

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP HỒ CHÍ MINH



ĐỒ ÁN 1

Ứng dụng deep learning trong phát hiện bệnh ở phổi người

Nguyễn Thanh Toàn

MSSV: 2014777

toan.nguyenpsc@hcmut.edu.vn

Khoa Điện-Điện tử

Chuyên ngành điện tử viễn thông

Giảng viên hướng dẫn: Nguyễn Chí Ngọc

Bộ môn: Điện tử viễn thông

Khoa: Điện-Điện tử

TP HỒ CHÍ MINH, 7/2023

Mục Lục

PHẦN MỞ ĐẦU	1
1. Tính cấp thiết, ý nghĩa thực tiễn của đề tài.....	1
2. Nhiệm vụ đồ án.....	1
3. Lĩnh vực ứng dụng:.....	1
CHƯƠNG 1: MÔ HÌNH NEURAL NETWORK	2
1. Giới thiệu mô hình neural network	2
2. Loss function.....	5
3. Gradient Descent.....	5
CHƯƠNG 2: XÂY DỰNG MÔ HÌNH NEURAL NETWORK CHO VIỆC CHẨN ĐOÁN BỆNH ...	7
1. Mô hình CNN (Convolution neural network)	7
2. Chi tiết mô hình CNN sẽ được xây dựng	9
CHƯƠNG 3: THỰC NGHIỆM	12
1. Xử lý dữ liệu	12
2. Xây dựng mô hình.....	17
3. Kết quả huấn luyện model.....	18
4. Thực hành dự đoán kết quả của model.....	19
CHƯƠNG 4: KHẢ NĂNG PHÁT TRIỂN THÊM CỦA ĐỀ TÀI	30
1. GPU phù hợp hơn cho các ứng dụng liên quan đến máy học.....	30
2. Xây dựng mạng neural network trên ASIC hoặc FPGA.....	30
3. Cải tiến mô hình neural network hiện tại và liên kết nhiều mô hình neural network lại với nhau.....	31
PHẦN KẾT LUẬN	33
TÀI LIỆU THAM KHẢO	34

Danh mục hình ảnh

Hình 1: Hình ảnh mô hình hoạt động cơ bản của neural network	2
Hình 2: Ảnh neural network cơ bản dạng rút gọn	2
Hình 3: Mô hình tổng quang neural network.....	2
Hình 4: Mô hình neural network chi tiết.....	4
Hình 5: Loss function với $y = 1$ và $y=0$	5
Hình 6: Minh họa cơ bản thuật toán gradient descent.....	5
Hình 7: Mô hình CNN (convolution neural network).....	7
Hình 8: minh họa output khi qua lớp tích chập	8
Hình 9: Thuật toán max pooling 2x2	8
Hình 10: Trãi thẳng dữ liệu trước khi vào mạng fully connected.....	8
Hình 11: Hàm sigmoid.....	11
Hình 12: Tổ chức dữ liệu trong file csv.....	12
Hình 13: Source code python phân loại file ảnh và nén vào file zip	13
Hình 14: File ảnh source code thực tế trên subline text	14
Hình 15: Thực thi file app.py để phân loại ảnh	14
Hình 16: Kết quả trả về của source code phân loại file.....	15
Hình 17: Kết quả source code phân loại và nén file.	15
Hình 18: Các file ảnh được ném vào file dataNoF.zip	16
Hình 19: Các file được nén vào file dataInf.zip	16
Hình 20: Kết quả xây dựng model.....	17
Hình 21: Xây dựng model.....	17
Hình 22: Kết quả huấn luyện model.....	18
Hình 23: model.h5 chứa các trọng số của model sau khi huấn luyện	18
Hình 24: Source code phần thực hành chẩn đoán bệnh.....	19
Hình 25: Giao diện chính của chương trình	28
Hình 26: Giao diện chương trình khi ở chế độ kiểm tra danh sách bệnh nhân	28
Hình 27: Giao diện ở chế độ predict	29
Hình 28: Kết quả chẩn đoán chương trình trả về.....	29
Hình 29: Neural network trên FPGA hoặc ASIC.....	31
Hình 30: Hàm activation softmax.....	32

PHẦN MỞ ĐẦU

1. Tính cấp thiết, ý nghĩa thực tiễn của đề tài

Ngày nay có nhiều loại bệnh với những biểu hiện tương đối có sự chồng lấn nhau, gây khó khăn cho việc chẩn đoán bệnh nếu người chẩn đoán bệnh không có kiến thức về y khoa. Điều này đã gây ra một áp lực không nhỏ cho lực lượng y bác sĩ trong việc khám sàng lọc, phân loại bệnh nhân. Vì vậy cần thiết có một giải thuật, phần mềm AI dựa trên cơ sở dữ liệu thu thập được trong quá khứ để sàng lọc, chẩn đoán bệnh cơ bản dễ dàng theo dõi sức khỏe, cũng như chăm sóc sức khỏe cho bệnh nhân, Giúp giảm tải áp lực cho lực lượng y tế. Việc nhìn vào những bức ảnh chẩn đoán bệnh như ảnh chụp CT hoặc Xray điều gây ra rất nhiều khó khăn cho người chẩn đoán bệnh kể cả người có kiến thức chuyên môn vì những chi tiết thường khá nhỏ và khó nhận biết, Vì vậy sẽ giúp ích được rất nhiều cho y bác sĩ nên có chương trình phân tích và chẩn đoán bệnh qua ảnh chụp y tế. Cụ thể hơn là chẩn đoán bệnh “Inf” ở phổi người.

2. Nhiệm vụ đồ án

Xây dựng mô hình mạng neural thần kinh ảo cơ bản nhất cho việc chẩn đoán bệnh “inf” ở phổi người, Đầu vào là ảnh chụp Xray khoang ngực của bệnh nhân, Đầu ra sẽ là kết quả bệnh nhân ấy có bị bệnh “Inf” hay không.

3. Lĩnh vực ứng dụng:

Y tế, và các lĩnh vực có liên quan.

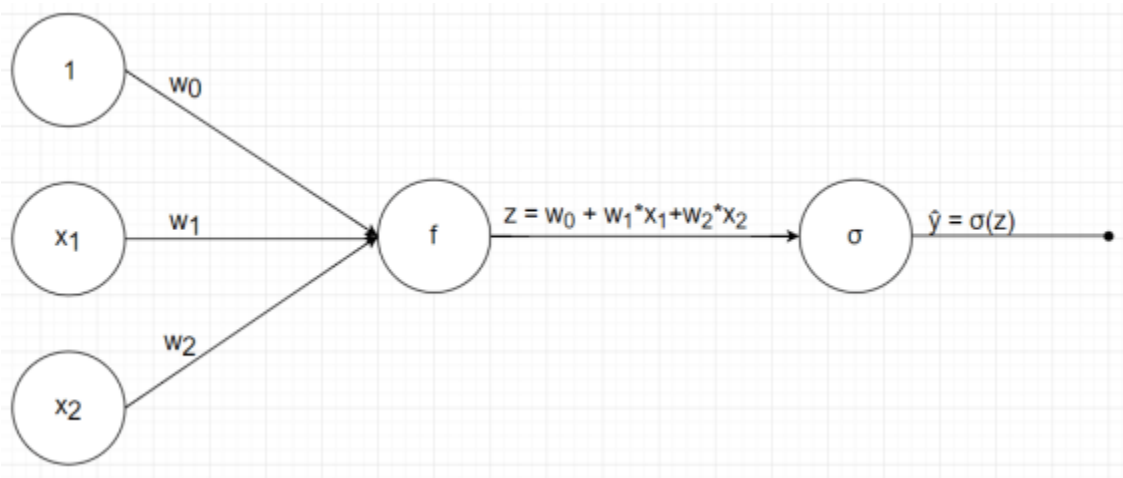
CHƯƠNG 1: MÔ HÌNH NEURAL NETWORK

1. Giới thiệu mô hình neural network

Logistic regression là mô hình neural network đơn giản nhất chỉ với input layer và output layer.

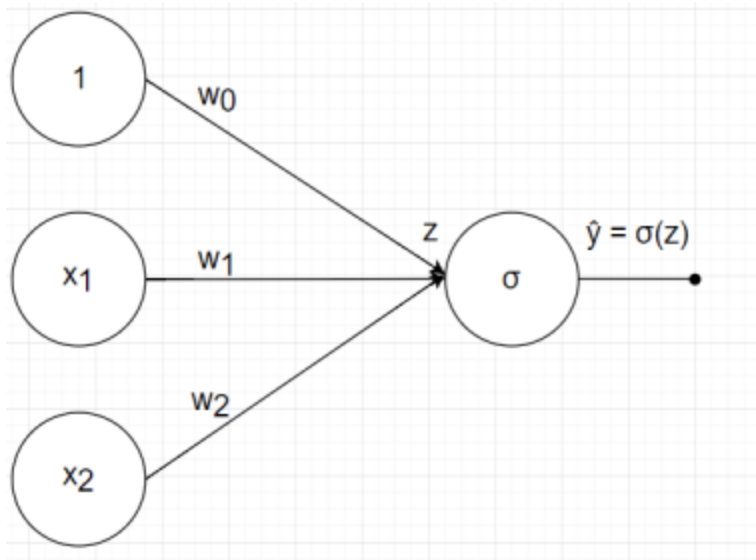
Mô hình của logistic regression $\hat{y} = \sigma(w_0 + w_1 * x_1 + w_2 * x_2)$. Có 2 bước:

- Tính tổng linear: $z = 1 * w_0 + x_1 * w_1 + x_2 * w_2$
- Áp dụng sigmoid function: $\hat{y} = \sigma(z)$



Hình 1: Hình ảnh mô hình hoạt động cơ bản của neural network

Để biểu diễn gọn lại ta sẽ gộp hai bước trên thành một trên biểu đồ

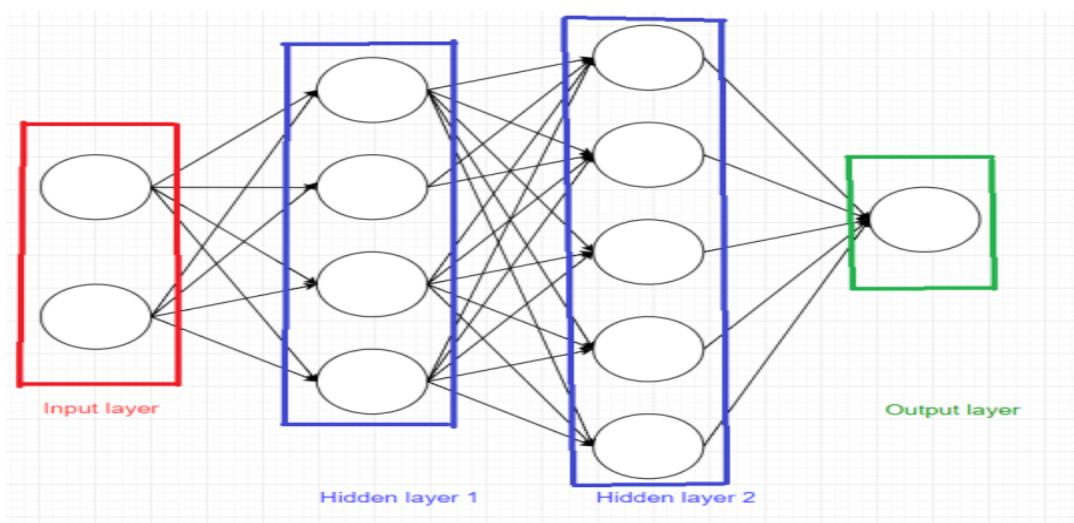


Hình 2: Ảnh neural network cơ bản dạng rút gọn

Hệ số w_0 còn được gọi là bias. Để ý từ những bài trước đến giờ dữ liệu khi tính toán luôn được thêm 1 để tính hệ số bias w_0 . phương trình đường thẳng sẽ thế nào nếu bỏ w_0 , phương trình giờ có dạng: $w_1 * x + w_2 * y = 0$, sẽ luôn đi qua gốc tọa độ và nó không tổng quát hóa phương trình đường thẳng nên có thể không tìm được phương trình mong muốn. => Nên buộc phải thêm hệ số tự do bias.

