time.

Ở đây opencv được sử dụng để đọc và xử lý hình ảnh.

6.1.2. Tensorflow

Tensorflow là một thư viện mã nguồn mở cho máy học nổi tiếng, được phát triển bởi các nhà nghiên cứu từ Google. Việc hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning đã giúp việc tiếp cận các bài toán trở nên đơn giản, nhanh chóng và tiện lợi hơn nhiều.

Các hàm được dựng sẵn trong thư viện cho từng bài toán cho phép TensorFlow xây dựng được nhiều neural network. Nó còn cho phép bạn tính toán song song trên nhiều máy tính khác nhau, thậm chí trên nhiều CPU, GPU trong cùng 1 máy hay tạo ra các dataflow graph – đồ thị luồng dữ liệu để dựng nên các model. Nếu bạn muốn chọn con đường sự nghiệp trong lĩnh vực A.I. này, nắm rõ những điều cơ bản của TensorFlow thực sự rất quan trọng.

Trong đồ án, tensorflow là thư viện chính để sử dụng các phương pháp học sâu và huấn luyện mô hình, dự đoán dữ liệu.

6.2. Mô hình sử dụng

Việc sử dụng các mô hình khác nhau sẽ cho ra độ chính xác khác nhau rõ rệt. Tuy nhiên đối với bài toán này, mô hình Resnet50 kết hợp với các bước tiền xử lý dữ liệu sẽ được đưa vào ứng dụng. Sau đó, mô hình này sẽ được phát triển hơn trong tương lai, sử dụng các mạng khác mang lại hiệu quả và hiệu suất cao hơn như Efficient Net.

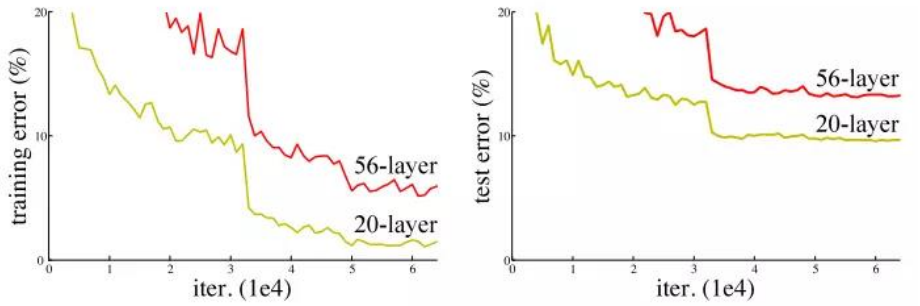
6.3. Mạng ResNet50

6.3.1. Định nghĩa

Mạng ResNet (R) là một mạng CNN được thiết kế để làm việc với hàng trăm hoặc hàng nghìn lớp chập. Một vấn đề xảy ra khi xây dựng mạng CNN với nhiều lớp chập sẽ xảy ra hiện tượng *Vanishing Gradient* dẫn tới quá trình học tập không tốt.

6.3.2. Vanishing Gradient

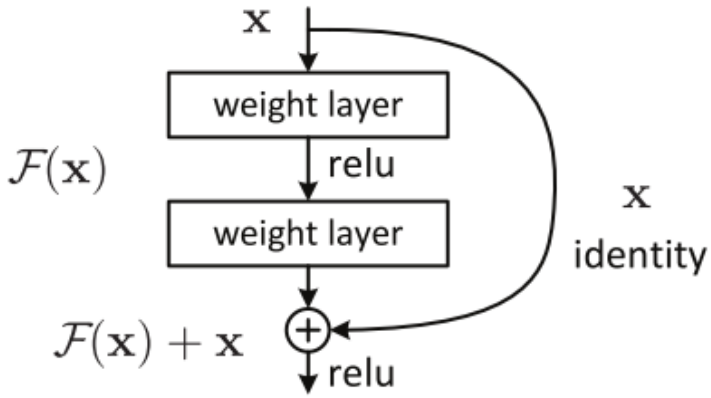
Backpropagation Algorithm là một kỹ thuật thường được sử dụng trong quá trình tranining. Ý tưởng chung của thuật toán lá sẽ đi từ output layer đến input layer và tính toán gradient của cost function tương ứng cho từng parameter (weight) của mạng. Gradient Descent sau đó được sử dụng để cập nhật các parameter đó.



Tuy nhiên, trong thực tế Gradients thường sẽ có giá trị nhỏ dần khi đi xuống các layer thấp hơn. Dẫn đến kết quả là các cập nhật thực hiện bởi Gradients Descent không làm thay đổi nhiều weights của các layer đó và làm chúng không thể hội tụ và mạng sẽ không thu được kết quả tốt. Vì thế mạng resnet ra đời để giải quyết vấn đề này.

