DOI: https://doi.org/10.62985/j.huit_ojs.vol24.no4.112

THIẾT KẾ CHƯƠNG TRÌNH CHẨN ĐOÁN DỰA TRÊN X-QUANG NGỰC

Ngô Hoàng Ấn[1*], Lê Minh Thanh[1], Lê Trạch Dinh[2]

[1]*Trường Đại học Công Thương Thành phố Hồ Chí Minh*

[2]*Công ty Splus Software*

*Email: *annh@huit.edu.vn*

Ngày nhận bài: 02/11/2022; Ngày chấp nhận đăng: 16/8/2023

TÓM TẮT

Tình trạng, các thành phần và các cấu trúc lân cận của ngực có thể được đánh giá thông qua chẩn đoán hình ảnh X-quang ngực (XQN). Tuy nhiên, dựa vào hình ảnh XQN để quan sát và rút ra các chẩn đoán chính xác sẽ tốn khá nhiều thời gian dù đã là bác sĩ có nhiều kinh nghiệm. Hơn nữa, để tránh tối đa những sai sót trong chẩn đoán bệnh bởi các yếu tố con người như bác sĩ mệt mỏi khi nhìn quá nhiều hình ảnh XQN trong ngày thì việc tạo ra công cụ thông minh giúp bác sĩ giảm thời gian kiểm tra và chẩn đoán chính xác hình ảnh XQN sẽ tiết kiệm đáng kể thời gian và chi phí chẩn đoán bệnh qua hình ảnh XQN. Bài báo này sẽ nghiên cứu các giải thuật tiên tiến của mạng nơ-ron trong học sâu và ứng dụng các giải thuật này vào bộ dữ liệu lớn các hình ảnh XQN để thiết kế mô hình học sâu cho một số loại bệnh thường gặp. Các mô hình đề xuất giúp cho người dùng có thể tự chẩn đoán hình ảnh XQN của chính mình hay bác sĩ có thể có kênh tư vấn chẩn đoán bệnh nhanh, chính xác và hàng loạt từ các hình ảnh XQN thông qua chương trình chẩn đoán được viết trên nền tảng trình duyệt web.

Từ khóa: X-quang ngực, trí tuệ nhân tạo, chẩn đoán bệnh.

1. MỞ ĐẦU

Việc đọc X-quang ngực (XQN) cần rất nhiều kinh nghiệm chuyên môn cùng kinh nghiệm thực tế của các bác sĩ [1]. Ngoài ra, việc xem xét kỹ lưỡng hình ảnh XQN cũng mất khá nhiều thời gian dù đã là bác sĩ có nhiều kinh nghiệm. Việc giúp bác sĩ giảm thời gian kiểm tra hình ảnh XQN cho những bệnh phổ biến dựa trên XQN như viêm phổi, ung thư, tràn khí màn phổ, tim bất thường, … là rất cần thiết nhằm giảm thời gian và chi phí chẩn đoán bệnh qua hình ảnh XQN [2, 3].

Người bình thường không có nhiều kinh nghiệm chuyên môn cũng mong muốn có thể kiểm tra hình ảnh XQN của chính mình để xem khả năng mình có thể bị những bệnh nào. Ngoài ra, khi có nhiều bệnh nhân bị bệnh cùng một lúc thì không thể tuần tự chờ bác sĩ xác nhận kịp. Vì vậy, cần công cụ có thể phân tích hàng loạt hình ảnh XQN cùng một lúc để phân loại sàng lọc bệnh trước. Vấn đề đặt ra là công cụ thông minh nào có thể đọc được hình ảnh XQN để hỗ trợ các yêu cầu trên.

Dựa trên bộ dữ liệu lớn của các hình ảnh XQN đã chụp được trong thực tế được dán nhãn một số loại bệnh phổ biến, bài báo này sẽ ứng dụng mạng nơ-ron trong học sâu (Deep Learning: DL) để thiết kế các mô hình huấn luyện [4, 5]. Dựa trên chính các mô hình này, các hình ảnh XQN mới được đưa vào sẽ cho ra các chẩn đoán theo từng loại bệnh. Trên nền tảng học sâu này, công cụ trợ giúp những chẩn đoán bệnh liên quan đến ngực được xây dựng.

Phát hiện các bất thường và dự đoán khả năng xuất hiện bệnh trong hình ảnh XQN đòi hỏi các bác sĩ chuyên ngành phải có kinh nghiệm chuyên môn cao. Các bác sĩ sẽ đọc hình ảnh XQN và đưa ra những chẩn đoán dựa trên kinh nghiệm chuyên môn được tích lũy qua nhiều năm công tác. Nhằm tránh các sai sót trong chẩn đoán cũng như là công cụ trợ giúp những bác sĩ chẩn đoán các bệnh từ đọc XQN tốt hơn thì máy tính với trí tuệ nhân tạo (Artificial Intelligence: AI) gần đây đã giúp đỡ bác sĩ đọc nhanh các XQN và chẩn đoán chính xác bệnh. Các máy tính ứng dụng trí tuệ nhân tạo này đã dựa trên bộ dữ liệu XQN rất lớn đến từ hàng chục nghìn bệnh nhân đã được phát hiện các loại bệnh cơ bản. Bộ dữ liệu XQN này sẽ được xử lý để khai thác và huấn luyện học sâu nhằm tạo ra các mô hình chẩn đoán thông minh mà từ các mô hình chẩn đoán này khi áp dụng trên một hình ảnh XQN mới sẽ cho ra kết quả chẩn đoán chính xác. Các mô hình mà máy tính được học sâu giúp chẩn đoán và phát hiện các bệnh trên XQN đã được hiện thực tốt trên thực tế. Chúng không chỉ giúp tư vấn cho những bác sĩ chuyên khoa mà có thể còn giúp cho những người bình thường (không có chuyên môn sâu về y khoa) có thể tự chẩn đoán trên chính XQN của mình.

Nhiều nghiên cứu đã sử dụng trí tuệ nhân tạo trong học sâu của máy tính để chẩn đoán một loại bệnh cụ thể có thể phát hiện trên hình ảnh XQN có thể được liệt kê như sau:

- Castiglioni, I. đã sử dụng XQN của các bệnh nhân tại hai bệnh viện ở Lombardy, Ý để huấn luyện và kiểm tra nội bộ một tập hợp gồm 10 mạng nơ-ron CNN (Convolutional Neural Network) với các XQN chủ yếu tại giường của 250 đối tượng mắc COVID-19 và 250 đối tượng không mắc COVID-19. Sau đó, Castiglioni đã thử nghiệm hệ thống này trên các máy chụp XQN tại giường của một nhóm độc lập gồm 110 bệnh nhân (74 mắc COVID-19, 36 không mắc COVID-19) từ một trong hai bệnh viện. Kết quả khi xác nhận chéo 10 lần thì mô hình trí tuệ nhân tạo do Castiglioni đề xuất đã phân loại được bệnh nhân mắc hay không mắc COVID-19 với độ chính xác 78% [6].
- Riêng trường hợp viêm phổi, một bệnh xảy ra ở phổi do nhiễm vi khuẩn, thì bệnh này có thể được chẩn đoán từ hình ảnh XQN bởi một bác sĩ chuyên khoa X-quang. Các chẩn đoán có thể chủ quan vì một số lý do như biểu hiện của bệnh có thể chưa được rõ ràng trên hình ảnh XQN hoặc có thể bị nhầm lẫn với các bệnh khác. Vì vậy, cần có các chương trình chẩn đoán để tư vấn cho những bác sĩ lâm sàng. Ayan, E. và Unver, H. M. đã khai thác hai mô hình mạng nơ-ron CNN nổi tiếng là Xception và Vgg16 để chẩn đoán viêm phổi. Phương pháp học chuyển giao và tinh chỉnh trong giai đoạn huấn luyện được sử dụng trong hai mô hình này. Kết quả thử nghiệm cho thấy mạng Vgg16 vượt mạng Xception với độ chính xác lần lượt là 87% và 82% [7].
- Ung thư phổi là một bệnh nguy hiểm và cần chẩn đoán chính xác các hình ảnh XQN. Kanan, C. và Cottrell, G. đã nghiên cứu dựa trên học sâu cùng với xử lý hình ảnh (loại trừ bóng xương) trên XQN để chẩn đoán ung thư phổi chính xác hơn [8]. Gordienko, Y. cùng các cộng sự đã tiền xử lý bộ dữ liệu trước không có bóng xương, rồi sau đó dùng học sâu để huấn luyện dữ liệu. Kết quả đã chứng minh hiệu quả và tính hữu dụng cao của học sâu trong chẩn đoán bệnh ung thư phổi [9].
- Behzadi-khormouji, H. đã sử dụng trí tuệ nhân tạo, đặc biệt là mạng nơ-ron sâu DCNN (Deep CNN) để trợ giúp bác sĩ chẩn đoán tốt hơn nhưng trên dữ liệu hình ảnh y tế tương đối nhỏ. Kỹ thuật học chuyển giao với các DCNN nổi tiếng được huấn luyện trước với tập dữ liệu ImageNet2 được khai thác để nâng cao độ chính xác. Kết quả là độ chính xác lên tới 94,67% [10].
- Cohen, J. P. đã đưa các nhà nghiên cứu học sâu đến gần hơn với các chuyên gia y tế. Ông đã phát triển hệ thống nguyên mẫu miễn phí rất dễ tiếp cận mà các chuyên gia y tế có thể sử dụng để hiểu thực tế của các công cụ học sâu để chẩn đoán dựa trên XQN.

Chương trình được thiết kế để trở thành ý kiến thứ hai, nơi người dùng có thể tải hình ảnh XQN vào chương trình để xác nhận hoặc hỗ trợ chẩn đoán [11].

Như vậy, nhiều cách tiếp cận XQN để chẩn đoán một bệnh cụ thể, sử dụng các giải thuật học sâu khác nhau, không chỉ để giúp cho chuyên gia y tế mà còn giúp cho nhà nghiên cứu trí tuệ nhân tạo và người dùng tiếp cận và thiết kế các mô hình được huấn luyện để tự họ có thể phát hiện chính xác các bệnh. Tuy nhiên, các mô hình huấn luyện học sâu và ứng dụng đọc phân tích XQN cho người dùng, không phải chuyên gia y tế, tại Việt Nam còn rất ít (hầu như không tìm thấy). Vì vậy, bài báo này sử dụng các giải thuật trong mạng nơ-ron học sâu của lĩnh vực trí tuệ nhân tạo để huấn luyện các mô hình nhằm ứng dụng vào chương trình chẩn đoán 14 bệnh thường gặp trên XQN. Đặc biệt, bài báo hướng đến mục tiêu là người dùng có thể sử dụng trình duyệt web để tự chẩn đoán trên file XQN nhưng dữ liệu hay file XQN vẫn còn trên máy người dùng và toàn bộ quy trình xử lý diễn ra cục bộ. Cụ thể, dựa trên bộ dữ liệu XQN của Viện Y tế Quốc gia Hoa Kỳ được chia sẽ trên trang Kaggle [13] gồm 112120 hình ảnh XR có dán nhãn bệnh từ 30805 bệnh nhân với 14 bệnh cơ bản, bài báo sẽ tiến hành dùng thuật toán trong mạng nơ-ron để huấn luyện các mô hình trí tuệ nhân tạo cho 14 loại bệnh phổ biến trên. Kế đến, bài báo xây dựng công cụ web giúp các bác sĩ hoặc người dùng có tải các hình ảnh XR lên hệ thống và dựa trên mô hình trí tuệ nhân tạo đã được huấn luyện, hệ thống sẽ đưa ra những hỗ trợ chẩn đoán các bệnh phổ biến trên XQN. Vì vậy, bài báo thể hiện hai đóng góp chính sau:

- Đánh giá, so sánh và phân tích những thuật toán học sâu để tạo nên các mô hình mà căn cứ vào những mô hình đó để chẩn đoán các loại bệnh thường gặp trong hình ảnh XQN.
- Thiết kế chương trình giúp cho người dùng tự kiểm tra hình ảnh XQN của chính mình và tư vấn hỗ trợ các nhân viên y khoa tránh sai sót khi đánh giá các hình ảnh XQN và giúp tiết kiệm chi phí nhân lực, thời gian và chi phí hình ảnh XQN (tránh lãng phí chụp XQN lại nhiều lần).

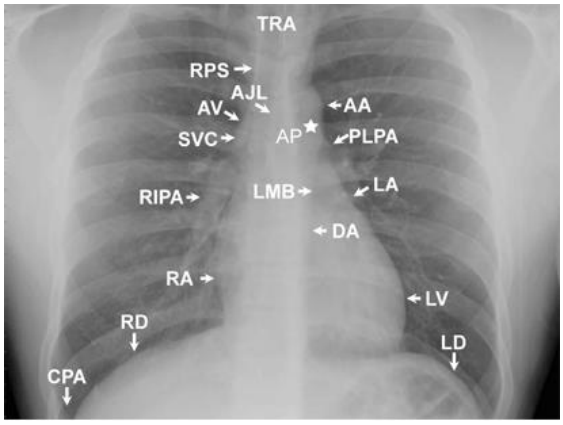
Phần 2 của bài báo giới thiệu tổng quan về hình ảnh XQN trong khi đó Phần 3 trình bày xây dựng chương trình chẩn đoán trên các hình ảnh XQN. Các kết quả minh họa được phân tích trong Phần 4, trong khi đó Phần 5 nêu ra các kết luận cho bài báo này.

2. TỔNG QUAN VỀ HÌNH ẢNH XQN

Hình ảnh XQN có thể được khai thác để xem các cấu trúc và các cơ quan trong lồng ngực như tim, phổi, phế quản, động mạch chủ, động mạch phổi, vùng ngực giữa (trung thất), xương ngực. Hình 1 cho thấy cấu trúc thường có trong hình ảnh XQN:

- Cung động mạch chủ (aortic arch: AA) thường chiếm ưu thế đường viền tim mạch vượt trội bên trái của bệnh nhân, mặc dù có thể là một phần của đường viền tim mạch ngay ở những người lớn tuổi.
- Thành bên trái của động mạch chủ đoạn xuống (descending aorta: DA) thường có thể nhìn thấy được khi đi xuống thấp hơn qua lồng ngực.
- Động mạch phổi cận bên trái (proximal left pulmonary artery: PLPA) được hiển thị trong khu vực rốn phổi trái nhưng kém hơn so với cung động mạch chủ. Động mạch phổi phân thùy trái (left interlobar pulmonary artery: LIPA) được ghi nhận là nằm dưới và nằm bên so với PLPA.
- Phế quản chính bên trái (left mainstem bronchus: LMB) thường thấy trên quan điểm trán ngay dưới đoạn động mạch phổi chính và động mạch phổi trái.