Hàm sigmoid ở đây được gọi là **activation function**.

Mô hình tổng quát:



Hình 3: Mô hình tổng quát neural network

Layer đầu tiên là input layer, các layer ở giữa được gọi là hidden layer, layer cuối cùng được gọi là output layer. Các hình tròn được gọi là node.

Mỗi mô hình luôn có 1 input layer, 1 output layer, có thể có hoặc không các hidden layer.

Tổng số layer trong mô hình được quy ước là số layer – 1 (Không tính input layer).

Ví dụ như ở hình trên có 1 input layer, 2 hidden layer và 1 output layer. Số lượng layer của mô hình là 3 layer.

Mỗi node trong hidden layer và output layer :

- Liên kết với tất cả các node ở layer trước đó với các hệ số w riêng.
- Mỗi node có 1 hệ số bias b riêng.
- Diễn ra 2 bước: tính tổng linear và áp dụng activation function.

Kí hiệu

Số node trong hidden layer thứ i là $l(i)$.

Ma trận $W(k)$ kích thước $l(k-1)*l(k)$ là ma trận hệ số giữa layer $(k-1)$ và layer k , trong đó $w_{ij}(k)$ là hệ số kết nối từ node thứ i của layer $k-1$ đến node thứ j của layer k .

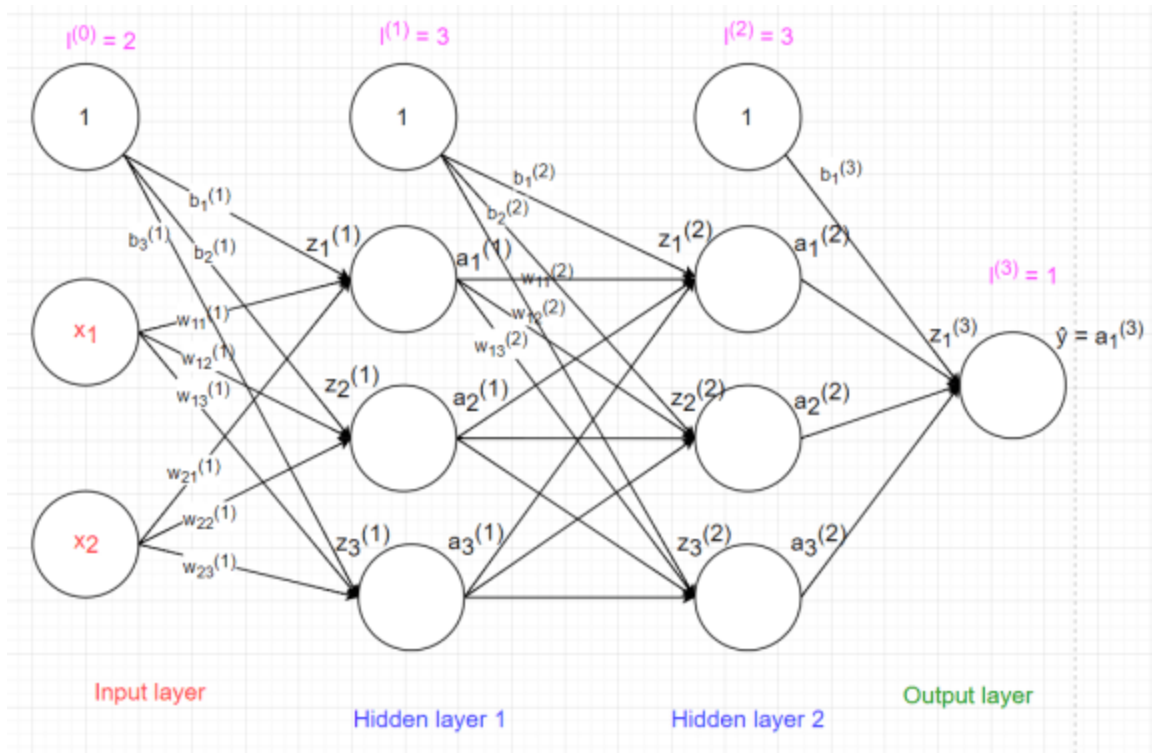
Vector $b(k)$ kích thước $l(k)*1$ là hệ số bias của các node trong layer k , trong đó $b_i(k)$ là bias của node thứ i trong layer k .

Với node thứ i trong layer l có bias $b_i(l)$ thực hiện 2 bước:

- Tính tổng linear: $z_i(l) = \sum_{j=1}^{l(l-1)} a_j(l-1) * w_{ji}(l) + b_i(l)$, là tổng tất cả các node trong layer trước nhân với hệ số w tương ứng, rồi cộng với bias b .
- Áp dụng activation function: $a_i(l) = \sigma(z_i(l))$

Vector $z(k)$ kích thước $l(k)*1$ là giá trị các node trong layer k sau bước tính tổng linear.

Vector $a(k)$ kích thước $l(k)*1$ là giá trị của các node trong layer k sau khi áp dụng hàm activation function.



Hình 4: Mô hình neural network chi tiết

Mô hình neural network trên gồm 3 layer. Input layer có 2 node $l(0)=2$, hidden layer 1 có 3 node, hidden layer 2 có 3 node và output layer có 1 node.

Do mỗi node trong hidden layer và output layer đều có bias nên trong input layer và hidden layer cần thêm node 1 để tính bias (nhưng không tính vào tổng số node layer có).

Tại node thứ 2 ở layer 1, ta có:

- $z_2(1) = x_1 * w_{12}(1) + x_2 * w_{22}(1) + b_2(1)$
- $a_2(1) = \sigma(z_2(1))$

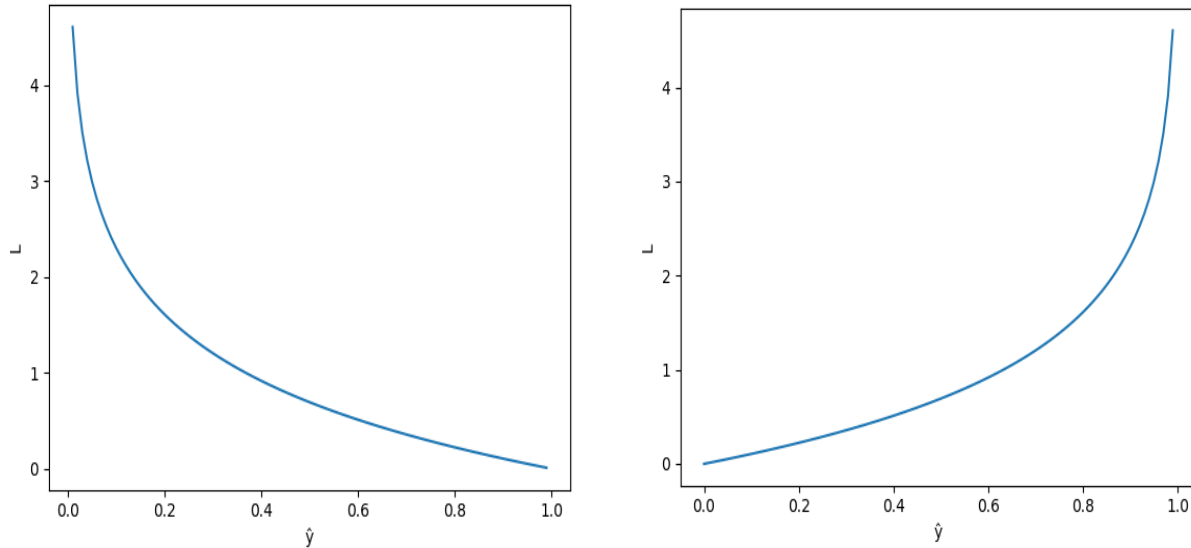
Hay ở node thứ 3 layer 2, ta có:

- $z_3(2) = a_1(1) * w_{13}(2) + a_2(1) * w_{23}(2) + a_3(1) * w_{33}(2) + b_3(2)$
- $a_3(2) = \sigma(z_3(2))$

2. Loss function

để đánh giá độ tốt của model. Ta cần tìm các hệ số sao cho y^{\wedge} càng gần y càng tốt
hàm loss function

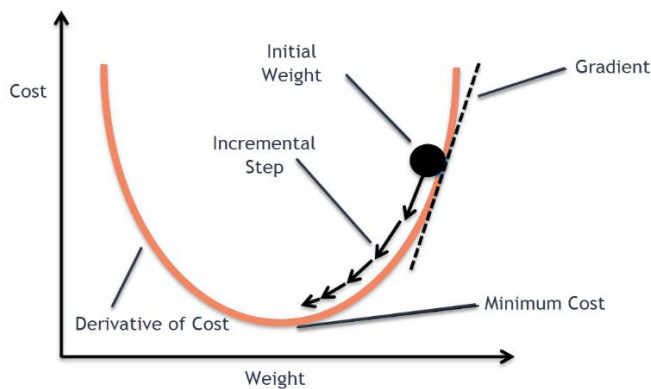
$$L = -(y_i \log(y_i^{\wedge}) + (1 - y_i) \log(1 - y_i^{\wedge}))$$



Hình 5: Loss function với $y = 1$ và $y = 0$

3. Gradient Descent

Gradient Descent là thuật toán tìm tối ưu chung cho các hàm số. Ý tưởng chung của thuật toán là điều chỉnh các tham số để lặp đi lặp lại thông qua mỗi dữ liệu huấn luyện để giảm thiểu hàm chi phí.



Hình 6: Minh họa cơ bản thuật toán gradient descent

Gradient Descent là một thuật toán tối ưu lặp (iterative optimization algorithm) được sử dụng trong các bài toán Machine Learning và Deep Learning (thường là các bài toán tối ưu lồi — Convex Optimization) với mục tiêu là tìm một tập các biến nội tại (internal parameters) cho việc tối ưu models. Trong đó:

- Gradient: là tỷ lệ độ nghiêng của đường dốc (rate of inclination or declination of a slope). Về mặt toán học, Gradient của một hàm số là đạo hàm của hàm số đó tương ứng với mỗi biến của hàm. Đối với hàm số đơn biến, chúng ta sử dụng khái niệm Derivative thay cho Gradient.
- Descent: là từ viết tắt của descending, nghĩa là giảm dần.

