

**TRƯỜNG ĐẠI HỌC TRÀ VINH**  
**KHOA KỸ THUẬT VÀ CÔNG NGHỆ**



**ISO 9001:2015**

**LÊ THANH TRUYỀN**

**PHÁT HIỆN BẤT THƯỜNG TRONG ẢNH Y KHOA**

**ĐỒ ÁN TỐT NGHIỆP**  
**NGÀNH CÔNG NGHỆ THÔNG TIN**

**TRÀ VINH, NĂM 2024**

TRƯỜNG ĐẠI HỌC TRÀ VINH  
KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**PHÁT HIỆN BẤT THƯỜNG TRONG ẢNH Y KHOA**

**ĐỒ ÁN TỐT NGHIỆP  
NGÀNH CÔNG NGHỆ THÔNG TIN**

Sinh viên: **Lê Thanh Truyền**  
Lớp: **DA20TTB**  
MSSV: **110120163**  
GVHD: **Nguyễn Mộng Hiền**

**TRÀ VINH, NĂM 2024**

## LỜI MỞ ĐẦU

Bước vào thế kỷ 21, công nghệ thông tin đã trở thành trụ cột trong hầu hết mọi lĩnh vực trọng điểm như giao thông và vận tải, sản xuất công nghiệp, dịch vụ, tài chính, giáo dục,... và trở thành một phần không thể thiếu trong cuộc sống hàng ngày, đem lại những thành tựu đáng kể và cải thiện lớn cho đời sống xã hội. Trong đó, sự phát triển vượt bậc của trí tuệ nhân tạo và học máy đã thúc đẩy sự tiến bộ đáng kể trong mọi lĩnh vực, từ sản xuất công nghiệp đến dịch vụ tài chính. Đặc biệt, trong lĩnh vực y học, y học ngày càng phát triển nhờ vào sự tiến bộ vượt bậc của công nghệ. Nổi bật là sức mạnh của trí tuệ nhân tạo đang được khai thác mạnh mẽ để nâng cao khả năng chẩn đoán và điều trị bệnh. Trí tuệ nhân tạo và học sâu đang đóng vai trò quan trọng trong việc phát hiện các bất thường trong ảnh y khoa, giúp nâng cao hiệu quả chẩn đoán, điều trị và giảm thiểu rủi ro cho bệnh nhân. Khả năng của trí tuệ nhân tạo trong việc xử lý và phân tích dữ liệu lớn, kết hợp với khả năng học và tự điều chỉnh, đã tạo ra những tiềm năng mới trong việc cải thiện chăm sóc sức khỏe và cứu sống bệnh nhân.

Khóa luận này tập trung nghiên cứu và ứng dụng các mô hình học sâu vào việc phát hiện bất thường trong ảnh y khoa. Nội dung của khóa luận gồm các chương: Chương 1 giới thiệu về các khái niệm cơ bản và bối cảnh nghiên cứu. Chương 2 giới thiệu một số mô hình học sâu phổ biến. Chương 3 mô tả việc hiện thực hoá nghiên cứu, áp dụng các mô hình vào thực nghiệm trên tập dữ liệu cùng với các phương pháp đánh giá và huấn luyện chúng. Chương 4 là phân kết luận, đánh giá công việc đã thực hiện trong chương 3. Cuối cùng, chương 5 kết luận những gì đã đạt được và đề xuất những hướng phát triển tiếp theo.

Em hy vọng rằng khóa luận này sẽ đóng góp một phần nhỏ vào việc nâng cao hiệu quả chẩn đoán y khoa bằng cách ứng dụng các công nghệ tiên tiến, đồng thời mở ra những hướng nghiên cứu mới trong lĩnh vực này. Bằng cách tận dụng sức mạnh của trí tuệ nhân tạo và học máy, con người có thể hy vọng vào một tương lai với các phương pháp chẩn đoán và điều trị tiên tiến hơn, mang lại lợi ích to lớn cho xã hội.

## LỜI CẢM ƠN

Em xin chân thành cảm ơn các thầy cô trong Bộ môn Công nghệ Thông tin, Khoa Kỹ thuật và Công nghệ Trường Đại học Trà Vinh, đã truyền đạt kiến thức và hỗ trợ em trong suốt quá trình học tập và nghiên cứu thực hiện khoá luận này.

Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến Thầy Nguyễn Mộng Hiền, người giảng viên hướng dẫn kiên nhẫn và tận tình đã hỗ trợ em hoàn thành khoá luận. Những góp ý quý báu và sự hỗ trợ nhiệt tình của thầy đã giúp em vượt qua những khó khăn và hoàn thiện công việc nghiên cứu của mình.

Em cũng xin cảm ơn các anh chị và bạn bè đã luôn bên cạnh, động viên và giúp đỡ em trong suốt quá trình học tập và thực hiện khoá luận. Những lời động viên và sự hỗ trợ từ các bạn đã giúp em có thêm động lực để hoàn thành tốt.

Cuối cùng, em xin gửi lời cảm ơn đến gia đình, là nguồn động viên to lớn và là chỗ dựa vững chắc cho em. Sự ủng hộ và tình yêu thương của gia đình đã giúp em vượt qua mọi thử thách và đạt được kết quả như ngày hôm nay.

Em xin trân trọng cảm ơn!

*Trà Vinh, ngày ..... tháng ... năm 2024*

Sinh viên thực hiện

**Lê Thanh Truyền**

## This image shows a single sheet of white paper with horizontal blue ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

**Giảng viên hướng dẫn**  
(ký và ghi rõ họ tên)

**BẢN NHẬN XÉT ĐỒ ÁN, KHÓA LUẬN TỐT NGHIỆP**  
(*Của giảng viên hướng dẫn*)

Họ và tên sinh viên: ..... MSSV: .....

Ngành: ..... Khóa: .....

Tên đề tài: .....

.....

Họ và tên Giáo viên hướng dẫn: .....

Chức danh: ..... Học vị: .....

**NHẬN XÉT**

1. Nội dung đề tài:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

2. Ưu điểm:

.....

.....

.....

3. Nhược điểm:

.....

.....

.....

.....

4. Điểm mới đề tài:

.....

.....

.....

.....

.....

5. Giá trị thực trên đề tài:

.....

.....

.....

.....

.....

.....

.....

7. Đề nghị sửa chữa bổ sung:

.....

.....

.....

.....

.....

.....

.....

8. Đánh giá:

.....

.....

.....

.....

Trà Vinh, ngày ..... tháng ..... năm 2024  
Giảng viên hướng dẫn  
(Ký & ghi rõ họ tên)

[illegible]

**Giảng viên chấm**  
(ký và ghi rõ họ tên)



**BẢN NHẬN XÉT ĐỒ ÁN, KHÓA LUẬN TỐT NGHIỆP**  
(Của cán bộ chấm đồ án, khóa luận)

Họ và tên người nhận xét: .....

Chức danh: ..... Học vị: .....

Chuyên ngành: .....

Cơ quan công tác: .....

Tên sinh viên: .....

Tên đề tài đồ án, khóa luận tốt nghiệp: .....

.....

.....

**I. Ý KIẾN NHẬN XÉT**

**1. Nội dung:**

.....

.....

.....

.....

.....

.....

.....

.....

.....

**2. Điểm mới các kết quả của đồ án, khóa luận:**

.....

.....

.....

**3. Ứng dụng thực tế:**

.....

.....

.....

.....

.....

.....

(Các câu hỏi của giáo viên phản biện)

### III. KẾT LUẬN

(Ký & ghi rõ họ tên)