- Phần phụ của tâm nhĩ trái (left atrium: LA) nằm thấp hơn so với phế quản thân trái và dọc theo đường viền tim mạch bên trái. Các tâm thất trái (left ventricle: LV) hoàn thành phần còn lại của đường viền trung thất trái.
- Các tĩnh mạch chủ trên (superior vena cava: SVC) được nhìn thấy trong phần phía trên của đường viền trung thất của bệnh nhân.
- Đường cạnh phải khí quản (right paratracheal stripe: RPS) là sọc mô mềm được tạo ra bởi giao diện của bức tường khí quản bên phải và bên thùy trên. Gần RPS dưới ở góc khí quản bên phải có thể thấy tĩnh mạch azygos (azygos vein: AV).
- Động mạch phổi phân thùy phải (right interlobar pulmonary artery: RIPA) thoát qua rốn ngay ở phía dưới và sang hai bên và là cái bóng chiếm ưu thế trong khu vực này.
- Các tâm nhĩ phải (right atrium: RA) tạo thành biên dưới tim phải, và đôi khi chảy qua góc tâm hoành giữa tim và cơ hoành bên phải là cái bóng đại diện cho các tĩnh mạch chủ dưới (inferior vena cava: IVC).
- Các khí quản (trachea: TRA) thường được dễ dàng nhìn thấy trên PA (tư thế chụp sau – trước hoặc AP (tư thế chụp trước – sau) của hình ảnh XQN.
- Các đường viền của cơ hoành bên phải (right diaphragm: RD) và bên trái (left diaphragm: LD) có thể nhìn thấy rõ ràng.
- Các góc sườn hoành (costophrenic angle: CPA) có thể nhìn thấy ở phần dưới bên trái của ngực.
- Đường nối trước (anterior junction line: AJL) có thể được xem như là một đường xiên nằm trên trung thất đại diện cho các điểm tiếp xúc giữa hai phổi.



Hình 1. Các cấu trúc thường có trong hình ảnh XQN [12]

Một số phát hiện bất thường phổ biến có thể thấy khi chẩn đoán XQN là viêm phổi (bóng trắng hoặc mờ bất thường trên các trường phổi mà bình thường sẽ trông tối; áp xe trong phổi; tụ dịch giữa phổi và thành ngực có vẻ trắng hơn phổi và làm cho đường viền phổi sắc nét trên phim mờ hơn; phù phổi (chất lỏng tích tụ trong phổi hoặc các mạch máu) được coi là cảm giác khó chịu lan tỏa trên các trường phổi; kích thước tim; gãy xương sườn hoặc xương cánh; gãy đốt sống hoặc gãy đốt sống; trật khớp vai; ung thư phổi hoặc các khối khác ở phổi.

Bảng 1. Thông tin cơ bản của bộ dữ liệu hình ảnh XQN của Viện Y tế Quốc gia Hoa Kỳ

No	Item	Records	Type	Note
1	Image Index	112120	text	Tên file hình ảnh XQN Ví dụ: 00000001_000.png
2	Finding Labels	112120	text	Loại bệnh (phân loại bệnh được dán nhãn) Ví dụ: Có 1 nhãn: Cardiomegaly (tim to) Có nhiều nhãn: Pleural Effusion (tràn dịch màn phổi) |Pneumonia (viêm phổi) |Pneumothorax (tràn khí màn phổi)
3	Follow-up	112120	int	Giám sát
4	Patient ID	112120	int	ID của bệnh nhân
5	Patient Age	112120	int	Tuổi bệnh nhân
6	Patient Gender	112120	text	Giới tính bệnh nhân
7	View Position	112120	text	Hướng tia XQN
8	OriginalImageWidth	112120	int	Chiều rộng hình ảnh gốc
9	OriginalImageHeight	112120	int	Chiều cao hình ảnh gốc
10	OriginalImagePixelSpacing_x	112120	float	Khoảng cách pixel của hình ảnh gốc x
11	OriginalImagePixelSpacing_y	112120	float	Khoảng cách pixel hình ảnh gốc y

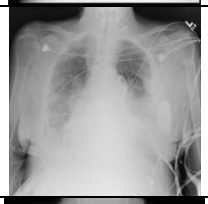
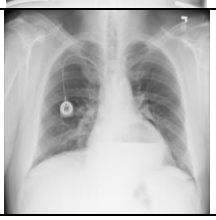
Bộ dữ liệu hình ảnh XQN được dùng trong bài báo này là bộ dữ liệu XQN của Viện Y tế Quốc gia Hoa kỳ [14]. Bộ hình ảnh này bao gồm 112120 hình ảnh chụp X-quang có nhãn bệnh từ 30805 bệnh nhân duy nhất. Đây là bộ dữ liệu có nhãn. Các thông tin cơ bản của bộ dữ liệu được nêu ra trong Bảng 1.

Mẫu dữ liệu sẽ có dạng sau:

	Image Index	Finding Labels	Follow-up #	Patient ID	Patient Age	Patient Gender	View Position	OriginalImage[Width	Height]	OriginalImagePixelSpacing[x	y]
45062	00011575_000.png	No Finding	0	11575	14	F	PA	2426	2205	0.143	0.143
31961	00008357_000.png	Infiltration	0	8357	47	F	PA	3056	2488	0.139	0.139
73309	00018044_016.png	Atelectasis	16	18044	34	F	AP	2500	2048	0.168	0.168

Các loại bệnh cơ bản trên nguồn dữ liệu của Viện Y tế Quốc gia Hoa Kỳ gồm: 14 loại bệnh có dán nhãn và 1 nhãn không tìm thấy bệnh (No Finding) như Bảng 2.

Bảng 2. Các loại bệnh cơ bản và hình ảnh XQN mẫu

STT	Nhãn bệnh	Lý giải bệnh	Hình ảnh XQN mẫu
1	Atelectasis	Xẹp phổi là tình trạng tắc nghẽn phổi hoặc chèn ép dẫn đến giảm hay mất khả năng trao đổi khí	
2	Cardiomegaly	Chứng tim to có thể xảy ra do một số nguyên nhân nhưng thường là kết quả của huyết áp cao hoặc bệnh động mạch vành	
3	Consolidation	Đông đặc phổi	
4	Edema	Phù nề phổi	
5	Effusion	Tràn dịch màng phổi hay tình trạng "phổi ứ nước" là sự tích tụ bất thường của chất lỏng trong khoang màng phổi	
6	Emphysema	Khí thủng là tình trạng liên quan đến tổn thương thành phế nang phổi	
7	Fibrosis	Xơ nang là một rối loạn di truyền ảnh hưởng chủ yếu đến phổi	
8	Hernia	Thoát vị thành bụng	

9	Infiltration	Xâm nhập là sự khuếch tán hoặc tích lũy của các chất lạ hoặc với số lượng vượt quá mức bình thường	
10	Mass	Khối u trên phổi	
11	Nodule	Bướu trên phổi	
12	Pleural_Thickening	Dày màng phổi chủ yếu do tràn dịch màng phổi gây nên	
13	Pneumonia	Viêm phổi cấp tính là một bệnh lý về nhiễm trùng cấp tính ở phổi	
14	Pneumothorax	Tràn khí màng phổi	

3. XÂY DỰNG CHƯƠNG TRÌNH CHẨN ĐOÁN TRÊN CÁC HÌNH ẢNH XQN

3.1. Sử dụng các giải thuật học sâu dùng mạng nơ-ron

Bài báo này dùng thư viện TensorFlow trên cơ sở công cụ dành riêng cho huấn luyện máy học của Google, đó là Colab phiên bản Pro để viết code (mã) định nghĩa cho ba giải thuật tiêu biểu của mạng nơ-ron CNN: GoogLeNet, ResNet và DenseNet. Sau đó, bài báo dùng bộ dữ liệu mẫu cũng được lấy từ trang kaggle có đầy đủ thuộc tính như bộ dữ liệu lớn nhưng chỉ chứa khoảng 5606 tập tin hình ảnh XQN. Ba giải thuật trên bộ dữ liệu mẫu này được sử dụng để đánh giá và so sánh độ chính xác trong chẩn đoán.

3.1.1. Định nghĩa các model

Số classification được định nghĩa bằng số len(all_lables) các bệnh cơ bản là 14.