Gradient Descent có nhiều dạng khác nhau như Stochastic Gradient Descent (SGD), Mini-batch SDG. Nhưng về cơ bản thì đều được thực thi như sau:

- Khởi tạo biến nội tại.
- Đánh giá model dựa vào biến nội tại và hàm mất mát (Loss function).
- Cập nhật các biến nội tại theo hướng tối ưu hàm mất mát (finding optimal points).
- Lặp lại bước 2, 3 cho tới khi thỏa điều kiện dừng.
- Công thức cập nhật cho thuật toán có thể được viết là:

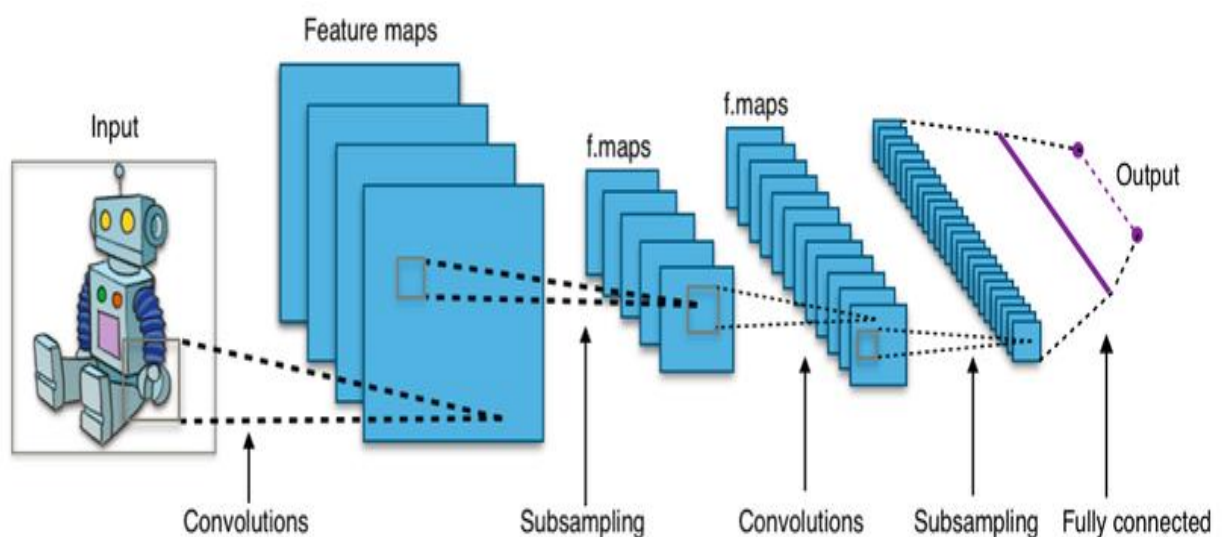
$$\theta^{(\text{nextStep})} = \theta - \eta \Delta_{\theta}$$

trong đó θ là tập các biến cần cập nhật, η là tốc độ học (learning rate), Δ_{θ} là Gradient của hàm mất mát f theo tập θ .

CHƯƠNG 2: XÂY DỰNG MÔ HÌNH NEURAL NETWORK CHO VIỆC CHẨN ĐOÁN BỆNH

1. Mô hình CNN (Convolution neural network)

Đối với mô hình Logistic regression fully connected khi có đầu vào là một dữ liệu ảnh thường sẽ có kích thước từ vài trăm đến vài nghìn pixel theo mỗi chiều và 3 chiều hệ màu RGB, Vì vậy mạng logistic regression fully connected phải nhận tất cả các pixel của dữ liệu ảnh vào lớp input điều này sẽ gây ra một áp lực rất lớn cho việc tính toán vì vậy ta cần phải thay đổi mô hình thành mô hình CNN (convolution neural network). Khác với mô hình logistic regression fully connected thì mô hình CNN nhìn vào dữ liệu ảnh theo từng vùng theo từng kenel với kích thước định sẵn, điều này sẽ giúp giảm đáng kể áp lực tính toán cho phần cứng. Giúp cho việc huấn luyện mô hình trở nên dễ dàng hơn tránh gây quá tải cho phần cứng.



Hình 7: Mô hình CNN (convolution neural network)

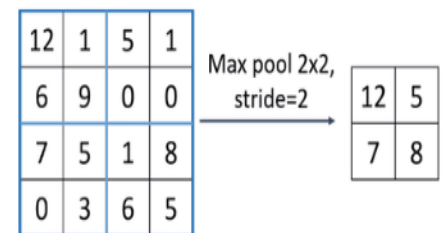
Ta thấy rằng ban đầu dữ liệu sẽ được đi qua các lớp convolution hay các lớp tích chập. Điều này rất đơn giản là lấy từng ma trận pixel của dữ liệu đầu vào và nhân lần lượt với từng tham số của ma trận kenel, thường thì việc này sẽ trích xuất ra những đặc trưng cơ bản của dữ liệu đầu vào. Các trọng số của kenel không phải là những con số cố định mà

sẽ được điều chỉnh trong quá trình huấn luyện model để đạt được hàm loss function thấp nhất sẽ được thực hiện thông qua thuật toán gradient descent.



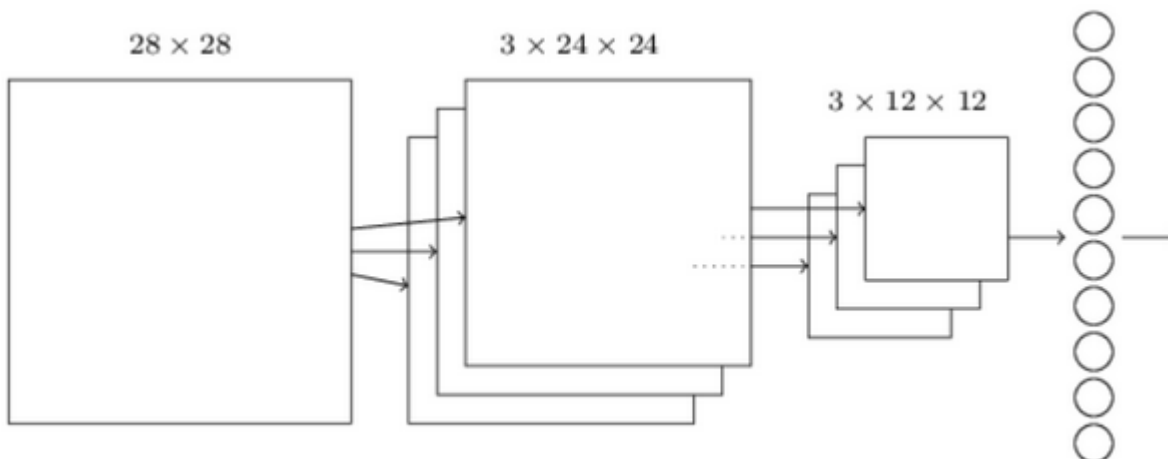
Hình 8: minh họa output khi qua lớp tích chập

Lớp Subsampling dùng để củng cố thêm các đặc trưng của dữ liệu đầu vào và làm giảm số lượng các tham số trước khi vào lớp tiếp theo. Thường sẽ có nhiều các thực hiện tuy nhiên thường dùng nhất là max pooling 2x2.



Hình 9: Thuật toán max pooling 2x2

Và sau khi đã qua các lớp tích chập sẽ tiến hành trải phẳng dữ liệu và đưa đến lớp fully connected, về bản chất giống như mô hình logistic regression fully connected mà đã tìm hiểu ở chương 1.



Hình 10: Trải phẳng dữ liệu trước khi vào mạng fully connected

Cuối cùng là lớp output là lớp sẽ đưa ra kết quả, hay giá trị dự đoán của mô hình.

2. Chi tiết mô hình CNN sẽ được xây dựng

Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Trong mô hình mạng truyền ngược (feedforward neural network) thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo.

Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Còn trong mô hình CNNs thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution.

Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó.

Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong quá trình huấn luyện mạng (training) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.

Mô hình CNN được sử dụng cho ứng dụng chẩn đoán bệnh phổi ở người sẽ có cấu trúc như sau:

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

Mô hình gồm layer đầu vào có cấu trúc của một ma trận có kích thước 512x512x3, gồm chiều dài là 512 pixel với một pixel được đại diện bởi một số không dấu có giá trị từ 0 đến 255, tương tự vậy thì ma trận input cũng có chiều rộng là 512. Và có chiều sâu là 3 tương ứng với 3 hệ màu cơ bản là R (red), B (blue), G (green).

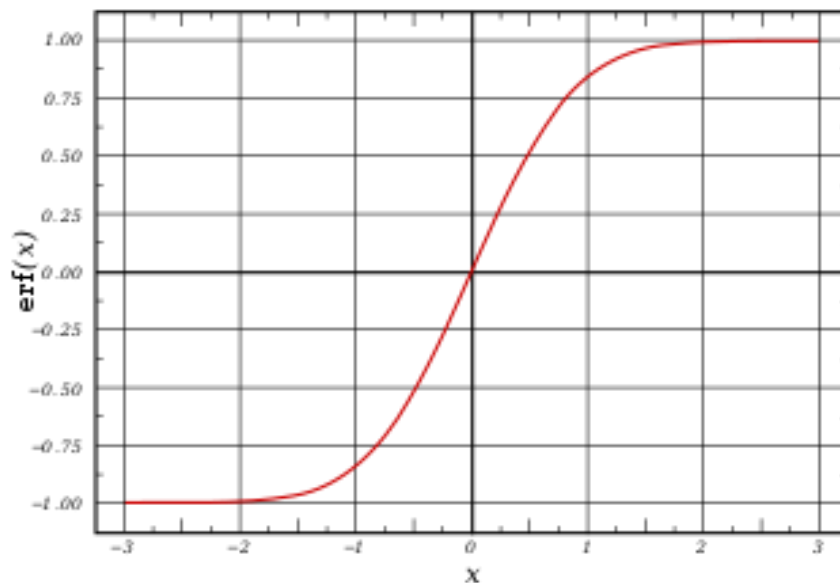
Tiếp theo dữ liệu đầu vào sẽ được đi qua một lớp tích chập 2 chiều có kích thước kernel là 3x3 (chiều dài 3, chiều rộng là 3), Đầu ra là 16 ma trận mới đều có kích thước là 512x512x3 với activation function là hàm Relu (dùng để khử tuyến tính).

Tiếp theo toàn bộ dữ liệu sẽ được tiếp tục đi đến maxpolling, hàm này sẽ lấy ra pixel có độ lớn lớn nhất trong 4 pixel bên cạnh, Sau khi qua lớp này thì dữ liệu sẽ được giảm đi gấp đôi, tuy nhiên sẽ không ảnh hưởng đến kết quả đầu ra mà còn làm nổi bật thêm các đặc trưng của dữ liệu đồng thời làm giảm đi áp lực tính toán cho phần cứng. Tương tự với các lớp tiếp theo.

`tf.keras.layers.Flatten()`, Đây là lớp để trải phẳng dữ liệu, dữ liệu tại đầu ra của ma trận sẽ được sắp xếp lại theo đường thẳng lần lượt theo từng hàng, các ma trận hệ màu cũng sẽ được gộp lại thành một và xếp theo đường thẳng.

`tf.keras.layers.Dense(512, activation='relu')`, Lớp fully connected có nghĩa là mọi đầu ra của lớp này sẽ được nối với toàn bộ các node của neural của lớp tiếp theo, Điều này tạo ra một áp lực tính toán rất lớn cho phần cứng.

`tf.keras.layers.Dense(1, activation='sigmoid')`, Lớp cuối cùng là lớp output layer, Chỉ có một node duy nhất và có activation là hàm sigmoid, Hàm này sẽ đưa kết quả đầu ra có giá trị từ 0 đến 1. Tương ứng như vậy ta sẽ thu được kết quả chẩn đoán của mô hình nếu kết quả đầu ra lớn hơn 0.5 thì ta kết luận dữ liệu không có bệnh “Inf”, và ngược lại nếu đầu ra cho kết quả nhỏ hơn 0.5 thì ta kết luận dữ liệu này bị bệnh “Inf”.



Hình 11: Hàm sigmoid

CHƯƠNG 3: THỰC NGHIỆM

1. Xử lý dữ liệu

Dữ liệu ban đầu chưa được phân ra thành từng file, các hình ảnh được chia thành từng thư mục với số lượng file trong một thư mục là rất nhiều, tuy nhiên các file ảnh không được đặt tên hay phân loại đúng theo từng label mà các label của từng file được lưu trong một file csv. Vì vậy ta cần phải phân loại từng file theo label của ảnh vào một file zip để thuận tiện cho việc upload và huấn luyện cho mô hình.