## MỤC LỤC

<b>CHƯƠNG 1. ĐẶT VẤN ĐỀ .....</b>	<b>1</b>
1.1. Lý do chọn đề tài.....	1
1.2. Mục tiêu .....	1
1.3. Nội dung.....	1
1.4. Đối tượng và phạm vi nghiên cứu .....	2
1.5. Phương pháp nghiên cứu .....	2
<b>CHƯƠNG 2. CƠ SỞ LÝ THUYẾT .....</b>	<b>4</b>
2.1. Sơ lược về học sâu .....	4
2.1.1. Định nghĩa.....	4
2.1.2. Ứng dụng .....	4
2.1.3. Cấu tạo chung của một mô hình học sâu .....	4
2.1.4. Các bước hoạt động của một mô hình học sâu .....	5
2.2. Các mô hình học sâu .....	6
2.2.1. Mạng CNN.....	6
2.2.2. Mạng RNN.....	7
2.2.3. Mạng Autoencoder .....	9
2.2.4. Mạng GAN.....	10
2.2.5. Mạng ResNet .....	11
2.2.6. Mạng DenseNet .....	12
2.3. Quy trình huấn luyện và đánh giá một mô hình học sâu .....	13
<b>CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU .....</b>	<b>14</b>
3.1. Mô tả bài toán .....	14
3.2. Những kết quả đã đạt được trước đó .....	14
3.3. Đề xuất cách giải quyết bài toán .....	15
3.3.1. Kiến trúc của CNN .....	15
3.3.2. Kiến trúc của ResNet .....	17
3.3.3. Kiến trúc của DenseNet .....	18
3.4. Giới thiệu tập dữ liệu .....	19
3.5. Môi trường cài đặt.....	20
<b>CHƯƠNG 4. KẾT QUẢ NGHIÊN CỨU .....</b>	<b>22</b>
4.1. Chuẩn bị thư viện và dữ liệu.....	22
4.2. Xây dựng các mô hình .....	25
4.3. Huấn luyện các mô hình .....	26
4.4. Đánh giá các mô hình .....	28
4.5. Xây dựng giao diện phát hiện bất thường trong ảnh .....	31

4.6. Nhận xét.....	33
4.7. Hướng dẫn sử dụng ứng dụng.....	33
<b>CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....</b>	<b>37</b>
5.1. Kết luận.....	37
5.2. Hướng phát triển .....	37
<b>DANH MỤC TÀI LIỆU THAM KHẢO.....</b>	<b>38</b>

## DANH MỤC CÁC BẢNG, SƠ ĐỒ, HÌNH

Hình 1. Cấu tạo chung một mô hình học sâu .....	5
Hình 2. Cấu tạo chung của mạng CNN.....	7
Hình 3. Cấu tạo chung của một mạng RNN.....	8
Hình 4. Cấu tạo của ba mạng RNN, LSTM và GRU.....	9
Hình 5. Cấu tạo của mạng Autoencoder.....	10
Hình 6. Cấu tạo của mạng GAN.....	11
Hình 7. Cấu tạo của một biến thể ResNet – ResNet18 .....	12
Hình 8. Cấu tạo của mạng DenseNet.....	13
Hình 9. Sơ đồ khối kiến trúc CNN được đề xuất .....	16
Hình 10. Sơ đồ khối kiến trúc ResNet được đề xuất.....	18
Hình 11. Sơ đồ khối kiến trúc DenseNet được đề xuất.....	19
Hình 12. Cây thư mục của tập dữ liệu .....	20
Hình 13. Logo của Kaggle .....	21
Hình 14. Các thư viện cho việc xây dựng các mô hình.....	22
Hình 15. Các thư viện dùng cho việc sắp xếp dữ liệu .....	22
Hình 16. Các biến dựa trên cấu trúc của tập dữ liệu .....	23
Hình 17. Số lượng hình ảnh trong từng thư mục.....	23
Hình 18. Khai báo vị trí của thư mục mới cho việc sắp xếp dữ liệu .....	24
Hình 19. Thực hiện việc xáo trộn và phân chia hình ảnh theo tỷ lệ.....	24
Hình 20. Sao chép lần lượt hình ảnh vào các thư mục mới .....	24
Hình 21. Kiểm tra lại số lượng hình ảnh sau khi chia.....	25
Hình 22. Mã nguồn kiến trúc mô hình CNN .....	25
Hình 23. Mã nguồn kiến trúc mô hình ResNet .....	25
Hình 24. Mã nguồn kiến trúc mô hình DenseNet .....	26
Hình 25. Biên dịch và huấn luyện mô hình CNN.....	26
Hình 26. Biên dịch và huấn luyện mô hình ResNet .....	26
Hình 27. Biên dịch và huấn luyện mô hình DenseNet .....	27
Hình 28. Minh họa việc huấn luyện mô hình CNN.....	27
Hình 29. Minh họa việc huấn luyện mô hình ResNet .....	28
Hình 30. Minh họa việc huấn luyện mô hình DenseNet .....	28
Hình 31. Chỉ số độ chính xác trên tập kiểm thử của mô hình CNN .....	28