A. Xây dựng mô hình GoogleNet phiên bản V3 dựa trên thư viện của TensorFlow như sau:

```
import tensorflow
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import AUC
from keras import backend as K
auc=AUC()
IMG_SIZE = (IMG_WH,IMG_WH)
base_model = InceptionV3(input_shape = (*IMG_SIZE, 3),
                                 include_top = False, weights = None)

x = base_model.output
# add a global spatial average pooling layer
x = GlobalAveragePooling2D()(x)
# and a logistic layer
predictions = Dense(len(all_labels), activation="sigmoid")(x)
googlenet_model = Model(inputs=base_model.input, outputs=predictions)
googlenet_model.compile(optimizer='adam', loss="binary_crossentropy",
              metrics=[tensorflow.keras.metrics.binary_accuracy])
googlenet_model.summary()
```

```
==================================================================================================
Total params: 21,831,470
Trainable params: 21,797,038
Non-trainable params: 34,432
```

Sử dụng lớp thư viện InceptionV3 thì với định nghĩa mô hình GoogleNet sẽ có hơn 21,8 triệu tham số được tạo ra.

B. Xây dựng mô hình ResNet phiên bản V2 dựa trên thư viện của TensorFlow như sau:

```
from tensorflow.keras.applications.inception_resnet_v2 import InceptionResNetV2
base_model = InceptionResNetV2(input_shape = (*IMG_SIZE, 3),
                               include_top = False, weights = None)

x = base_model.output
# add a global spatial average pooling layer
x = GlobalAveragePooling2D()(x)
# and a logistic layer
predictions = Dense(len(all_labels), activation="sigmoid")(x)
resnet_model = Model(inputs=base_model.input, outputs=predictions)
resnet_model.compile(optimizer='adam', loss="binary_crossentropy",
             metrics=[tensorflow.keras.metrics.binary_accuracy])
resnet_model.summary()
```

```
==================================================================================================
Total params: 54,358,254
Trainable params: 54,297,710
Non-trainable params: 60,544
```

Sử dụng lớp thư viện InceptionResNetV2 thì với định nghĩa mô hình ResNet sẽ có hơn 54,3 triệu tham số được tạo ra.

C. Xây dựng mô hình DenseNet phiên bản DenseNet121 dựa trên thư viện của TensorFlow như sau:

```
from tensorflow.keras.applications.densenet import DenseNet121
base_model = DenseNet121(input_shape = (*IMG_SIZE, 3),
                         include_top = False, weights = None)
x = base_model.output
# add a global spatial average pooling layer
x = GlobalAveragePooling2D()(x)
# and a logistic layer
predictions = Dense(len(all_labels), activation="sigmoid")(x)
densenet_model = Model(inputs=base_model.input, outputs=predictions)
densenet_model.compile(optimizer='adam', loss="binary_crossentropy",
            metrics=[tensorflow.keras.metrics.binary_accuracy])
densenet_model.summary()
```

```
==================================================================================================
Total params: 7,051,854
Trainable params: 6,968,206
Non-trainable params: 83,648
```

Sử dụng lớp thư viện DenseNet121 thì với định nghĩa mô hình DenseNet sẽ có hơn 7 triệu tham số được tạo ra.

Như vậy với kiến trúc của ba mô hình được xây dựng thì số lượng tham số cũng khác nhau, nhiều nhất vẫn là ResNet và thấp nhất là DenseNet chỉ có khoảng 7 triệu tham số.

3.1.2. Lấy dữ liệu mẫu từ kaggle và xử lý dữ liệu

```
!kaggle datasets download "nih-chest-xrays/sample"
```

Tải dữ liệu mẫu từ kaggle về Colab, sau đó đưa dữ liệu vào các biến và vùng nhớ để xử lý với ba dòng dữ liệu được xuất ra để kiểm tra:

```
data = pd.read_csv('./data/sample/sample_labels.csv')
data.columns = ['imgindex', 'label', 'followup', 'patientID',
                'age', 'gender', 'viewposition', 'width',
                'height', 'x', 'y']
data_image_paths = {os.path.basename(x): x for x in
                    glob(os.path.join('.', 'data/sample', 'images*', '*.png'))}
print('Scans found:', len(data_image_paths), ', Total Headers', data.shape[0])
data['imgindex'] = data['imgindex'].map(data_image_paths.get)
# dropping last empty column
data = data.drop(data.columns[-1], axis=1)
data.sample(3)
```

	imgindex	label	followup	patientID	age	gender	viewposition	width	height	x
3521	./data/sample/images/00017401_005.png	Effusion|Infiltration	5	17401	039Y	M	PA	2992	2991	0.143
4876	./data/sample/images/00025642_000.png	No Finding	0	25642	042Y	M	PA	2992	2991	0.143
830	./data/sample/images/00004344_042.png	No Finding	42	4344	048Y	F	AP	2500	2048	0.168

Định nghĩa 14 loại bệnh và dùng để phân các lớp cho việc huấn luyện mô hình:

```
all_labels = ['Atelectasis', 'Consolidation', 'Infiltration', 'Pneumothorax', 'Edema',
              'Emphysema', 'Fibrosis', 'Effusion', 'Pneumonia', 'Pleural_Thickening',
              'Cardiomegaly', 'Nodule', 'Mass', 'Hernia']
print(data[data.label.isin(all_labels)].label.value_counts())
```

Số hình ảnh XQN trên 14 loại bệnh được liệt kê ra từ dữ liệu mẫu được lấy về được thể hiện trên Hình 2.

```
Infiltration          503
Effusion              203
Atelectasis           192
Nodule                144
Pneumothorax          114
Mass                   99
Consolidation          72
Pleural_Thickening     65
Cardiomegaly           50
Emphysema              42
Edema                  41
Fibrosis               38
Pneumonia              14
Hernia                  5
```

Hình 2. Số hình ảnh XQN trên 14 loại bệnh

Do có thể có nhiều bệnh trên cùng một nhãn nên tách các bệnh này ra theo từng loại bệnh. Ví dụ: với hình ảnh XQN 00017401_005.png có hai bệnh trên cùng nhãn là Effusion|Infiltration nên sẽ tách ra hai nhãn bệnh Effusion và Infiltration và một bệnh có giá trị là 1, còn bệnh khác sẽ là 0.

Mã giả cho việc phân loại các nhãn bệnh như sau:

```
def create_categorical(dataframe):
    """
    Creates dummy variables through pandas and creates a 'target' column
    with a scalar vector of the dummy variables for later comparison of
    the convolutional neural networks results. Returns a dataframe.
    """
    # create dummy variables for the labels
    for i in all_labels:
        dataframe[i] = dataframe.label.map(lambda result: 1 if i in result else 0)
    # creating the target vector for cross-checking with the predictions later
    # since CNN gives out as a vector of 0s and 1s
    dataframe['target'] = dataframe.apply(lambda x: [x[all_labels].values],
                                          1).map(lambda x: x[0])
    return dataframe
```

```
# The create_categorical function creates dummy variables through pandas and
#creates a 'target' column by one hot encoding the labels for later comparison
#of the convolutional neural networks results.
data = create_categorical(data)
data.sample(5)
```

Như vậy sau khi phân loại nhãn bệnh cho 14 loại bệnh thì dữ liệu sẽ có dạng như sau:

imgindex	label
./data/sample/images/00009526_000.png	No Finding
./data/sample/images/00004144_002.png	No Finding
./data/sample/images/00014909_000.png	No Finding
./data/sample/images/00020623_000.png	No Finding
./data/sample/images/00010698_014.png	Infiltration

Atelectasis	Consolidation	Infiltration	Pneumothorax	Edema	Emphysema	Fibrosis	Effusion	Pneumonia	Pleural_Thickening	Cardiomegaly	Nodule	Mass	Hernia
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0