	A	B	C	D	E	F
1	Image Index	Finding Labels	Follow-up	Patient ID	Patient Age	Patient Gender
2	00000001_000.png	Cardiomegaly	0	1	58	M
3	00000001_001.png	Cardiomegaly Emphysema	1	1	58	M
4	00000001_002.png	Cardiomegaly Effusion	2	1	58	M
5	00000002_000.png	No Finding	0	2	81	M
6	00000003_000.png	Hernia	0	3	81	F
7	00000003_001.png	Hernia	1	3	74	F
8	00000003_002.png	Hernia	2	3	75	F
9	00000003_003.png	Hernia Infiltration	3	3	76	F
10	00000003_004.png	Hernia	4	3	77	F
11	00000003_005.png	Hernia	5	3	78	F
12	00000003_006.png	Hernia	6	3	79	F
13	00000003_007.png	Hernia	7	3	80	F
14	00000004_000.png	Mass Nodule	0	4	82	M
15	00000005_000.png	No Finding	0	5	69	F
16	00000005_001.png	No Finding	1	5	69	F
17	00000005_002.png	No Finding	2	5	69	F
18	00000005_003.png	No Finding	3	5	69	F
19	00000005_004.png	No Finding	4	5	70	F
20	00000005_005.png	No Finding	5	5	70	F
21	00000005_006.png	Infiltration	6	5	70	F
22	00000005_007.png	Effusion Infiltration	7	5	70	F
23	00000006_000.png	No Finding	0	6	81	M
24	00000007_000.png	No Finding	0	7	82	M
25	00000008_000.png	Cardiomegaly	0	8	69	F

dataLabel

+

Hình 12: Tổ chức dữ liệu trong file csv

Chương trình python giúp phân loại file ảnh dựa vào file csv vào file nén (zip) để tiện cho việc upload và huấn luyện mô hình.

```
import csv
import zipfile
counterInf = 0
counterNo = 0
counterNotfind1 = 0
counterNotfind2 = 0
with zipfile.ZipFile("dataInf.zip", 'w') as fileZipInf:
    with zipfile.ZipFile("dataNoF.zip", 'w') as fileZipNoF:
        with open("dataLable.csv", 'r') as fileCsv:
            csvRead = csv.reader(fileCsv)
            for data in csvRead:
                if data[1] == "Infiltration":
                    try:
                        fileZipInf.write(data[0])
                        counterInf = counterInf + 1
                        print("Success add this file to achirve data set Infiltration issue
"+str(data[0]))
                    except:
                        counterNotfind1 += 1
                elif data[1] == "No Finding":
                    try:
                        fileZipNoF.write(data[0])
                        print("Success add this file to achirve data set no findding
issue"+str(data[0]))
                    except:
                        counterNo += 1
                except:
                    counterNotfind2 += 1
print("Total leng Inf issue: " + str(counterInf) + "file. And " + str(counterNotfind1) + " is not
fund")
print("Total leng No Findding: " + str(counterNo) + "file. And " + str(counterNotfind2) + " is
not fund")
```

Hình 13: Source code python phân loại file ảnh và nén vào file zip

```
app.py
1 import csv
2 import zipfile
3 counterInf = 0
4 counterNo = 0
5 counterNotfind1 = 0
6 counterNotfind2 = 0
7 with zipfile.ZipFile("dataInf.zip", 'w') as fileZipInf:
8     with zipfile.ZipFile("dataNoF.zip", 'w') as fileZipNoF:
9         with open("dataLable.csv", 'r') as fileCsv:
10             csvRead = csv.reader(fileCsv)
11             for data in csvRead:
12                 if data[1] == "Infiltration":
13                     try:
14                         fileZipInf.write(data[0])
15                         counterInf = counterInf + 1
16                         print("Success add this file to achirve data set Infiltration issure "+str(data[0]))
17                     except:
18                         counterNotfind1 += 1
19                 elif data[1] == "No Finding":
20                     try:
21                         fileZipNoF.write(data[0])
22                         print("Success add this file to achirve data set no findding issure"+str(data[0]))
23                         counterNo += 1
24                     except:
25                         counterNotfind2 += 1
26 print("Total leng Inf issure: " + str(counterInf) + "file. And " + str(counterNotfind1) + " is not fund")
27 print("Total leng No Findding: " + str(counterNo) + "file. And " + str(counterNotfind2) + " is not fund")
28
```

Hình 14: File ảnh source code thực tế trên subline text

Để có thể thực thi source code python trên ta dùng lệnh trên powershell hoặc command port của PC.

Dùng lệnh : cd để di chuyển đến thư mục chứa file app.py

Dùng lệnh: python app.py để thực thi source code python, compiler là chương trình python3 được cài đặt trên windows và đã được setup môi trường.

Kết quả thực hiện source code app.py

```
Administrator: Windows Powe
PS C:\Users\nttoa> d:
PS D:\> cd .\DataBase\Xray\images_001\images\
PS D:\DataBase\Xray\images_001\images> python app.py
```

Hình 15: Thực thi file app.py để phân loại ảnh

```

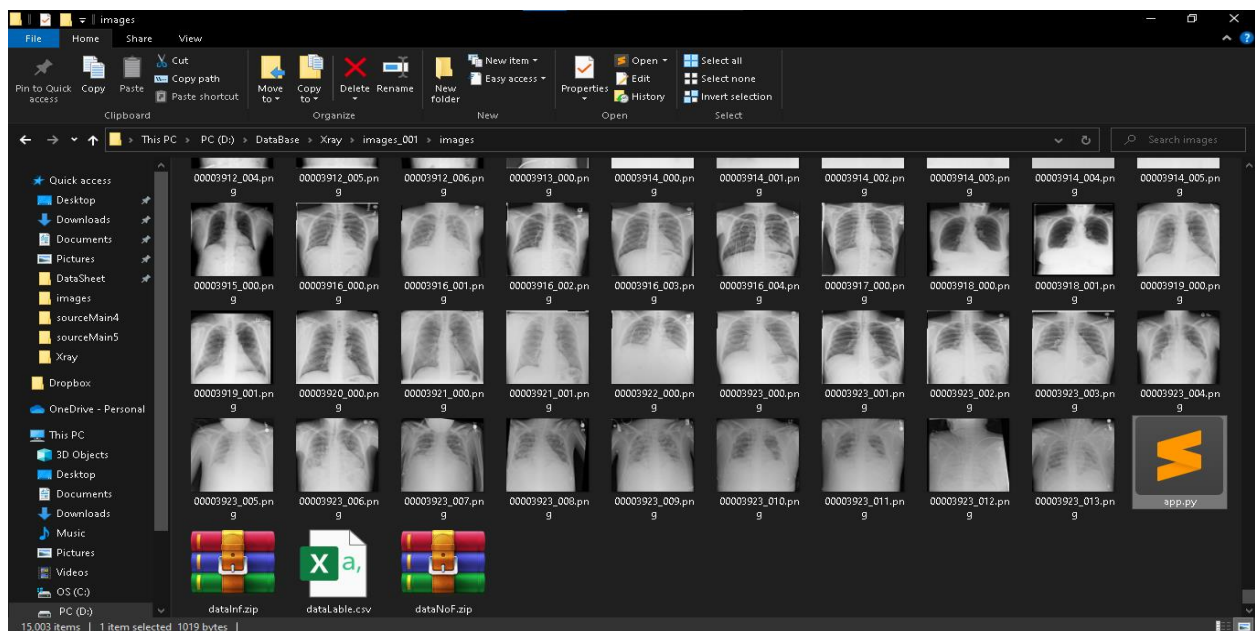
Success add this file to achirve data set Infiltration issure 00003915_000.png
Success add this file to achirve data set no findding issure00003916_000.png
Success add this file to achirve data set no findding issure00003916_003.png
Success add this file to achirve data set Infiltration issure 00003916_004.png
Success add this file to achirve data set no findding issure00003917_000.png
Success add this file to achirve data set no findding issure00003919_000.png
Success add this file to achirve data set no findding issure00003920_000.png
Success add this file to achirve data set no findding issure00003921_000.png
Success add this file to achirve data set no findding issure00003921_001.png
Success add this file to achirve data set no findding issure00003923_001.png
Success add this file to achirve data set Infiltration issure 00003923_002.png
Success add this file to achirve data set no findding issure00003923_004.png
Success add this file to achirve data set no findding issure00003923_005.png
Success add this file to achirve data set Infiltration issure 00003923_011.png
Success add this file to achirve data set no findding issure00003923_012.png
Success add this file to achirve data set no findding issure00003923_013.png
Total leng Inf issure: 1136file. And 8411 is not fund
Total leng No Findding: 8753file. And 51608 is not fund
PS D:\DataBase\Xray\images_001\images>

```

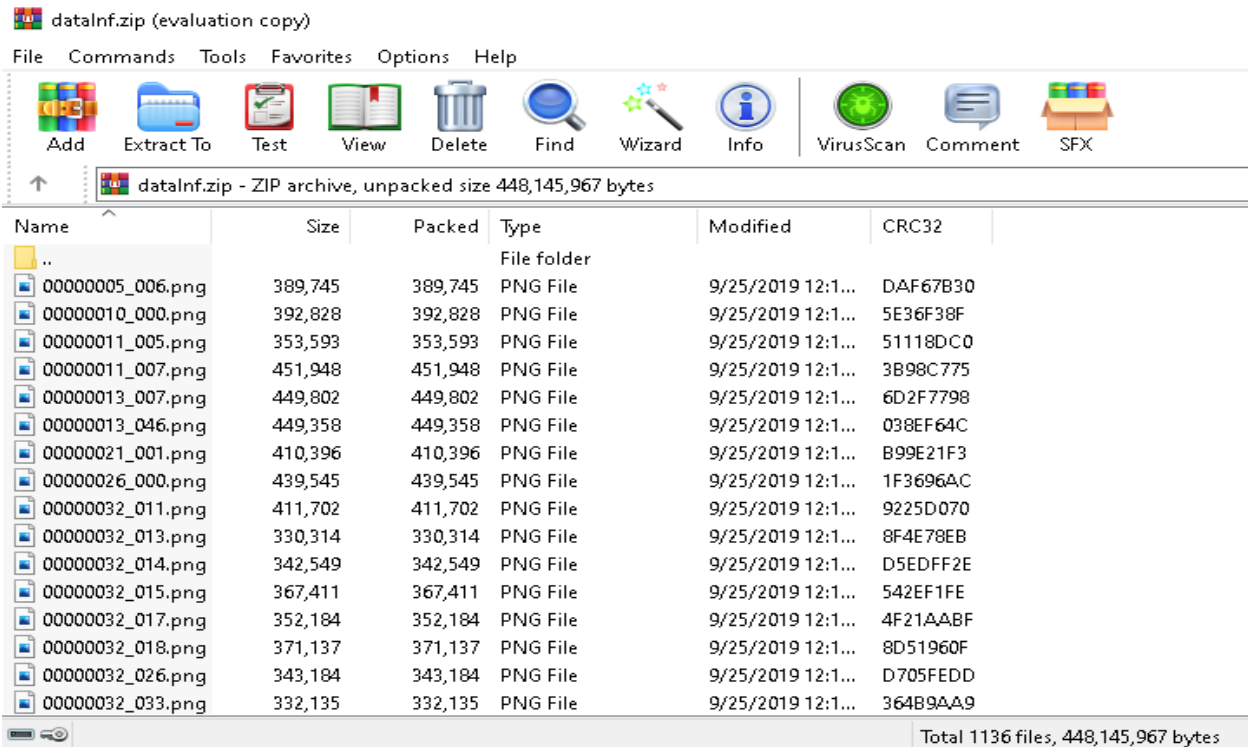
Hình 16: Kết quả trả về của source code phân loại file

Như vậy source code trên sẽ trả về tổng số lượng file đã được thêm thành công vào file nén zip. Và tổng số lượng file không tìm thấy. Và các file được tìm thấy sẽ được phân loại vào 2 tệp nén dành cho hai bộ data có lable khác nhau.