6.3.3. Resnet

ResNet là sử dụng kết nối "tắt" đồng nhất để xuyên qua một hay nhiều lớp. Một khối như vậy được gọi là một Residual Block, như trong hình sau:



ResNet gần như tương tự với các mạng gồm có convolution, pooling, activation và fully-connected layer. Ảnh bên trên hiển thị khối dư được sử dụng trong mạng. Xuất hiện một mũi tên cong xuất phát từ đầu và kết thúc tại cuối khối dư. Hay nói cách khác là sẽ bổ sung Input X vào đầu ra của layer.

6.3.4. Resnet50

ResNet50 là một mạng neural được thiết kế dựa trên ý tưởng của ResNet, mạng này có tổng cộng 50 layers được bố trí như sau:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Sử dụng mạng resnet50 đã được huấn luyện bằng ImageNet bởi tensorflow, ta sẽ tinh chỉnh nó để đạt được kết quả phù hợp. (Aston Zhang, Zack C. Lipton, Mu Li, Alex J. Smola, 2019)

6.4. Các bước thực hiện

6.4.1. Tải xuống tập dữ liệu huấn luyện và kiểm thử

Đối với dữ liệu mẫu cho ung thư da, chúng ta sẽ sử dụng dữ liệu từ cuộc thi ISIC 2020.

6.4.2. Phân tích tập dữ liệu

Dataset sẽ được chia vào trong 2 thư mục là train (dữ liệu huấn luyện) và ISIC_2020_Test_Input (dữ liệu thử nghiệm). Bên cạnh đó, một tập training_groudTruth.csv đóng vai trò là bảng cung cấp thông tin phụ cho từng bức ảnh (metadata)

Tập dữ liệu huấn luyện có 33,126 hình ảnh và tập dữ liệu kiểm nghiệm có 10,982 tấm ảnh.

File metadata của dataset có 33,126 dòng, và nhiều cột khác nhau. Tuy nhiên ở giới hạn của đồ án này, ta chỉ quan tâm đến 2 trường là image_name (tên hình ảnh) và target (thể hiện khối u lành tính hay ác tính).

	image_name	patient_id	sex	age_approx	anatom_site_general_challenge	diagnosis	benign_malignant	target
0	ISIC_2637011	IP_7279968	male	45.0	head/neck	unknown	benign	0
1	ISIC_0015719	IP_3075186	female	45.0	upper extremity	unknown	benign	0
2	ISIC_0052212	IP_2842074	female	50.0	lower extremity	nevus	benign	0
3	ISIC_0068279	IP_6890425	female	45.0	head/neck	unknown	benign	0
4	ISIC_0074268	IP_8723313	female	55.0	upper extremity	unknown	benign	0

```
[10] training_meta_data=training_meta_data[['image_name','target']]
     training_meta_data['target']=training_meta_data['target']
     training_meta_data
```

	image_name	target
0	ISIC_2637011	0
1	ISIC_0015719	0
2	ISIC_0052212	0
3	ISIC_0068279	0
4	ISIC_0074268	0
...	...	...
33121	ISIC_9999134	0
33122	ISIC_9999320	0
33123	ISIC_9999515	0
33124	ISIC_9999666	0
33125	ISIC_9999806	0

33126 rows × 2 columns

Tuy nhiên ta nhận thấy được rằng, tập dữ liệu được cung cấp đã bị mất cân bằng khi có đến 32542 tấm ảnh biểu thị cho lành tính (giá trị bằng 0) và 584 tấm ảnh biểu thị cho ác tính (giá trị bằng 1)

In this session, we only need to focus on 2 columns:

1. image_name
2. target (float) which depicts the type of cancer:
    - Benign (0)
    - Malignant (1)

```
targets=training_meta_data['target']
targets.value_counts()
```

```
0    32542
1      584
Name: target, dtype: int64
```

As we can see, the datasets is unbalanced heavily as the number of maligant result is much smaller than the number of benign result. Howev
this can be handled by using **under sampling**. **Notice that the maximum value of Malignant is 584**

Trong thực tế, khi thu thập một lượng lớn dữ liệu, ta hay gặp các trường hợp dễ xảy ra sự mất cân bằng dữ liệu. Ví dụ như thu thập số người đang bị bệnh, đối với địa điểm thu thấp là bệnh viện, có đến 70% người đang có mặt tại bệnh viện là bị bệnh, 30% là con số người không bị nhiễm bệnh. Tuy nhiên trong thực tế nếu ta thu thập dữ liệu ở trường học, công sở… thì chỉ có khoảng 1-5% cho số người đang bị nhiễm bệnh và còn lại là không nhiễm bệnh.