3.1.3. Chia dữ liệu mẫu cho huấn luyện, kiểm tra và kiểm thử

Dùng chính bộ dữ liệu hình ảnh để huấn luyện, làm dữ liệu xác nhận độ chính xác của các mô hình được tạo ra. Bộ dữ liệu kiểm tra được tạo một cách ngẫu nhiên từ bộ dữ liệu nguồn. Như vậy, tập huấn luyện có 3666 dữ liệu, tập kiểm tra có 408 dữ liệu, còn tập kiểm thử có 1358 dữ liệu.

```
from sklearn.model_selection import train_test_split

train_set, test_set = train_test_split(data, test_size=0.25,
                                       random_state=56,
                                       stratify=data.label)

train_set, valid_set = train_test_split(train_set, test_size=0.1,
                                        random_state=56,
                                        stratify=train_set.label)


print('training set values: ', train_set.shape[0])
print('validation set values: ', valid_set.shape[0])
print('testing set values: ', test_set.shape[0])
```

```
training set values:  3666
validation set values:  408
testing set values:  1358
```

Để tạo tính khách quan cho tập kiểm thử, loại bỏ các bệnh nhân có ID có trong tập kiểm thử mà tồn tại trong tập kiểm tra hoặc tập huấn luyện.

```
def check_for_leakage(df1, df2, patient_col):
    df1_patients_unique = set(df1[patient_col].values)
    df2_patients_unique = set(df2[patient_col].values)

    patients_in_both_groups = list(df1_patients_unique.intersection(df2_patients_unique))

    # leakage contains true if there is patient overlap, otherwise false.
    leakage = True if len(patients_in_both_groups)>0 else False  # boolean (true if there is at least 1 patient in both groups)

    return leakage
```

```
def drop_leakage(df, df_drop):
    # Extract patient id's for the df set
    ids_df = df.patientID.values
    # Extract patient id's for the df_drop set
    ids_drop = df_drop.patientID.values
    # Create a "set" datastructure of the df set id's to identify unique id's
    ids_df_set = set(ids_df)
    # Create a "set" datastructure of the df_drop set id's to identify unique id's
    ids_drop_set = set(ids_drop)
    # Identify patient overlap by looking at the intersection between the sets
    patient_overlap = list(ids_df_set.intersection(ids_drop_set))
    n_overlap = len(patient_overlap)
    df_drop_overlap_idxs = []
    for idx in range(n_overlap):
        df_drop_overlap_idxs.extend(df_drop.index[df_drop['patientID'] == patient_overlap[idx]].tolist())
    # Drop the overlapping rows from the df_drop set
    df_drop.drop(df_drop_overlap_idxs, inplace=True)
    return df_drop
```

```
test_set=drop_leakage(train_set, test_set)
test_set=drop_leakage(valid_set, test_set)
```

```
print("leakage between train and test: {}".format(check_for_leakage(train_set, test_set, 'patientID')))
print("leakage between valid and test: {}".format(check_for_leakage(valid_set, test_set, 'patientID')))
```

```
leakage between train and test: False
leakage between valid and test: False
```

Tóm lại, các bệnh nhân có ID có trong tập kiểm thử không còn tồn tại trong tập kiểm tra hoặc tập huấn luyện mà đồng nghĩa với việc có dữ liệu kiểm thử độc lập với dữ liệu của tập kiểm tra và tập huấn luyện. Tiếp theo, thực hiện tiền xử lý các dữ liệu hình ảnh có các tập huấn luyện, kiểm tra, kiểm thử để chuẩn bị cho quá trình tạo mô hình.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
IMG_SIZE = (128, 128)
# normalize images
idg = ImageDataGenerator()
```

Cấu hình có kích cỡ hình ảnh cho việc huấn luyện là 128×128 pixel.

```
train_generator = idg.flow_from_dataframe(
                        dataframe=train_set,
                        directory=None,
                        x_col ='imgindex',
                        y_col = 'label',
                        class_mode = 'categorical',
                        classes = all_labels,
                        target_size = IMG_SIZE,
                        color_mode = 'rgb',
                        batch_size = 32)
```

```
validation_generator = idg.flow_from_dataframe(
                        dataframe=valid_set,
                        directory=None,
                        x_col ='imgindex',
                        y_col = 'label',
                        class_mode = 'categorical',
                        classes = all_labels,
                        target_size = IMG_SIZE,
                        color_mode = 'rgb',
                        batch_size = 32)
```

```
x_test, y_test = next(idg.flow_from_dataframe(
                        dataframe=test_set,
                        directory=None,
                        x_col ='imgindex',
                        y_col = 'label',
                        class_mode = 'categorical',
                        classes = all_labels,
                        target_size = IMG_SIZE,
                        color_mode = 'rgb',
                        batch_size = 1024))
```

Với tập hình ảnh cho huấn luyện thì tập kiểm tra có batch_size = 32, tập kiểm thử có batch_size = 1024. Số lớp trên các tập đều là 14 classes và color_mode = ‘rgb’.

3.1.4. Huấn luyện dữ liệu và kết quả trên mỗi giải thuật

A. Huấn luyện được thực hiện với GoogleNet:

```
steps_epoch=int(train_generator.n/train_generator.batch_size)
vali_steps=int(validation_generator.n/validation_generator.batch_size)
history_googlenet = googlenet_model.fit_generator(train_generator,
                                    steps_per_epoch=steps_epoch,
                                    validation_data=validation_generator,
                                    validation_steps=vali_steps,
                                    epochs=30)
```

```
print('test binary accuracy = ',googlenet_model.evaluate(x_test, y_test, verbose=0)[1])
```

```
test binary accuracy =  0.928571343421936
```

Sau khi tạo được mô hình thì dựa trên tập kiểm thử độc lập, tiến hành kiểm thử và đạt được độ chính xác của GoogleNet là 92,857%.

B. Huấn luyện được thực hiện với ResNet

```
steps_epoch=int(train_generator.n/train_generator.batch_size)
vali_steps=int(validation_generator.n/validation_generator.batch_size)
history_resnet = resnet_model.fit_generator(train_generator,
                                    steps_per_epoch=steps_epoch,
                                    validation_data=validation_generator,
                                    validation_steps=vali_steps,
                                    epochs=30)
```

```
print('test binary accuracy = ',resnet_model.evaluate(x_test, y_test, verbose=0)[1])
```

```
test binary accuracy =  0.928571343421936
```

Sau khi tạo được mô hình thì dựa trên tập kiểm thử độc lập, tiến hành kiểm thử và đạt được độ chính xác của ResNet là 92,857%.

C. Huấn luyện được thực hiện với DenseNet

```
steps_epoch=int(train_generator.n/train_generator.batch_size)
vali_steps=int(validation_generator.n/validation_generator.batch_size)
history_densenet = densenet_model.fit_generator(train_generator,
                                    steps_per_epoch=steps_epoch,
                                    validation_data=validation_generator,
                                    validation_steps=vali_steps,
                                    epochs=30)
```

```
print('test binary accuracy = ',densenet_model.evaluate(x_test, y_test, verbose=0)[1])
```

```
test binary accuracy =  0.9273345470428467
```

Sau khi tạo được mô hình thì dựa trên tập kiểm thử độc lập, tiến hành kiểm thử và đạt được độ chính xác của DenseNet là 92,733%.

3.1.5. Đánh giá và so sánh các giải thuật

Đồ thị về miền hội tụ sử dụng Area Under the Curve (AUC) có thể cho biết một mô hình có hiệu quả hay không. Giá trị AUC là một số dương nhỏ hơn hoặc bằng 1. Giá trị này càng

lớn thì mô hình càng tốt. Tổng hợp lại ba thuật toán chạy trên 14 loại bệnh để so sánh chỉ số trung bình của cả ba thuật toán.