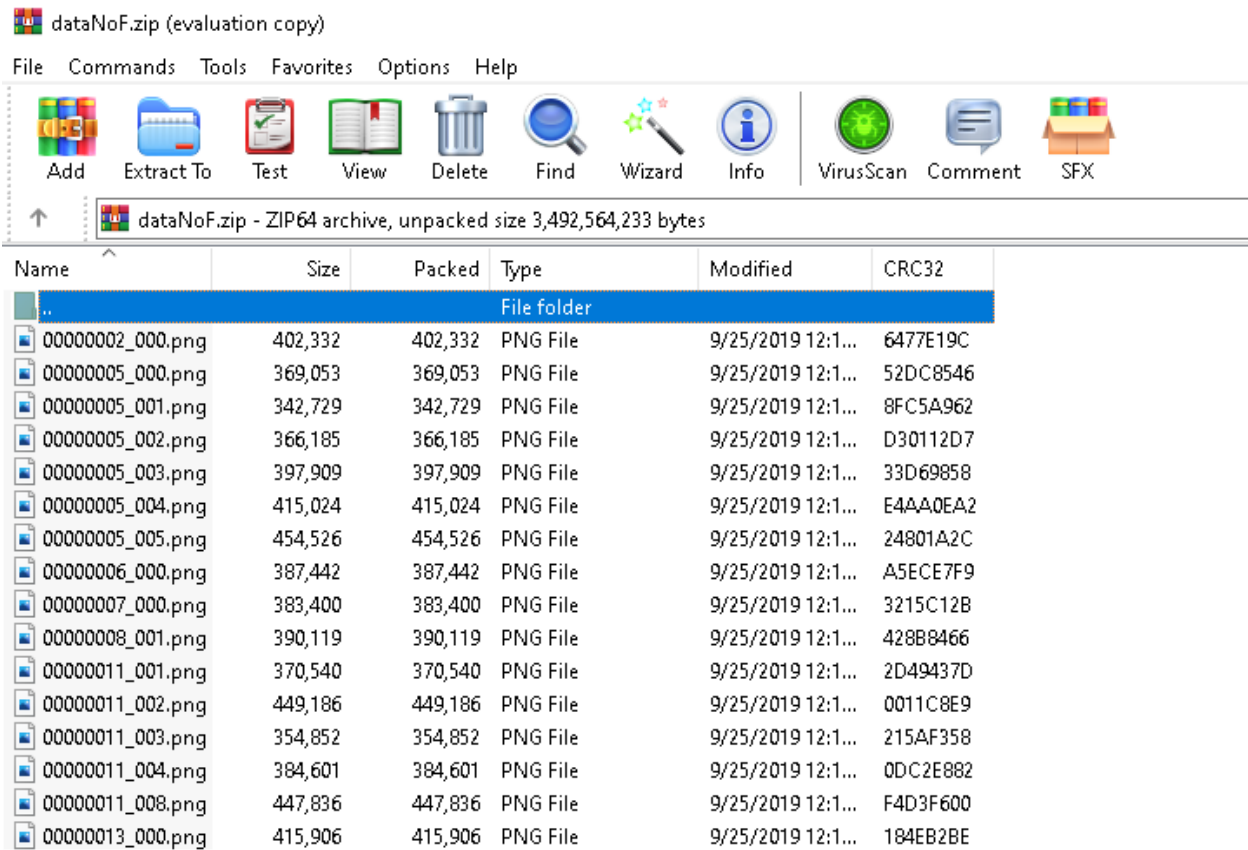
Các file ảnh theo đúng lable sẽ được nén vào 2 tệp tin nén bao gồm dataInf.zip và dataNoF.zip



Hình 17: Kết quả source code phân loại và nén file.



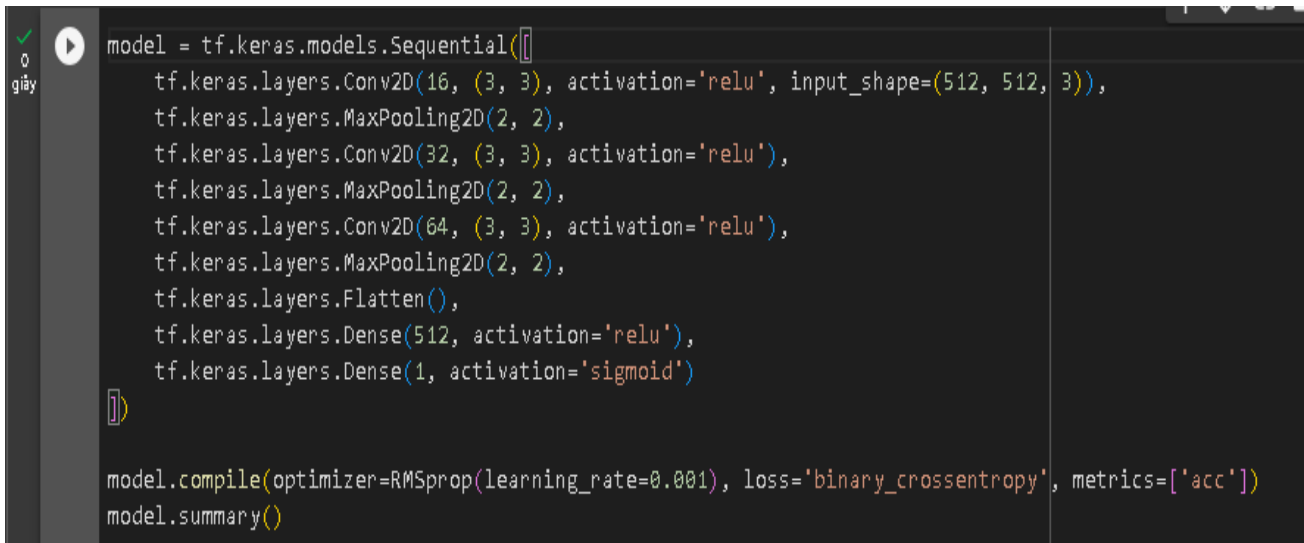
Hình 19: Các file được nén vào file dataInf.zip



Hình 18: Các file ảnh được nén vào file dataNoF.zip

2. Xây dựng mô hình

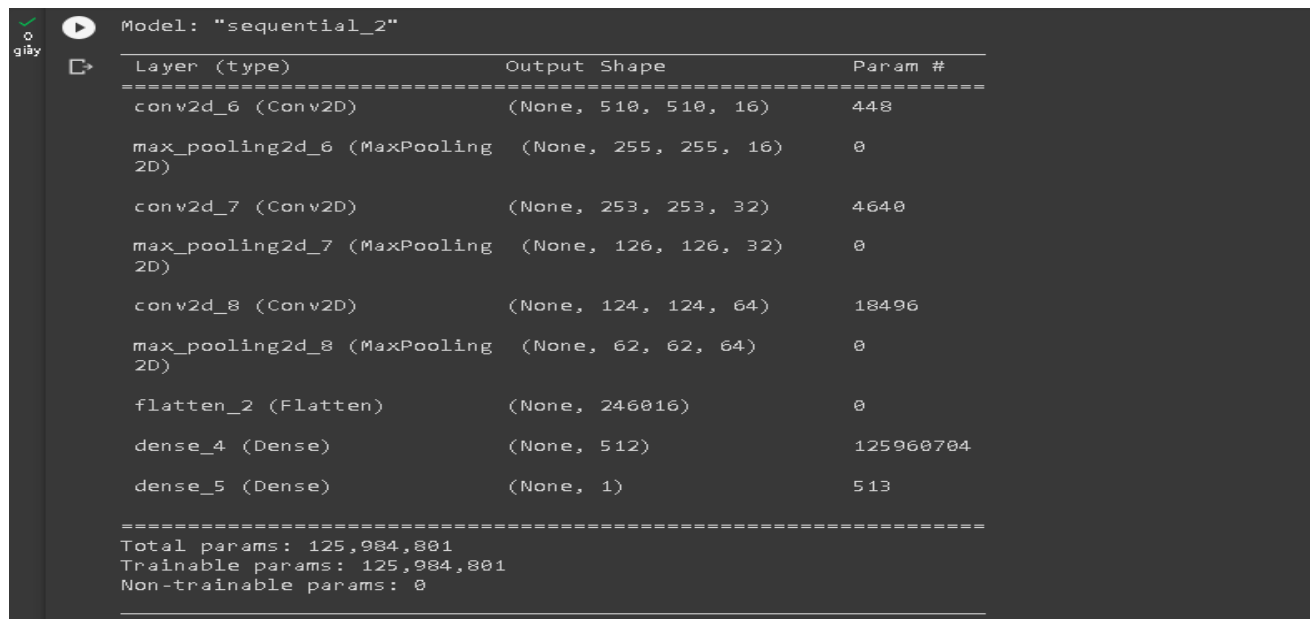
Dữ liệu đầu vào ban đầu có kích thước khá lớn 1024x1023x3 sẽ gây ra áp lực lớn cho phần cứng khi huấn luyện model. Vì vậy khi xây dựng model sẽ rút nhỏ kích thước của dữ liệu đầu vào lại còn khoảng 512x512x3. Vậy vậy ta có đoạn code python xây dựng model như sau.



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

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

Hình 21: Xây dựng model



```
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 510, 510, 16)	448
max_pooling2d_6 (MaxPooling 2D)	(None, 255, 255, 16)	0
conv2d_7 (Conv2D)	(None, 253, 253, 32)	4640
max_pooling2d_7 (MaxPooling 2D)	(None, 126, 126, 32)	0
conv2d_8 (Conv2D)	(None, 124, 124, 64)	18496
max_pooling2d_8 (MaxPooling 2D)	(None, 62, 62, 64)	0
flatten_2 (Flatten)	(None, 246016)	0
dense_4 (Dense)	(None, 512)	125960704
dense_5 (Dense)	(None, 1)	513

```
-----
Total params: 125,984,801
Trainable params: 125,984,801
Non-trainable params: 0
-----
```

Hình 20: Kết quả xây dựng model

Tổng số các tham số của model là gần 126 triệu vì vậy huấn luyện model sẽ tốn chút ít thời gian.

Vui lòng tham khảo source tại đường link:

<https://colab.research.google.com/drive/1qt7c57P01JiF496uSCcIk24-zNT7fdI3?usp=sharing>

```
# Note that this may take some time.
history = model.fit(train_generator, epochs=10, steps_per_epoch=20,
                    validation_data=validation_generator, validation_steps=6)
```