<b>Hình 32. Chỉ số độ chính xác trên tập kiểm thử của mô hình ResNet.....</b>	<b>29</b>
<b>Hình 33. Chỉ số độ chính xác trên tập kiểm thử của mô hình DenseNet.....</b>	<b>29</b>
<b>Hình 34. Hàm vẽ biểu đồ thể hiện độ chính xác khi huấn luyện của các mô hình ....</b>	<b>29</b>
<b>Hình 35. Biểu đồ thể hiện độ chính xác khi huấn luyện mô hình CNN .....</b>	<b>30</b>
<b>Hình 36. Biểu đồ thể hiện độ chính xác khi huấn luyện mô hình ResNet .....</b>	<b>30</b>
<b>Hình 37. Biểu đồ thể hiện độ chính xác khi huấn luyện mô hình DenseNet .....</b>	<b>31</b>
<b>Hình 38. Các thư mục của hệ thống .....</b>	<b>31</b>
<b>Hình 39. Giao diện chính.....</b>	<b>32</b>
<b>Hình 40. Kết quả hiển thị ảnh bình thường .....</b>	<b>32</b>
<b>Hình 41. Kết quả hiển thị ảnh bất thường.....</b>	<b>33</b>
<b>Hình 42. Vị trí tệp Python cần tải.....</b>	<b>34</b>
<b>Hình 43. Cấu trúc thư mục của ứng dụng.....</b>	<b>34</b>
<b>Hình 44. Kích hoạt môi trường ảo.....</b>	<b>34</b>
<b>Hình 45. Thông báo chạy giao diện ứng dụng.....</b>	<b>35</b>
<b>Hình 46. Giao diện ứng dụng trên trình duyệt.....</b>	<b>35</b>
<b>Hình 47. Chọn tệp hình ảnh cần thử.....</b>	<b>35</b>
<b>Hình 48. Giao diện sau khi chọn.....</b>	<b>36</b>
<b>Hình 49. Kết quả nhận dạng “bình thường” .....</b>	<b>36</b>

# DANH MỤC TỪ VIẾT TẮT

(Sắp xếp danh mục từ viết tắt theo thứ tự alphabet của từ viết tắt)

Từ viết tắt	Ý nghĩa
CNN	Mạng neural tích chập
DenseNet	Mạng kết nối dày đặc
LSTM	Mạng bộ nhớ ngắn hạn dài
GRU	Mạng neural hồi tiếp với nút có cổng
GAN	Mạng sinh đối nghịch
ReLU	Hàm tuyến tính chỉnh lưu
ResNet	Mạng phần dư
RNN	Mạng neural hồi quy

## **CHƯƠNG 1. ĐẶT VẤN ĐỀ**

### **1.1. Lý do chọn đề tài**

Trong những năm gần đây, công nghệ thông tin đã phát triển với tốc độ chóng mặt, đặc biệt là trong lĩnh vực y khoa, khi việc ứng dụng công nghệ thông tin vào quá trình chẩn đoán và điều trị bệnh đã trở nên phổ biến. Ảnh y khoa là một công cụ không thể thiếu trong quá trình chẩn đoán từ X-quang đến siêu âm. Đối với các bác sĩ, phân tích và diễn giải ảnh y khoa đòi hỏi sự chính xác và kinh nghiệm lâm sàng cao. Điều này tạo ra áp lực lớn cho các bác sĩ, đặc biệt là trong bối cảnh số lượng ảnh y khoa ngày càng tăng và cần được xử lý nhanh chóng. Công việc này yêu cầu một phương pháp hiệu quả để tăng cường khả năng chẩn đoán và điều trị, giúp bác sĩ nhanh chóng xác định vấn đề sức khỏe của bệnh nhân.