```
predictionList = [googlenet_preds, resnet_preds, densenet_preds]
weightList = [0.3, 0.3, 0.3]
def SimpleAverageEnsemble(predictionList):
    ensemble_preds = np.zeros(predictionList[0].shape)
    for pred in predictionList:
        ensemble_preds += pred

    ensemble_preds /= len(predictionList)

    return ensemble_preds
```

```
simple_ensemble_preds = SimpleAverageEnsemble(predictionList)
```

```
summaryDF = pd.DataFrame(
    {'Class':all_labels,
     'Googlenet': googlenetAUC,
     'Resnet': resnetAUC,
     'DenseNet': densenetAUC,
     'Avg Ensemble': simpleensembleAUC}
)
```

```
summaryDF.round(3)
```

Bảng 3. Giá trị AUC của ba thuật toán

	Class	Googlenet	Resnet	DenseNet	Avg Ensemble
0	Atelectasis	0.9814815	0.9692623	0.96857774	0.9639194
1	Consolidation	0.972973	0.9685658	0.96803904	0.96374226
2	Infiltration	0.9659091	0.96755725	0.9674493	0.9634183
3	Pneumothorax	0.9607843	0.96672827	0.9669465	0.96317273
4	Edema	0.96913576	0.9690635	0.96844184	0.964083
5	Emphysema	0.96511626	0.96792763	0.9623199	0.9637022
6	Fibrosis	0.9623656	0.9670418	0.9619565	0.9634116
7	Effusion	0.95789474	0.9656	0.96128035	0.96289766
8	Pneumonia	0.96875	0.96879333	0.96295905	0.96441686
9	Pleural_Thickening	0.9685534	0.9685408	0.96311194	0.96438915
10	Cardiomegaly	0.96961325	0.96867007	0.96354085	0.9648975
11	Nodule	0.96954316	0.96855736	0.96355355	0.9649256
12	Mass	0.96733665	0.9675245	0.9629726	0.9644646
13	Hernia	0.96774197	0.96783626	0.9632073	0.9645414
		0.967657049	**0.967976348**	**0.96459689**	

Cả ba thuật toán đều có AUC rất cao và khá sát nhau, đều bằng khoảng 0,96. Như vậy, chọn hai thuật toán có số lượng tham số thấp nhất nhưng có độ chính xác cao: GoogleNet và DenseNet để thực hiện huấn luyện trên bộ dữ liệu lớn để so sánh và chọn một thuật toán có mô hình tốt nhất để đưa vào chương trình chẩn đoán của bài báo này.

3.2. Giải thuật GoogleNet và DenseNet trên bộ dữ liệu lớn

3.2.1. Tải dữ liệu lớn từ Kaggle

```
!kaggle datasets download "nih-chest-xrays/data"
```

```
Downloading data.zip to /content
100% 42.0G/42.0G [08:33<00:00, 61.9MB/s]
```

Như vậy với bộ dữ liệu lớn data.zip được tải về có kích thước 42 Gbyte, sau khi giải nén sẽ có được 112120 hình ảnh XQN. Dựa trên bộ dữ liệu lớn này, tiến hành tạo mô hình như đối với bộ dữ liệu mẫu nhưng lần này chỉ áp dụng trên hai thuật toán: GoogleNet và DenseNet. Từ mô hình được tạo ra, tiến hành đánh giá độ chính xác trên từng giải thuật này.

3.2.2. Độ chính xác với giải thuật GoogleNet

Xây dựng đường dẫn lưu lại mô hình đã huấn luyện và thiết lập chỉ những mô hình có chỉ số huấn luyện tốt nhất mới được lưu lại.

```
from tensorflow.keras.callbacks import ModelCheckpoint, LearningRateScheduler,
EarlyStopping, ReduceLROnPlateau
weight_path="/content/gdrive/My Drive/Colab Notebooks/Model/Model_GoogleNet.hdf5"
checkpoint = ModelCheckpoint(weight_path, monitor='val_loss', verbose=1,
                             save_best_only=True, mode='min',
                             save_weights_only = False)

early = EarlyStopping(monitor="val_loss",
                      mode="min",
                      patience=50)
callbacks_list = [checkpoint, early]
```

Tiến hành huấn luyện lần 1 với 50 lần (step) chạy:

```
steps_epoch=int(train_gen.n/train_gen.batch_size)
vali_steps=int(valid_gen.n/valid_gen.batch_size)
print(steps_epoch, vali_steps)
history = model.fit_generator(train_gen,
                              steps_per_epoch=steps_epoch,
                              validation_data=valid_gen,
                              validation_steps=vali_steps,
                              callbacks = callbacks_list,
                              epochs=50)
```

Sau khi huấn luyện lần 1 xong, tiến hành kiểm tra chất lượng mô hình được huấn luyện:

```
y_pred = multi_disease_model.predict(test_X)
```

```
# look at how often the algorithm predicts certain diagnoses
for c_label, p_count, t_count in zip(all_labels,
                                     100*np.mean(y_pred,0),
                                     100*np.mean(test_Y,0)):
    print('%s: actual: %2.2f%%, predicted: %2.2f%%' % (c_label, t_count, p_count))
```

```
1 from sklearn.metrics import roc_auc_score
2 roc_auc_score(test_Y.astype(int), y_pred)
```

```
0.6489357976080449
```

Với độ chính xác trung bình trên 14 loại bệnh là 64,89% thì mô hình được huấn luyện bằng giải thuật GoogleNet có độ chính xác khá thấp. Do có thời gian huấn luyện ngắn nên độ hội tụ thấp. Do vậy, huấn luyện thêm lần 2 để xem độ hội tụ có dẫn đến độ chính xác cao hơn hay không?

Tiến hành huấn luyện lần 2 với 50 lần chạy:

```
history = model.fit_generator(train_gen,
                              steps_per_epoch=steps_epoch,
                              validation_data=valid_gen,
                              validation_steps=vali_steps,
                              callbacks = callbacks_list,
                              epochs=50)
```

```
1 from sklearn.metrics import roc_auc_score
2 roc_auc_score(test_Y.astype(int), y_pred)
```

```
0.6377276783540495
```

Sau khi thêm huấn luyện lần 2 thì độ chính xác trung bình trên 14 loại bệnh giảm từ 64,89% xuống còn 63,77%. Như vậy, mô hình được huấn luyện bằng giải thuật GoogleNet có độ chính xác khá thấp và thời gian huấn luyện ngắn lại cho kết quả tốt hơn.

3.2.3. Độ chính xác với giải thuật Densenet

Tương tự như giải thuật GoogleNet, huấn luyện 2 lần với số lần chạy là 50 cho giải thuật Densenet và sau 2 vòng chạy, đo độ hội tụ:

```
y_pred = multi_disease_model.predict(test_X)
```

```
# look at how often the algorithm predicts certain diagnoses
for c_label, p_count, t_count in zip(all_labels,
                                     100*np.mean(y_pred,0),
                                     100*np.mean(test_Y,0)):
    print('%s: actual: %2.2f%%, predicted: %2.2f%%' % (c_label, t_count, p_count))
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(test_Y.astype(int), y_pred)
```

```
0.7597105249338536
```

Với độ chính xác 75,97% thì mô hình được huấn luyện thuật giải Densenet có độ chính xác khá.

Dựa trên tổng hợp số liệu đo độ chính xác của các mô hình được huấn luyện bằng các giải thuật GoogleNet và Densenet, nhận thấy rằng mô hình được huấn luyện bằng giải thuật Densenet có độ chính xác tốt nhất. Do đó, bài báo sẽ khai thác mô hình này vào hình ảnh XQN thực tế. Để thuận tiện khi khai thác mô hình Densenet, mô hình được xuất ra dưới dạng file là Model_Densenet_New.hdf5.

4. CÁC KẾT QUẢ

4.1. Bệnh nhân có một loại bệnh

Sử dụng hình ảnh XQN từ nguồn khác để kiểm thử mô hình được đề xuất, cụ thể là một vài hình ảnh từ bản mẫu PadChest [15]. Ví dụ về một loại bệnh là Cardiomegaly trên Hình 3.

Tiến hành tải mô hình đã được tạo ra từ phần trước bằng thuật giải DenseNet:

```
from tensorflow.keras.models import load_model
densenet_model = load_model("/content/gdrive/My Drive/Colab Notebooks/Model/Use/Model_Densenet.hdf5")
```

Sau đó tiến hành chẩn đoán hình ảnh XQN bằng mô hình này:

```
img = image.img_to_array(imgs[idx])
img = np.expand_dims(img, axis = 0)
densenet_preds = densenet_model.predict(img)
display(imgs[idx])
print(img_names[idx])
createROC(all_labels, densenet_preds)
```

Hình 3. Hình ảnh XQN của bệnh Cardiomegaly được lấy từ PadChest

Kết quả nhận được với độ chính xác là 94% trên hình ảnh XQN của bệnh nhân Cardiomegaly (chứng tim to) cho thấy mô hình đã huấn luyện đạt được độ chính xác cao với bệnh nhân chỉ có tiền sử một loại bệnh trên hình ảnh XQN.