Đây chính là một ví dụ cho tính mất cân bằng dữ liệu thường xuyên gặp phải trong thực tế với các mẫu dataset.

Vì thế để tránh việc mất cân bằng dữ liệu , tạm thời ta sẽ sử dụng phương pháp undersampling

Resolving unbalance data

```
# Shuffle the Dataset.
shuffled_data = training_meta_data.sample(frac=1,random_state=4)

# Put all the fraud class in a separate dataset.
fraud_data = shuffled_data.loc[shuffled_data['target'] == 1]

#Randomly select 600 observations from the non-fraud (majority class)
non_fraud_data = shuffled_data.loc[shuffled_data['target'] == 0].sample(n=600,random_state=42)

# Concatenate both dataframes again
normalized_data = pd.concat([fraud_data, non_fraud_data])

training_meta_data=normalized_data
```

```
[13] unique, counts = np.unique(training_meta_data['target'], return_counts=True)

     print('The number of begnign (0) and maligant(1) images')
     dict(zip(unique, counts))
```

```
The number of begnign (0) and maligant(1) images
{0: 600, 1: 584}
```

Now the unbalanced data problem is resolved

6.4.3. Tiền xử lý dữ liệu

Bên cạnh việc giải quyết tính mất cân bằng dữ liệu,

Ta cần phải tiến hành gán nhãn cho dữ liệu, chỉnh sửa lại kích thước ảnh để tối ưu thời gian và phân chia tập dữ liệu

```
[9] from tensorflow.python.keras.engine import training
    def load_images_from_folder(folder):
        images = []
        targets = training_meta_data['target']
        filenames = training_meta_data['image_name']

        for idx,filename in filenames.items():
          filename = filename+'.jpg'
          path = os.path.join(folder,filename)
          img = cv2.imread(path)
          img = cv2.resize(img,dsize=(128,128))

          if img is not None:
                images.append(img)

          print('Importing: ',filename, f'current imported: {len(images)}')

        return images,targets
```

```
images,targets=load_images_from_folder('/content/train')
```

6.4.4. Tiến hành huấn luyện mô hình

1. *Sử dụng mô hình đã được huấn luyện sẵn*

Ở đây ta sẽ sử dụng mạng ResNet50 đã được huấn luyện sẵn dựa trên ImageNet sau đó đóng băng các lớp của mô hình.

```
[20] def build_res_model(lr):
        base_model = ResNet50(weights='imagenet', include_top=False,input_tensor=Input(shape=(128,128,3

        # freezing layers
        for layer in base_model.layers:
            layer.trainable = False
```

2. *Thay đổi kiểu dữ liệu đầu vào và lớp cuối của mô hình cho phù hợp với bài toán.*

- Ở đây ta thêm lớp cuối là lớp Dense với hàm kích hoạt là sigmoid. Như đã nói ở trên đây là hàm kích hoạt được sử dụng trong bài toán phân lớp nhị phân.
- Đối với thuật toán tối ưu (optimizer), ta sẽ sử dụng thuật toán Stochastic Gradient Descent. Đây là thuật toán tối ưu nên được sử dụng trong bài toán này vì số lượng dữ liệu đầu vào có thể lớn hơn trong tương lai.
- Đối với hàm mất mát, binary_crossentropy sẽ được sử dụng cho bài toán phân lớp nhị phân này.

```
[20] def build_res_model(lr):
        base_model = ResNet50(weights='imagenet', include_top=False,input_tensor=Input(shape=(128,128,3

        # freezing layers
        for layer in base_model.layers:
            layer.trainable = False

        head_model = base_model.output
        head_model = AveragePooling2D(pool_size=(4, 4))(head_model)
        head_model = Flatten(name='flatten')(head_model)
        head_model = Dense(256, activation='relu')(head_model)
        head_model = Dropout(0.2)(head_model)
        head_model = Dense(1, activation='sigmoid')(head_model)

        # Compile
        model = Model(inputs=base_model.input, outputs=head_model)

        optimizer = SGD(learning_rate=lr)
        model.compile(loss=BinaryCrossentropy(), optimizer=optimizer, metrics=['accuracy'])

        return model
```