Với những lý do trên, em đã chọn đề tài "Phát hiện bất thường trong ảnh y khoa" để nghiên cứu nhằm hỗ trợ bác sĩ trong việc phân tích ảnh y khoa, giảm bớt gánh nặng công việc và tăng độ chính xác trong chẩn đoán. Bằng cách sử dụng các mô hình học sâu, chúng có khả năng tự động phát hiện các dấu hiệu bất thường, từ đó cung cấp thông tin hữu ích cho quá trình đánh giá và quyết định lâm sàng của bác sĩ.

Ngoài ra, việc phát triển các mô hình này cũng góp phần nâng cao chất lượng dịch vụ y tế và mở ra hướng nghiên cứu mới trong việc ứng dụng công nghệ thông tin vào y khoa. Em hy vọng rằng, đề tài này sẽ góp phần vào sự phát triển của ngành y tế và công nghệ thông tin. Đây chính là động lực và mục tiêu quan trọng của em trong quá trình thực hiện khoá luận tốt nghiệp này.

### **1.2. Mục tiêu**

Mục tiêu của khoá luận này là đề xuất và phát triển phương pháp hiệu quả để phát hiện bất thường trong ảnh y khoa. Cụ thể, khoá luận sẽ tập trung vào các mục tiêu sau:

- Giới thiệu về lĩnh vực phát hiện bất thường trên ảnh y khoa và các phương pháp hiện đang có, từ đó xác định những hạn chế và thách thức cần được giải quyết.
- Tìm hiểu về các mô hình học sâu, xây dựng các mô hình sao cho chúng có khả năng tự động phát hiện các dấu hiệu bất thường từ dữ liệu ảnh y khoa với độ chính xác cao và thời gian xử lý nhanh chóng.
- Đánh giá hiệu suất của các mô hình thông qua các thử nghiệm trên bộ dữ liệu thực tế, so sánh và đưa ra các cải tiến cần thiết.

### **1.3. Nội dung**

Giới thiệu về học sâu, trình bày cách hoạt động của các mô hình học sâu phổ biến.



Mô tả về bài toán, giới thiệu một số kết quả đã đạt được trong việc giải quyết bài toán và mô tả về các phương pháp hiện đã và đang được sử dụng với những kết quả đã đạt được và những hạn chế của chúng.

Đề xuất mô hình cho quá trình phát hiện bất thường, cùng với việc thu thập và tiền xử lý dữ liệu, xây dựng mô hình, huấn luyện mô hình, và đánh giá hiệu suất của mô hình.

#### **1.4. Đối tượng và phạm vi nghiên cứu**

Đối tượng nghiên cứu của khoá luận này sẽ tập trung vào các hình ảnh y khoa bình thường và bất thường, có thể là các bức ảnh chụp từ thiết bị, hoặc có thể là ảnh chụp từ một nhóm bệnh nhân với điều kiện sức khỏe nhất định được sử dụng để phát hiện các bất thường, bệnh lý, hoặc các dấu hiệu sớm của bệnh tật.

Phạm vi của nghiên cứu sẽ bao gồm các phương pháp và mô hình học sâu được áp dụng để phát hiện bất thường trong ảnh y khoa. Mô hình sẽ được thử nghiệm và đánh giá trên tập dữ liệu chứa các hình ảnh y khoa đa dạng và phong phú. Ngoài ra, phạm vi cũng xem xét đến các yếu tố như độ phức tạp của mô hình, thời gian xử lý, việc phân tích và xử lý số liệu độ chính xác, hiệu quả của các mô hình.

#### **1.5. Phương pháp nghiên cứu**

- Phương pháp nghiên cứu lý thuyết:

Đọc và tổng hợp tài liệu liên quan: Tìm kiếm và nghiên cứu các bài báo khoa học, sách, và tài liệu chuyên ngành liên quan đến phát hiện bất thường trong ảnh y khoa và các mô hình học sâu.

Phân tích các phương pháp hiện có: Đánh giá các phương pháp và kỹ thuật hiện có trong lĩnh vực học sâu và xử lý ảnh y khoa để xác định các ưu điểm và hạn chế của chúng.

Xây dựng cơ sở lý thuyết: Phát triển nền tảng lý thuyết dựa trên kiến thức thu thập được, từ đó định hướng cho phương pháp nghiên cứu thực nghiệm.