```
Epoch 1/10
20/20 [=====] - ETA: 0s - loss: 4.2712 - acc: 0.5015 WARNING:tensorflow:Your input ran out of
20/20 [=====] - 1024s 50s/step - loss: 4.2712 - acc: 0.5015 - val_loss: 0.6646 - val_acc: 0.6
Epoch 2/10
20/20 [=====] - 86s 4s/step - loss: 0.6962 - acc: 0.5305
Epoch 3/10
20/20 [=====] - 49s 2s/step - loss: 0.6857 - acc: 0.5647
Epoch 4/10
20/20 [=====] - 48s 2s/step - loss: 0.7201 - acc: 0.5249
Epoch 5/10
20/20 [=====] - 48s 2s/step - loss: 0.6684 - acc: 0.5922
Epoch 6/10
20/20 [=====] - 50s 2s/step - loss: 0.6547 - acc: 0.6165
Epoch 7/10
20/20 [=====] - 48s 2s/step - loss: 0.6424 - acc: 0.6385
Epoch 8/10
20/20 [=====] - 50s 2s/step - loss: 0.6178 - acc: 0.6471
Epoch 9/10
20/20 [=====] - 47s 2s/step - loss: 0.5807 - acc: 0.6690
Epoch 10/10
20/20 [=====] - 48s 2s/step - loss: 0.5514 - acc: 0.6945
```

Hình 22: Kết quả huấn luyện model

3. Kết quả huấn luyện model

Accuracy cao nhất của model là 0.6945 với loss function là 0.5514.

Tên	↑	Chủ sở hữu
	Test	 tôi
	Train	 tôi
	model.h5	 tôi

Hình 23: model.h5 chứa các trọng số của model sau khi huấn luyện

Lưu các trọng số sau khi huấn luyện model vào file model.h5

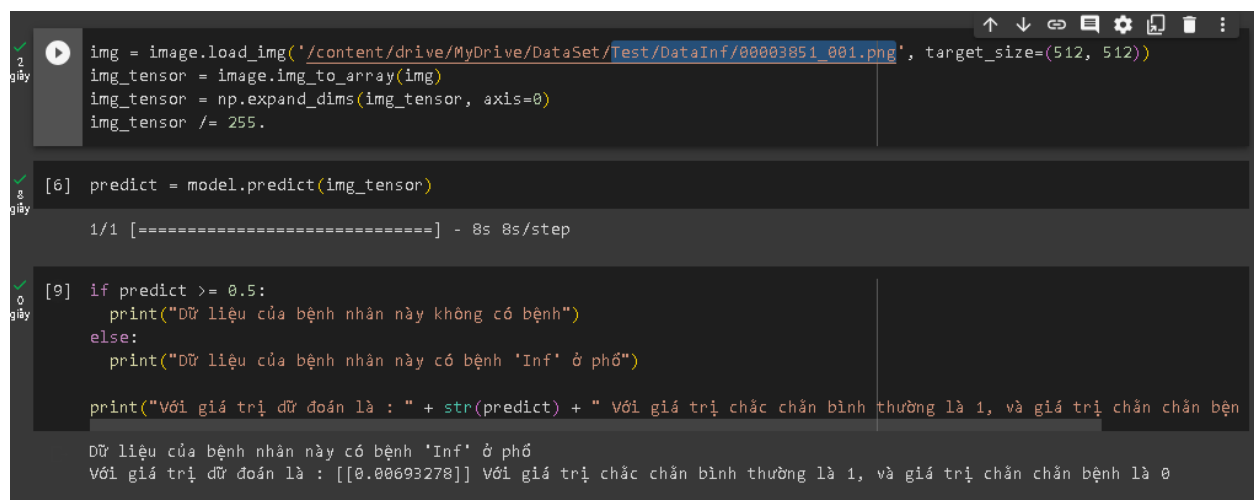
4. Thực hành dự đoán kết quả của model.

Các bước để thực hành dự đoán như sau:

B1: Load bức ảnh vào một biến, Định dạng cho phù hợp với cấu trúc đầu vào của model.

B2: Ép biến chứa dữ liệu ảnh vào đầu vào của model.

B3: Đọc kết quả đầu ra, biến đổi số liệu đầu ra trực quan hơn cho người dùng.



```
img = image.load_img('/content/drive/MyDrive/DataSet/Test/DataInf/00003851_001.png', target_size=(512, 512))
img_tensor = image.img_to_array(img)
img_tensor = np.expand_dims(img_tensor, axis=0)
img_tensor /= 255.

[6] predict = model.predict(img_tensor)

1/1 [=====] - 8s 8s/step

[9] if predict >= 0.5:
    print("Dữ liệu của bệnh nhân này không có bệnh")
else:
    print("Dữ liệu của bệnh nhân này có bệnh 'Inf' ở phổi")

print("Với giá trị dữ đoán là : " + str(predict) + " Với giá trị chắc chắn bình thường là 1, và giá trị chắn chắn bện

Dữ liệu của bệnh nhân này có bệnh 'Inf' ở phổi
Với giá trị dữ đoán là : [[0.00693278]] Với giá trị chắc chắn bình thường là 1, và giá trị chắn chắn bệnh là 0
```

Hình 24: Source code phần thực hành chẩn đoán bệnh.

Khối lệnh đầu tiên tương ứng với bước đầu tiên, load dữ liệu vào một biến là tiến hành xử lý cho phù hợp với cấu trúc đầu vào của model. Bao gồm các công việc như chuyển định dạng file ảnh sang cấu trúc mảng dữ liệu, Thêm đánh số các chiều của mảng, chia tất cả các giá trị của pixel cho 255, Mục đích để đầu vào input của model có giá trị từ 0 đến 1.

Khối lệnh thứ 2: Có chức năng tương tự bước 2, ép biến vào đầu vào của model và tiếp hành tính toán.

Khối lệnh thứ 3: Có chức năng của bước 3. Mục đích chuyển giá trị dự đoán của mạng neural network thành lời để trực quan cho người dùng. Cụ thể nếu giá trị dự đoán trả về kết quả là một số lớn hơn 0.5 thì người đó bình thường, không có mắc bệnh ở phổi.

Nếu kết quả của model trả về là một số nhỏ hơn 0.5 thì kết luận người đó có khả năng rất cao đã mắc bệnh “Inf”.

Ngoài ra cần thực hiện chương trình viết bằng ngôn ngữ python tạo nên giao diện giao tiếp giữa người dùng và model để dễ dàng sử dụng hơn, Tuy giao diện được xây dựng theo giao diện dòng lệnh (terminal) nhưng cũng phần nào trở nên thân thiện với người dùng hơn đôi chút.

Source code chương trình giao diện giao tiếp người dùng và hệ thống:

Có thể tham khảo source code tại: https://github.com/nttoan-khiem/DA1_Predict-infIssure/blob/main/source.py

```
import time
import os
class data:
    name = ""
    address = ""
    status = ""
    result = ""
    def __init__(self, name, address):
        self.name = name
        self.address = address
        self.status = "undiagnosed"
        self.result = "unknow"
    def show(self):
```



```

        print("Name: "+self.name +" address: " +self.address + " status:
"+self.status+" result: " + self.result)

    def diagnos(self):

        img = image.load_img(self.address, target_size=(512, 512))

        img_tensor = image.img_to_array(img)

        img_tensor = np.expand_dims(img_tensor, axis=0)

        img_tensor /= 255.

        predict = model.predict(img_tensor)

        if predict >= 0.5:

            print("Dữ liệu của bệnh nhân này: \33[92m không có bệnh \33[0m ")

            self.status = "diagnosed"

            self.result = "\33[92m normal \33[0m"

        else:

            print("Dữ liệu của bệnh nhân này: \33[91m có bệnh 'Inf' ở phổi \33[0m ")

            self.status = "diagnosed"

            self.result = "\33[91m Infiltrate \33[0m"


mData = []

mData.append(data("Nguyen Thanh Test", "test/test.png"))

os.system("cls")

stringOpen = "Hello, welcome"

for char in stringOpen:

    print(char, end="")

    time.sleep(1)

```

```

stringOpen =
"=====

for char in stringOpen:
    print(char, end="")
    time.sleep(1)
print(" ")
print(" <>Initialization system: ")
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, AveragePooling2D
from keras.layers import Dense, Activation, Dropout, Flatten
import keras.utils as image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import load_model
import matplotlib.pyplot as plt
import zipfile
import random
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from shutil import copyfile
import cv2
import numpy as np
import matplotlib.pyplot as plt
print("1. Complete Initialization import library")
print("2. Initialization model....")

```

```

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

print("3. Complete Initialization model.")

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

print("4. Complete compile model")

model.load_weights('model.h5')

print("5. Complete load weights from the last update")

stringOpen =
"=====

for char in stringOpen:
    print(char, end="")
    time.sleep(0.4)

print(" ")

print("Initialization system successfull, with summary on below")

model.summary()

print("Screen will be cleared in 5 second")

```

```
time.sleep(5)
```

```
while 1:
```

```
    os.system("cls")
```

```
    print("          Main Menu          ")
```

```
    print("\n\n\n\n")
```

```
    print("<> Nhập command '/list' để in danh sách các ảnh có sẵn.")
```

```
    print("<> Nhập command '/predict' để chẩn đoán bệnh INF.")
```

```
    print("<> Nhập command: ", end="")
```

```
    command = input()
```

```
    if command == "/list":
```

```
        os.system("cls")
```

```
        while 1:
```

```
            counter = 0
```

```
            print("=====Danh sách ảnh=====")
```

```
            for num in mData:
```

```
                counter += 1
```

```
                print("<> " + str(counter) + " ", end="")
```

```
                num.show()
```

```
            print("Nhập lệnh '/add' để thêm bệnh nhân mới")
```

```
            print("Nhập lệnh '/edit' để thay đổi thông tin")
```

```
            print("Nhập lệnh '/rm' để xóa bỏ dữ liệu của bệnh nhân")
```

```
            print("\n\n\n")
```

```
            print("Nhập lệnh: ", end="")
```

```
            command = input()
```

```

if command == "/add":
    os.system("cls")
    while 1:
        counter = 0
        os.system("cls")
        print("=====Thêm dữ liệu bệnh nhân
mới=====")
        for num in mData:
            counter += 1
            print("<> " + str(counter) + " ",end="")
            num.show()
        print("\n \n \n")
        print("Nếu muốn thoát xin vui lòng nhập lệnh '/exit' vào ô tên của
bệnh nhân")

        print("Nhập tên bệnh nhân mới: ",end="")
        name = input()
        if name == "/exit":
            break

        print("Nhập địa chỉ ảnh của bệnh nhân mới", end="")
        address = input()
        mData.append(data(name,address))
        print("<> dữ liệu của bệnh nhân "+name+ " đã được thêm thành
công")

elif command == "/rm":
    print("Nhập số thứ tự của bệnh nhân cần xóa: ",end="")
    number = input()

```