3. *Tăng cường dữ liệu (Augmentation Data)*

- Ta sẽ sử dụng Lớp ImageDataGenerator để tạo ra các hình ảnh khác nhau với những yếu tố khác nhau tránh trường hợp mô hình bị overfitting

```
[24] def training_model(model,epochs):
        model.fit_generator(
            aug.flow(
                x_train,
                y_train,
                batch_size=100,
            ),
            epochs=epochs,
            validation_data=aug.flow(
                x_test,
                y_test,
                batch_size=100
            ),
            callbacks=[],
            verbose=2
        )
```

```
[25] training_model(model,2)

     /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: UserWarning: `Model.fit_generator`
       from ipykernel import kernelapp as app
     Epoch 1/2
     10/10 - 65s - loss: 0.7435 - accuracy: 0.4995 - val_loss: 0.7014 - val_accuracy: 0.4937 - 65s/epoch
     Epoch 2/2
     10/10 - 59s - loss: 0.7125 - accuracy: 0.4879 - val_loss: 0.6915 - val_accuracy: 0.4937 - 59s/epoch
```

```
[22] from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
[23] # construct the training image generator for data augmentation
     aug = ImageDataGenerator(
         rotation_range=20,
         zoom_range=0.1,
         rescale=1./255,
         width_shift_range=0.1,
         height_shift_range=0.1,
         horizontal_flip=True,
         vertical_flip=True,
         fill_mode="nearest"
     )

     aug_val = ImageDataGenerator(rescale=1./255)
```

4. *Tiến hành huấn luyện mô hình qua 2 vòng*

```
[24] def training_model(model,epochs):
       model.fit_generator(
           aug.flow(
               x_train,
               y_train,
               batch_size=100,
           ),
           epochs=epochs,
           validation_data=aug.flow(
               x_test,
               y_test,
               batch_size=100
           ),
           callbacks=[],
           verbose=2
       )
```

```
[25] training_model(model,2)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: UserWarning: `Model.fit_generator`
  from ipykernel import kernelapp as app
Epoch 1/2
10/10 - 65s - loss: 0.7435 - accuracy: 0.4995 - val_loss: 0.7014 - val_accuracy: 0.4937 - 65s/epoch
Epoch 2/2
10/10 - 59s - loss: 0.7125 - accuracy: 0.4879 - val_loss: 0.6915 - val_accuracy: 0.4937 - 59s/epoch
```

5. *Tiến hành gỡ đóng băng và huấn luyện lại mô hình*
6. *Dự đoán và đo độ chính xác của mô hình*

6.5. Kết quả bài toán

[skin-diagnosis.ipynb - Colaboratory (google.com)]

7. Kết luận và hướng phát triển

7.1. Kết quả đạt được

Trong quá trình phát triển và thực hiện đồ án 1, em đã nắm được kiến thức cơ bản về máy học (machine learning), học sâu (deep learning), mạng neural tích chập và các thuật toán xoay quanh nó như back-propagation. Một số mô hình đã được huấn luyện sẵn và cách áp dụng chúng để tự tạo mô hình riêng dựa trên tri thức có sẵn. Bên cạnh đó, em còn tìm hiểu thêm về cách thức xử lý dữ liệu mất cân bằng và phương pháp tăng cường dữ liệu ảnh (Augmented Data). Từ những kiến thức đó, em đã phối hợp và hoàn thành nên bài báo cáo này.

7.2. Hạn chế

- Việc sử dụng undersampling gây mất mát rất nhiều dữ liệu huấn luyện dẫn đến việc lãng phí
- Mô hình Resnet50 là một mô hình khá lâu đời nên chưa phải là phương pháp tối ưu ở thời điểm hiện tại.
- Bài toán chưa được mở rộng vì chỉ đang dừng lại ở việc phân lớp nhị phân.

7.3. Hướng phát triển

- Trong đồ án tiếp theo, em sẽ tiến hành thử nghiệm tập dữ liệu trên những mô hình khác nhau hiện đại và tối ưu hơn như Efficient Net, Convolutional và transformer
- Mở rộng tính ứng dụng của mô hình
- Nâng cao hơn ở việc tiền xử lý dữ liệu thay vì sử dụng undersampling để giải quyết imbalanced data
- Mở rộng bài toán từ classification thành bài toán Segmentation.

References

Aston Zhang, Zack C. Lipton, Mu Li, Alex J. Smola, 2019. *Dive Into Deep Learning.* [Trực tuyến]
Available at: https://d2l.ai/
[Đã truy cập 28 July 2021].

Tiep, V. H., 2019. *Machine Learning Co Ban.* [Trực tuyến]
Available at: https://machinelearningcoban.com/
[Đã truy cập August 2021].

Anon., 2019. *Kaggle.* [Trực tuyến]
Available at: https://www.kaggle.com/search?q=skin+cancer+in%3Anotebooks
[Đã truy cập 28 August 2021].

Google, 2016. *Tensorflow.* [Trực tuyến]
Available at: https://www.tensorflow.org/api_docs/python/tf
[Đã truy cập 30 September 2021].

Anon., 2019. *Py Image Search.* [Trực tuyến]
Available at: https://www.pyimagesearch.com/2020/04/27/fine-tuning-resnet-with-keras-tensorflow-and-deep-learning/
[Đã truy cập 14 Octorber 2021].