```
255433269247415893224655601475580025849_j5s1kc_cardiomegaly
Atelectasis :  0.0 %
Cardiomegaly :  94.0 %
Consolidation :  0.0 %
Edema :  0.0 %
Effusion :  0.0 %
Emphysema :  0.0 %
Fibrosis :  0.0 %
Infiltration :  0.0 %
Mass :  6.0 %
Nodule :  0.0 %
Pleural_Thickening :  3.0 %
Pneumonia :  0.0 %
Pneumothorax :  0.0 %
```

4.2. Bệnh nhân có nhiều loại bệnh

Sử dụng hình ảnh XQN từ nguồn mẫu PadChest để kiểm tra kết quả nhận dạng bệnh nhân có nhiều loại bệnh. Ví dụ về bệnh nhân có hai nhãn bệnh là Cardiomegaly và Effusion (tràn dịch màng phổi) như Hình 4.

Hình 4. Hình ảnh XQN của bệnh Cardiomegaly và Effusion được lấy từ PadChest

Kết quả nhận được là không đúng bệnh theo nhãn mà chẩn đoán ra bệnh Mass là 100%, Pneumonia là 52% trên hình ảnh XQN. Kết quả này cho thấy mô hình đã huấn luyện không đạt được độ chính xác đối với bệnh nhân có nhiều nhãn bệnh.

```
101103270798497222826083823719046670601_jw1fu2_Cardiomegaly_Effusion
Atelectasis :  27.0 %
Cardiomegaly :  0.0 %
Consolidation :  14.0 %
Edema :  0.0 %
Effusion :  0.0 %
Emphysema :  0.0 %
Fibrosis :  0.0 %
Infiltration :  0.0 %
Mass :  100.0 %
Nodule :  9.0 %
Pleural_Thickening :  22.0 %
Pneumonia :  52.0 %
Pneumothorax :  0.0 %
```

4.3. Bệnh nhân có loại bệnh khác

Bệnh nhân bị Covid-19 sẽ bị tổn thương phổi thì mô hình cho kết quả có khả năng bệnh Effusion và Pleural thickening (dày màng phổi chủ yếu do tràn dịch màng phổi) với độ chính xác lần lượt là 67% và 44%. Tổn thương phổi (vùng ngực) do Corona virus gây nên có thể gây tràn dịch màng phổi dẫn đến đông đặc phổi nên việc mô hình chẩn đoán bệnh nhân Covid-19 có thể bị Effusion là khá chính xác.

```
auntminnie-d-2020_01_28_23_51_6665_2020_01_28_Vietnam_coronavirus
Atelectasis :  0.0 %
Cardiomegaly :  0.0 %
Consolidation :  0.0 %
Edema :  0.0 %
Effusion :  67.0 %
Emphysema :  0.0 %
Fibrosis :  0.0 %
Infiltration :  0.0 %
Mass :  0.0 %
Nodule :  5.0 %
Pleural_Thickening :  44.0 %
Pneumonia :  0.0 %
Pneumothorax :  0.0 %
```

4.4. Ứng dụng mô hình huấn luyện vào chương trình thực tế

Để sử dụng mô hình của giải thuật Densenet nhằm tiến hành chẩn đoán bệnh trên các hình ảnh XQN thực tế, tiến hành thiết kế chương trình dựa trên nền tảng trình duyệt web với python flask.

4.4.1. Xây dựng chương trình phía Server (máy chủ)

Trên Server, cài đặt Python 3.8.8 và sau đó cài đặt flask bằng câu lệnh:

```
pip install flask
```

Tạo file app.py để chứa mã cụ thể của flask.

```python
from result import ResultModel
from flask import Flask, request, jsonify
import json
from werkzeug.utils import secure_filename
from flask.helpers import send_from_directory
from flask_cors import CORS, cross_origin
from flask_uploads import UploadSet, configure_uploads, IMAGES
# from scipy.misc import imsave, imread, imresize
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.densenet import preprocess_input, decode_predictions
# from tensorflow.keras.models import model_from_json
import tensorflow as tf
# for matrix math
import numpy as np
```

Tiến hành import các thư viện TensorFlow để sử dụng cho việc tải lại mô hình đã được chọn ở phần trước:

```python
model = tf.keras.models.load_model("model/Model_Densenet_New.hdf5")
```

Định nghĩa 14 lớp bệnh giống như đã được xây dựng ở phần huấn luyện mô hình:

```python
all_labels = ['Atelectasis', 'Cardiomegaly', 'Consolidation', 'Edema',
              'Effusion', 'Emphysema', 'Fibrosis', 'Hernia', 'Infiltration',
              'Mass', 'Nodule', 'Pleural Thickening', 'Pneumonia', 'Pneumothorax']
```

Khi người dùng (client) gửi lên một XQN thì server sẽ tiến hành chuyển đổi kích cở của hình sang 128×128 pixel để làm đầu vào cho mô hình tiến hành chẩn đoán:

```python
filename = photos.save(request.files['file'], "./static/img/")
img = image.load_img(filename, target_size=(128, 128))
x = image.img_to_array(img)
# make it the right size
x = np.expand_dims(x, axis=0)
```