- Phương pháp nghiên cứu thực nghiệm:

Thu thập các bộ dữ liệu ảnh y khoa từ các nguồn đáng tin cậy

Tiền xử lý dữ liệu để chuẩn hóa và làm sạch dữ liệu, bao gồm loại bỏ nhiễu, cân bằng dữ liệu, và tăng cường dữ liệu.

Nêu ra và xây dựng các mô hình học sâu.

Huấn luyện mô hình trên các bộ dữ liệu đã được tiền xử lý.

Đánh giá hiệu suất của mô hình bằng các độ đo.

Thực hiện các thí nghiệm so sánh giữa các mô hình để tìm ra mô hình hiệu quả nhất.

Phân tích kết quả thực nghiệm để rút ra các kết luận về hiệu suất và tính khả thi của các mô hình, và đề xuất các cải tiến hoặc hướng nghiên cứu tiếp theo dựa trên kết quả đó.

## **CHƯƠNG 2. CƠ SỞ LÝ THUYẾT**

### **2.1. Sơ lược về học sâu**

#### **2.1.1. Định nghĩa**

Học sâu là một nhánh của học máy, sử dụng các mô hình mạng neural sâu để xử lý dữ liệu phức tạp và thực hiện các nhiệm vụ như phân loại, dự đoán và phát hiện.

Học sâu tập trung vào việc xây dựng và huấn luyện các mạng neural với nhiều lớp ẩn. Khác với các phương pháp truyền thống, học sâu cho phép mô hình hóa các bài toán phức tạp và tìm ra mối liên hệ không tuyến tính giữa các đặc trưng đó bằng cách tự động học các đặc trưng của dữ liệu đầu vào [1].

Học sâu sử dụng các thuật toán được thiết kế để dạy máy tính xử lý dữ liệu tương tự như cách bộ não con người hoạt động. Mô hình học sâu có khả năng nhận diện và phân tích các đối tượng phức tạp trong hình ảnh, văn bản, âm thanh,... từ đó trích xuất ra các thông tin hữu ích và đưa ra dự đoán chính xác [2].

#### **2.1.2. Ứng dụng**

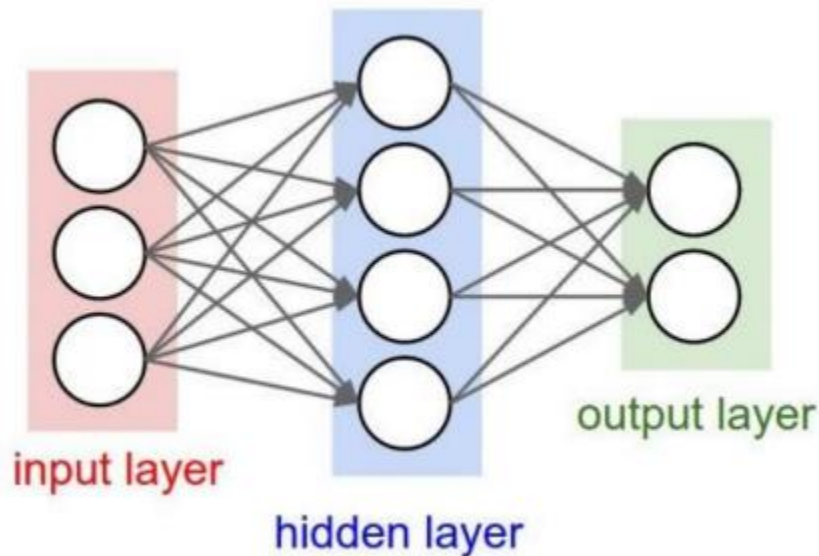
Học sâu được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau như: phân loại, nhận dạng, dự đoán và dự báo, xử lý ngôn ngữ tự nhiên,... [3]. Sự phát triển của các thuật toán học sâu đã thúc đẩy tiến bộ trong việc xử lý dữ liệu không cấu trúc, mở ra khả năng giải quyết các vấn đề phức tạp mà trước đây được coi là không thể. Với khả năng tự học từ dữ liệu lớn và phức tạp, học sâu đã chứng minh tiềm năng to lớn trong việc mô phỏng các quá trình nhận thức của con người và thậm chí còn vượt qua hiệu suất của con người trong một số lĩnh vực cụ thể.