```

        mData.pop(number-1)
        print("\nThông tin của bệnh nhân đã được xóa thành công")
        print("Nhấn bất kỳ phím nào để tiếp tục")
        number = input()
    elif command == "/exit":
        break
if command == "/predict":
    while 1:
        os.system("cls")
        print("=====Chẩn đoán bệnh=====")
        counter = 0
        print("\n \n \n")
        for num in mData:
            counter += 1
            print("<> " + str(counter) + " ",end="")
            num.show()
        print("\nNhập số thứ tự của bệnh nhân cần predict: ",end="")
        name = input()
        if name == "/exit":
            break
        else:
            try:
                name = int(name)
            except:
                print("Nhập mã số của bệnh nhân không đúng định dạng")

```

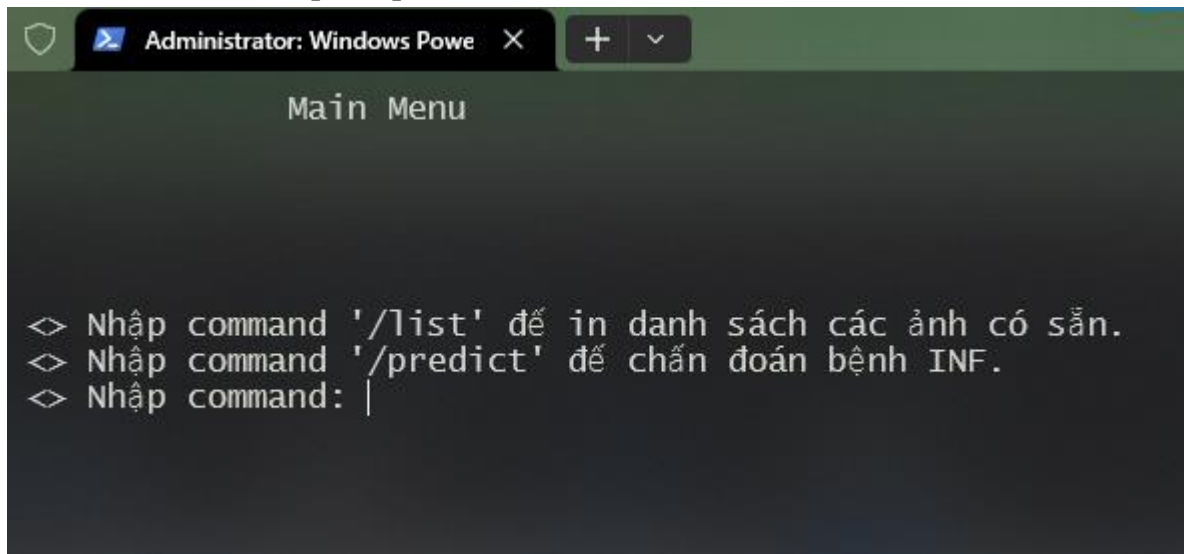
```

print("Nhấn bất kỳ phím nào để tiếp tục")
temp = input()
continue

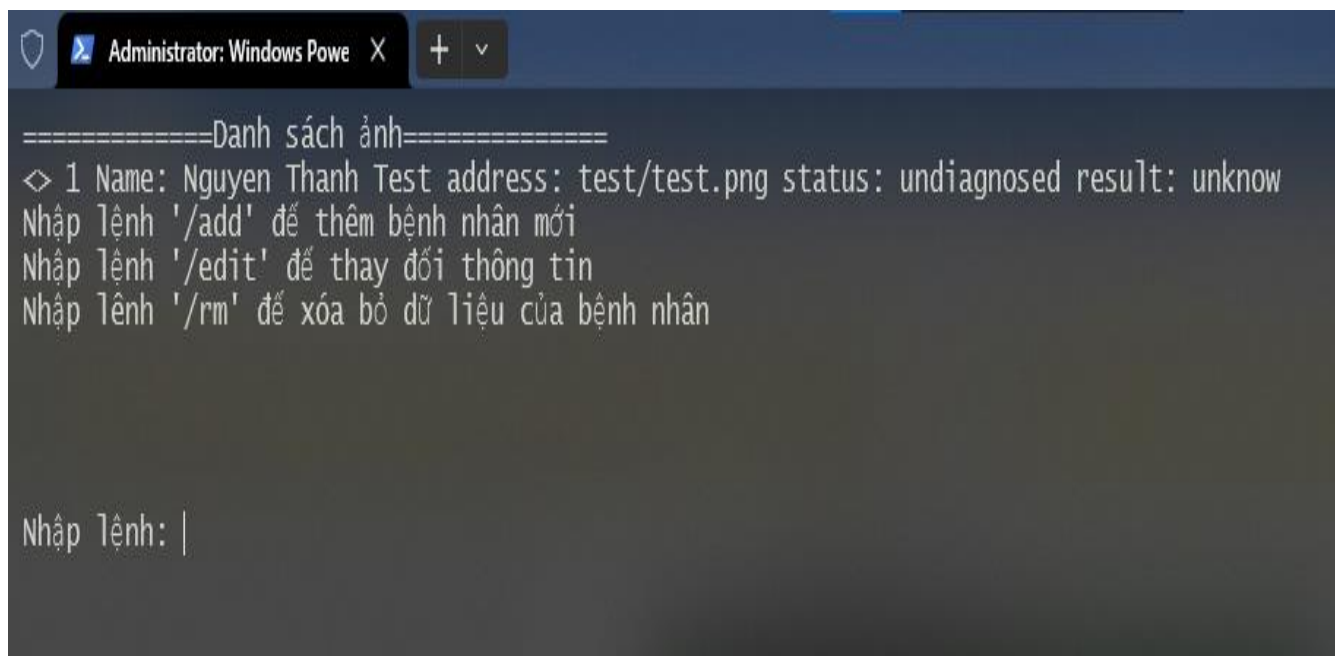
name -= 1
print("Kết quả chẩn đoán bệnh của bệnh nhân: "+mData[name].name)
mData[name].diagnos()
print("Nhấn bất kỳ phím nào để tiếp tục.")
img = cv2.imread(mData[name].address)
cv2.imshow("image", img)
cv2.waitKey(0)
cv2.destroyAllWindows()

temp = input()

```



Giao diện chính của chương trình sẽ có hướng dẫn nhằm dễ dàng sử dụng hơn, có hai tính năng chính là chức năng kiểm tra danh sách bệnh nhân để thực hiện chức năng này ta nhập lệnh “/list” vào ô dòng lệnh và nhấn Enter. Chức năng thứ hai là chức năng predict. Để vào chế độ này ta nhập lệnh “/redict” vào ô dòng lệnh.



```
Administrator: Windows Powe X + v

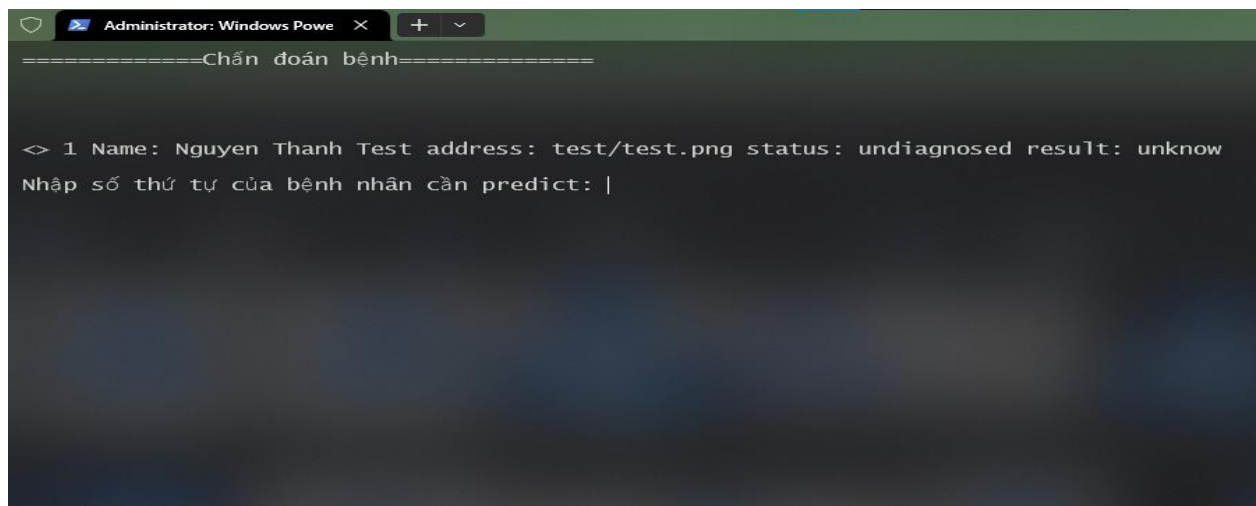
=====Danh sách ảnh=====
<> 1 Name: Nguyen Thanh Test address: test/test.png status: undiagnosed result: unknow
Nhập lệnh '/add' để thêm bệnh nhân mới
Nhập lệnh '/edit' để thay đổi thông tin
Nhập lệnh '/rm' để xóa bỏ dữ liệu của bệnh nhân

Nhập lệnh: |
```

Hình 26: Giao diện chương trình khi ở chế độ kiểm tra danh sách bệnh nhân

Ở chế độ kiểm tra danh sách bệnh nhân ta có thể xem được các thông tin như tên bệnh nhân, số thứ tự, địa chỉ dữ liệu của bệnh nhân, tình trạng đã được chẩn đoán hay chưa và kết quả chẩn đoán là gì. Tại chức năng này ta có thể thực hiện các thao tác như thêm bệnh nhân mới hoặc xóa bệnh nhân bằng cách nhập lệnh tương ứng của mỗi thao tác vào ô dòng lệnh.

Để thoát khỏi chế độ này cũng như mọi chế độ khác ta nhập lệnh “/exit” vào ô lệnh.

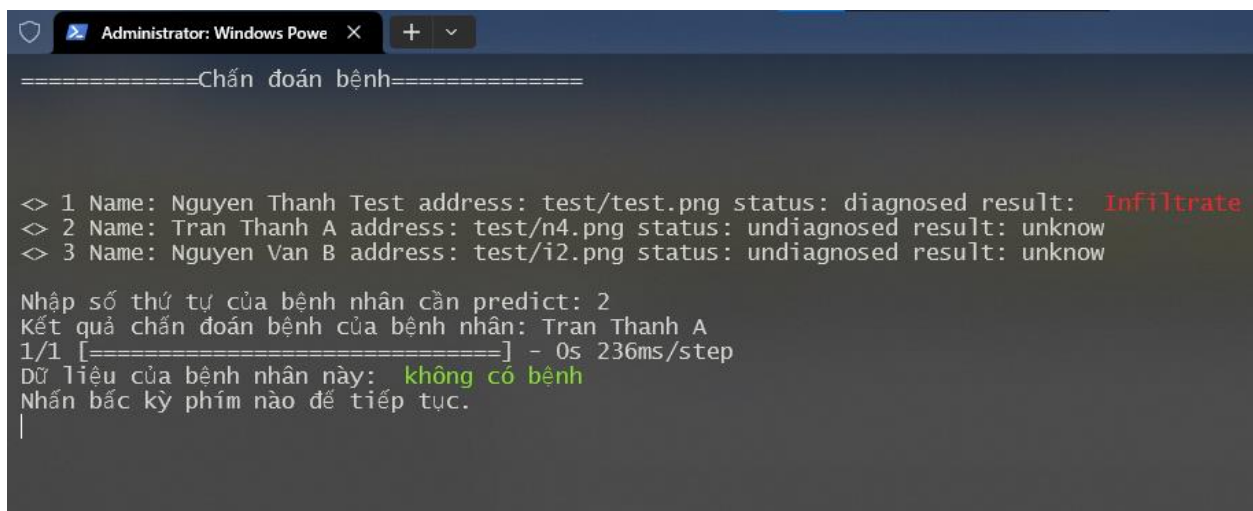


```
Administrator: Windows Powe
=====Chẩn đoán bệnh=====

<> 1 Name: Nguyen Thanh Test address: test/test.png status: undiagnosed result: unknow
Nhập số thứ tự của bệnh nhân cần predict: |
```

Hình 27: Giao diện ở chế độ predict

Ở chế độ predict chỉ cần nhập số thứ tự của bệnh nhân cần tiến hành chẩn đoán là được, Nếu muốn thoát khỏi chế độ predict thì chỉ cần nhập lệnh “/exit” là được.