```python
# perform the prediction
out = model.predict(x)
outlst = []
for (idx, c_label) in enumerate(all_labels):
    result = ResultModel(title=all_labels[idx], percentage=float("%.3f" % out[:, idx])*100)
    outlst.append(result)
```

Đầu ra của quá trình chẩn đoán bằng mô hình là dãy 14 số thực dao động từ 0 tới 1 với 1 là cao nhất và 0 là thấp nhất, ứng với 0% tới 100% khả năng được chẩn đoán, tương ứng với 14 lớp bệnh đã được định nghĩa ở trên. Như vậy trên máy chủ sau khi nhận được hình ảnh XQN sẽ trả về cho người dùng dãy số chẩn đoán khả năng bệnh trên 14 loại bệnh. Ngoài ra, để trả về cho người dùng thấy trực quan hơn về các loại bệnh thì bài báo này sử dụng thư viện Notejs v14.17.6 để có thể xây dựng gói json để trả về cho phía client:

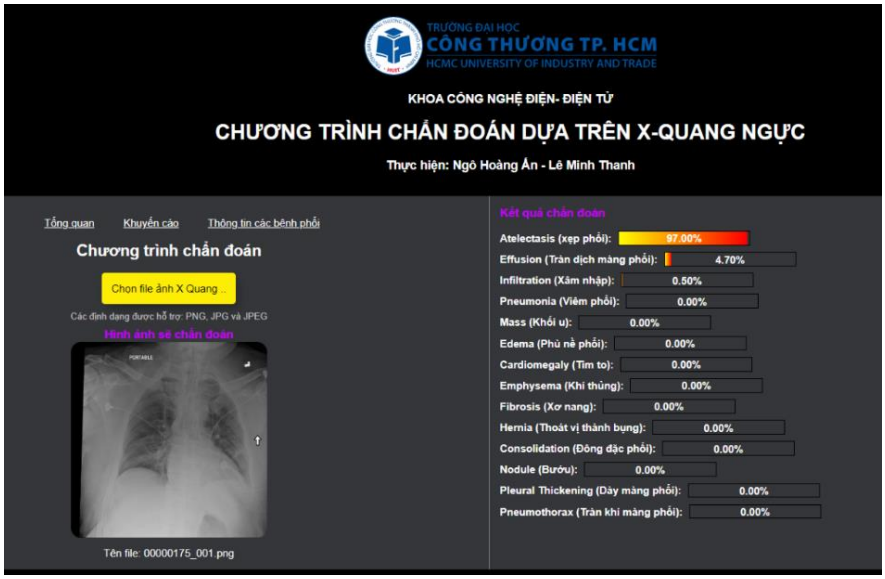
```python
# convert the response to a string
resp = json.dumps([ob.__dict__ for ob in outlst])
# resp.status_code = 201
return resp
```

4.4.2. Xây dựng chương trình phía Client

Để thuận tiện thao tác cho người dùng sử dụng trên nhiều hệ điều hành khác nhau thì chương trình ở phía client được viết trên nền tảng trình duyệt web như Hình 5.

Trang web sẽ có những phần sau:

(a) Phần tổng quan của chương trình: giới thiệu, mục đích chính và các tài liệu tham khảo.
(b) Phần khuyến cáo và thông tin chương trình: khuyến cáo người dùng rằng đây chỉ là chương trình trong bài báo này. Thông tin chương trình đã được thiết lập trên server.
(c) Phần chạy chính của chương trình: nơi để người dùng tải các hình ảnh XQN và người dùng có thể thấy được kết quả tự chẩn đoán 14 loại bệnh của chương trình.



Hình 5. Hình ảnh trang web phía người dùng – hình ảnh XQN đã chẩn đoán

5. KẾT LUẬN

Bài báo đã phân tích các đặc tính của hình ảnh XQN và 14 loại bệnh đặc trưng khi chẩn đoán dựa trên hình ảnh XQN. Dựa trên các loại bệnh này, bài báo nghiên cứu các mạng nơ-ron trong học sâu để có thể tiến hành huấn luyện cho máy học được các hình ảnh XQN nhằm có thể đưa ra mô hình chẩn đoán bệnh. Bài báo đã xây dựng được chương trình thực tế để tư vấn cho bác sĩ hoặc người dùng tự chẩn đoán dựa trên hình ảnh XQN của mình. Các kết quả thử nghiệm trên tập kiểm thử có sẵn cho thấy chương trình được đề xuất cho kết quả chẩn đoán với độ chính xác cao.

Lời cảm ơn: Nghiên cứu này do Trường Đại học Công nghiệp Thực phẩm Thành phố Hồ Chí Minh (nay là Trường Đại học Công Thương Thành phố Hồ Chí Minh) bảo trợ và cấp kinh phí theo Hợp đồng số 136/HĐ-DCT ngày 01 tháng 10 năm 2022.

TÀI LIỆU THAM KHẢO

1. Nguyễn Văn Thành - Thực hành X quang ngực, Nhà xuất bản Y học (2013).
2. Kermany, D., Zhang, K., and Goldbaum, M. - Labeled optical coherence tomography (OCT) and Chest X-Ray images for classification, Mendeley data **2** (2) (2018) https://doi.org/10.17632/RSCBJBR9SJ.2
3. Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., and Summers, R. M. - Chest x-ray8: Hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases, 2017 IEEE conference on

computer vision and pattern recognition USA (2017) 2097-2106 https://doi.org/10.1109/CVPR.2017.369.

4. Hopfield, A. J. J. - Neural networks and physical systems with emergent collective computational abilities, Proceedings of the national academy of sciences **79** (8) (1982) 2554-2558 https://doi.org/10.1073/pnas.79.8.2554.
5. Menshawy, A. - Deep learning by example: A hands-on guide to implementing advancedmachine learning algorithms and neural networks, Packt Publishing Ltd (2018).
6. Castiglioni, I., Ippolito, D., Interlenghi, M., Monti, C. B., Christian, et al. - Artificial intelligence applied on chest X-ray can aid in the diagnosis of COVID-19 infection: A first experience from Lombardy, Italy, European Radiology Experimental (2020) https://doi.org/10.1101/2020.04.08.20040907.
7. Ayan, E. and Ünver, H. M. - Diagnosis of Pneumonia from Chest X-Ray Images Using Deep Learning," 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT), (2019) 1-5 https://doi.org/10.1109/EBBT.2019.8741582.
8. Kanan C., and Cottrell, G. W. - Color-to-grayscale: does the method matter in image recognition?", PloS one **7** (1) (2012) https://doi.org/10.1371/journal.pone.0029740.
9. Gordienko Y., Gang P., Hui J., Zeng W., Kochura Y., Alienin O., Rokovyi O., Stirenko S. - Deep learning with lung segmentation and bone shadow exclusion techniques for chest X-ray analysis of lung cancer. In: Hu, Z., Petoukhov, S., Dychka, I., He, M. (eds) Advances in Computer Science for Engineering and Education. ICCSEEA 2018. Advances in Intelligent Systems and Computing **754**. Springer, Cham. https://doi.org/10.1007/978-3-319-91008-6_63
10. Behzadi-khormouji H., Rostami H., Salehi S., Derakhshande-Rishehri T., Masoumi M., Salemi S., Keshavarz A., Gholamrezanezhad A., Assadi M., Batouli A. - Deep learning, reusable and problem-based architectures for detection of consolidation on chest X-ray images, Computer Methods and Programs in Biomedicine **185** (2020) 105-162 https://doi.org/10.1016/j.cmpb.2019.105162.
11. Cohen, J. P., Bertin, P., Frappier, V. - Chester: A web delivered locally computed chest X-ray disease prediction system, arXiv (2020) https://doi.org/10.48550/arXiv.1901.11210.
12. Lechner, A. J., Matuschak, G. M., Brink, D. S. - Respiratory: An integrated approach to disease, McGraw-Hill Education LLC (2012).
13. National Institutes of Health - Chest Radiograph Dataset, 2020. Nguồn truy cập từ https://nihcc.app.box.com/v/ChestXray-NIHCC.
14. National Institutes of Health - Random Sample of NIH Chest X-ray Dataset, 2020. Nguồn truy cập từ: https://www.kaggle.com/datasets/nih-chest-xrays/.
15. PadChest - Báo cáo bộ dữ liệu dán nhãn X-quang tại Bệnh viện San Juan, Tây Ban Nha (2009-2017). Nguồn truy cập từ: https://bimcv.cipf.es/bimcv-projects/padchest/.

ABSTRACT

DESIGN A CHEST X-RAY-BASED DIAGNOSIS PROGRAM

Ngo Hoang An[1]*, Le Minh Thanh[1], Le Trach Dinh[2]

[1]*Ho Chi Minh City University of Industry and Trade*

[2]*Splus Software Company*

Email: annh@huit.edu.vn*

The condition, components, and adjacent structures of the chest can be assessed via Chest X-Ray (CXR) image–based diagnosis. However, relying on CXR images to observe and make accurate diagnoses will take a lot of time, even for an experienced doctor. In addition, in order to minimize errors in disease diagnosis due to human factors such as doctors' fatigue when looking at too many CXR images in a day, creating a smart tool that helps doctors reduce the time for CXR image inspection and diagnosis with high accuracy will save a lot of cost and time. This article will study the advanced algorithms of neural networks in deep learning and apply these algorithms to a big data set of CXR images to design deep learning models on several common diseases. The proposed models help users to diagnose their own CXR images or doctors can have a quick and mass diagnosis channel from CXR images with high accuracy through a diagnosis program based on Web Explorer.

Keywords: Chest X-Ray, Artificial Intelligence, disease diagnosis.