```
Administrator: Windows Powe
=====Chẩn đoán bệnh=====

<> 1 Name: Nguyen Thanh Test address: test/test.png status: diagnosed result: Infiltrate
<> 2 Name: Tran Thanh A address: test/n4.png status: undiagnosed result: unknow
<> 3 Name: Nguyen Van B address: test/i2.png status: undiagnosed result: unknow

Nhập số thứ tự của bệnh nhân cần predict: 2
Kết quả chẩn đoán bệnh của bệnh nhân: Tran Thanh A
1/1 [=====] - 0s 236ms/step
Dữ liệu của bệnh nhân này: không có bệnh
Nhấn bất kỳ phím nào để tiếp tục.
|
```

Hình 28: Kết quả chẩn đoán chương trình trả về

Sau khi chẩn đoán xong chương trình sẽ trả về kết quả là có bệnh hoặc không có bệnh. Để tiếp tục ta nhấn phím Enter trên bàn phím. Ngoài ra để trực quan hơn, đồng thời cũng dễ dàng đọc kết quả hơn thì nếu kết quả là bị bệnh “Inf” thì chương trình sẽ in kết quả là màu đỏ và ngược lại nếu không bị bệnh “Inf” thì chương trình sẽ in kết quả là màu xanh.

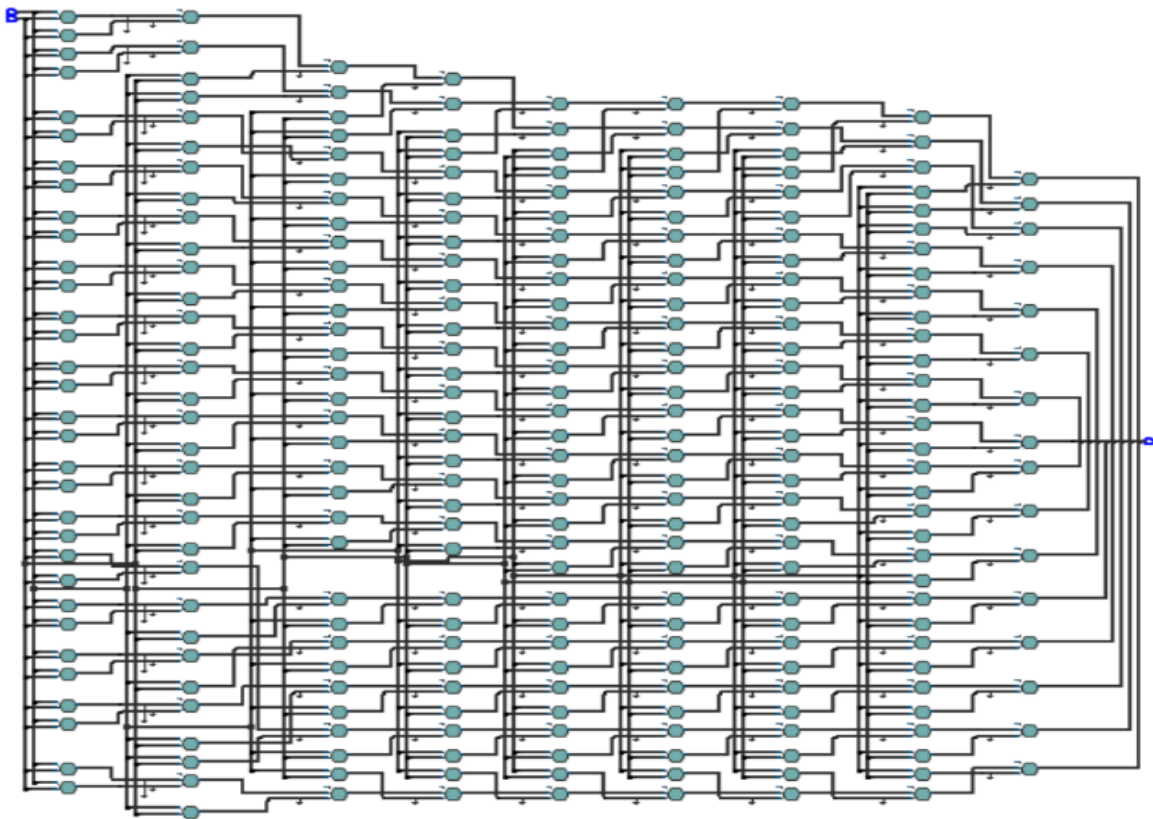
CHƯƠNG 4: KHẢ NĂNG PHÁT TRIỂN THÊM CỦA ĐỀ TÀI.

1. GPU phù hợp hơn cho các ứng dụng liên quan đến máy học.

Sự khác biệt và điểm nổi trội giữa việc tính toán trên GPU và tính toán thông thường trên CPU. Trong khi việc tính toán trên CPU (Central Processing Units) là bộ xử lý trung tâm, diễn ra theo cách tuần tự, nhưng GPU có khả năng tính toán song song nhiều phép tính một lúc. Với CPU, các tính toán phải phụ thuộc vào nhau và phép tính sau phải chờ phép tính trước được thực hiện xong. Tuy nhiên các phép tính trên GPU lại khác hoàn toàn, khi chúng được thực hiện song song với nhau. Ví dụ với tổ hợp tính $A + B + C \rightarrow D$, với CPU ta cần tính A, rồi tới B và tính C, sau đó mới trả kết quả ra D. Nhưng với GPU, ta có thể tính song song A, B và C cùng một lúc, và trả kết quả D ngay sau đó. Có thể nói, việc thêm nhiều nhân không thể giúp sức mạnh tính toán của CPU được tăng lên vì chúng vẫn phải tính tuần tự. Nếu thêm nhiều nhân cho GPU sẽ giúp chúng có sức mạnh tính toán vô cùng lớn và xử lý tính toán tốc độ cao với những ma trận khổng lồ trong mô hình máy học. Tuy GPU là rất nhanh và có thể đáp ứng được cho việc ứng dụng máy học, tuy nhiên để xử lý được những bức ảnh lớn có kích thước lớn và độ phân giải cao thì ngay cả GPU cũng sẽ có một thời gian delay tính toán nhất định. Vì vậy đối với những ứng dụng cần thời gian đáp ứng nhanh (Hard real time) như xử lý phát hiện vết tích bệnh, dấu hiệu bệnh trong phẫu thuật,...

2. Xây dựng mạng neural network trên ASIC hoặc FPGA.

Giả sử ta biết trước kích thước của mô hình neural network cần xây dựng (biết được số lớp, số neural trong một lớp, các hàm activation được sử dụng,...) Ta có thể xây dựng mô hình như vậy trên ASIC hoặc FPGA.

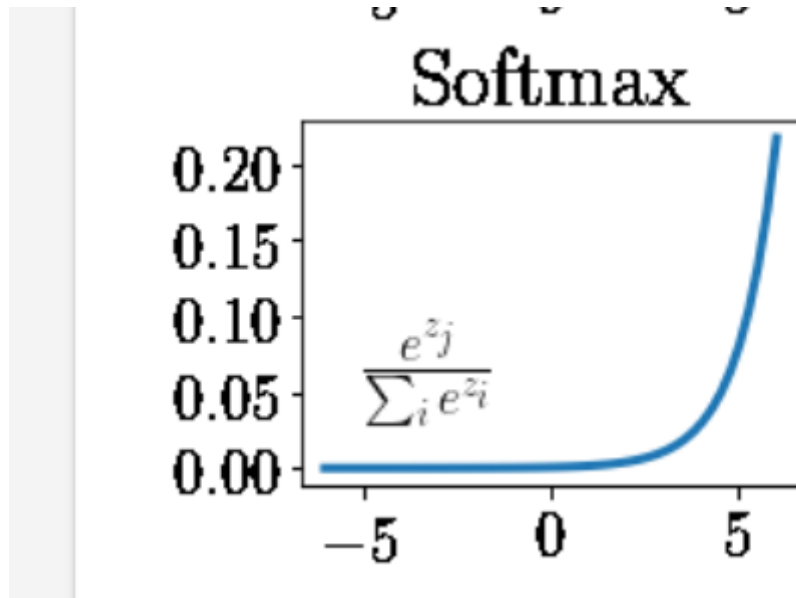


Hình 29: Neural network trên FPGA hoặc ASIC

Như vậy các kết quả tính toán của mô hình đều được tính toán thông qua các công logic được thiết kế sẵn, phù hợp với thuật toán tính toán của mô hình. Vì vậy độ trễ của mô hình không còn độ trễ của thuật toán tính toán mà chỉ còn lại độ trễ của công logic vì vậy thời gian trễ từ input đến output của mô hình sẽ được rút ngắn đáng kể.

3. Cải tiến mô hình neural network hiện tại và liên kết nhiều mô hình neural network lại với nhau.

Đối với ứng dụng phát hiện bệnh “Inf” ở phổ người như được trình bày phía trên thì mô hình chỉ mới dừng lại ở việc phát hiện được duy nhất một bệnh duy nhất là bệnh “Inf” và không thể phát hiện được các bệnh khác như bệnh viêm phổi, hay các tổn thương phổi,... Vì vậy ta cần cải thiện thêm mô hình hiện tại bằng cách chỉnh sửa lớp output thành dạng vector, mỗi chiều của vector sẽ tương ứng với một bệnh khác nhau. Như vậy hàm activation phù hợp nhất cho lớp output là hàm softmax.



Hình 30: Hàm activation softmax

Nhờ việc lớp output là một vector với nhiều chiều thì mô hình có thể đưa ra dự đoán được nhiều bệnh khác nhau.

Đồng thời ta cũng có thể liên kết nhiều mô hình neural network lại với nhau, xuất phát từ việc chụp hình Xray khoang ngực khá tốn kém và khi chụp Xray có hại cho sức khỏe của con người, vì vậy ra có thể xây dựng một mô hình neural network thêm vào trước với chức năng input là các thông số của bệnh nhân như huyết áp, các triệu chứng của bệnh nhân và kết quả đầu ra của mô hình này là có hay không sự nghi ngờ có bệnh ở phổi, Nếu có thì tiến hành khuyến nghị bệnh nhân chụp Xray khoang ngực và đưa đến mô hình neural network thứ hai hay là chẩn đoán bệnh từ ảnh Xray đã chụp, như vậy sẽ tiếp kiệm được tài nguyên y tế, thời gian,...

PHẦN KẾT LUẬN

Sau khi tham khảo các tài liệu, và được sự giúp đỡ của giáo viên hướng dẫn đã xây dựng được mô hình CNN (Convolution neural network) để chẩn đoán được bệnh “Inf” ở phổi người góp phần hỗ trợ y bác sĩ chẩn đoán bệnh với độ chính xác hơn 95% đối với bộ dữ liệu test mà đồ án này sử dụng.

Đồng thời tạo giao diện người dùng giao tiếp giữa người dùng và mô hình CNN nhằm việc giao tiếp với mô hình trở nên đơn giản hơn, dễ sử dụng hơn tuy mới dùng lại ở dạng giao diện dòng lệnh (command line). Đồng thời thực hiện được các thao tác như thêm, sửa, xóa dữ liệu là bước đầu của việc quản lý cơ sở dữ liệu.

Đánh giá về quá trình hoàn thành đề tài, được sự hướng dẫn của giáo viên hướng dẫn, cũng như các thư viện được tạo sẵn, bài đồ án đã xây dựng thành công mô hình chẩn đoán và đạt được độ chính xác đáng kỳ vọng.

Đồng thời xin được gửi lời cảm ơn chân thành đến Kaggle đã cung cấp miễn phí bộ dữ liệu cho việc huấn luyện mô hình CNN. Và chương trình colab của google đã cho phép cùng sử dụng tài nguyên máy chủ cho việc huấn luyện mô hình.

TÀI LIỆU THAM KHẢO

1. Adrian Brosebrock, Implementing feedforward neural networks with Keras and TensorFlow, (06/03/2021), truy cập tại <https://pyimagesearch.com/2021/05/06/implementing-feedforward-neural-networks-with-keras-and-tensorflow/>
2. Nguyễn Thanh Tuấn, Deep learning cơ bản (2019), bài 3 neural network, truy cập tại <https://nttuan8.com/bai-3-neural-network/>
3. Victo Zhou, Keras for Beginners: Implementing a Convolutional Neural Network, (08/08/2020), truy cập tại <https://victorzhou.com/blog/keras-cnn-tutorial/>
4. Kaggle, NIH Chest X-rays, truy cập ngày 15/07/2023 tại <https://www.kaggle.com/datasets/nih-chest-xrays/data>