Tuy nhiên, học sâu cũng đối mặt với nhiều thách thức lớn mà một trong những số đó là yêu cầu về dữ liệu lớn và sự phụ thuộc vào các tài nguyên tính toán mạnh mẽ. Mạng neural sâu cần một lượng lớn dữ liệu huấn luyện để có thể học được các đặc trưng phức tạp và tránh hiện tượng quá khớp khi mô hình chỉ hoạt động tốt trên dữ liệu huấn luyện nhưng không tổng quát hóa được trên dữ liệu mới. Ngoài ra, việc huấn luyện các mô hình học sâu thường đòi hỏi phần cứng chuyên biệt, như GPU hoặc TPU, để xử lý các phép tính song song cần thiết cho việc cập nhật trọng số mạng nhanh chóng [4].

#### **2.1.3. Cấu tạo chung của một mô hình học sâu**

Mô hình học sâu là một loại mô hình máy học có khả năng tự động học và trích xuất đặc trưng từ dữ liệu đầu vào thông qua việc sử dụng nhiều lớp xử lý tương tác với nhau, mỗi lớp thực hiện một phần nhất định của quá trình học. Thông thường các lớp này bao

gồm lớp đầu vào, lớp ẩn, và lớp đầu ra [5]. Lớp đầu vào là nơi mà dữ liệu được cung cấp cho mô hình, qua đó mô hình có thể học được các đặc trưng từ dữ liệu. Lớp ẩn, thường có nhiều lớp nhỏ hơn, là nơi xử lý thông tin, thông qua các hàm kích hoạt không tuyến tính, giúp mô hình có khả năng học được các đặc trưng phức tạp và mối quan hệ giữa chúng. Lớp đầu ra là nơi mô hình đưa ra dự đoán hoặc kết quả dựa trên những gì đã học được.



**Hình 1. Cấu tạo chung một mô hình học sâu**

Mỗi lớp trong mô hình học sâu chứa một tập hợp các đơn vị xử lý gọi là neural, mỗi neural có khả năng tính toán và truyền thông tin từ lớp này sang lớp khác thông qua các kết nối có trọng số.

Các mô hình này thường được huấn luyện thông qua một quá trình có giám sát, nửa giám sát hoặc không giám sát, tùy thuộc vào loại dữ liệu và nhiệm vụ cụ thể. Trong quá trình học, mô hình sẽ điều chỉnh các trọng số của mình để giảm thiểu sai số giữa đầu ra dự đoán và đầu ra thực tế [5].

Một trong những đặc điểm quan trọng của học sâu là khả năng tự học biểu diễn đặc trưng, loại bỏ nhu cầu phải thiết kế thủ công các đặc trưng vốn thường đòi hỏi kiến thức chuyên môn sâu và thời gian dài. Thay vào đó, mô hình học sâu có thể tự động phát hiện các đặc trưng quan trọng từ dữ liệu thông qua khả năng tự học và điều chỉnh các trọng số của các kết nối, làm mô hình trở nên mạnh mẽ và linh hoạt [5].

#### **2.1.4. Các bước hoạt động của một mô hình học sâu**

Mô hình học sâu hoạt động theo các bước cơ bản như sau [6, 7]:

- Nhận dữ liệu đầu vào: Mô hình nhận dữ liệu đầu vào từ tập huấn luyện, bao gồm các điểm dữ liệu và nhãn tương ứng (nếu có).

- Chuyển tiếp: Dữ liệu đầu vào được chuyển tiếp qua các lớp của mô hình từ lớp đầu vào đến lớp đầu ra. Trong quá trình này, các phép tính toán tương ứng với các trọng số của mô hình được thực hiện để tạo ra đầu ra dự đoán.
- Tính toán lỗi: Đầu ra dự đoán của mô hình được so sánh với nhãn thực tế của dữ liệu huấn luyện để tính toán lỗi. Lỗi thường được đo lường bằng một hàm mất mát phù hợp với loại bài toán, ví dụ như Cross-Entropy Loss cho bài toán phân loại.
- Lan truyền ngược: Sau khi tính toán lỗi, mô hình sử dụng thuật toán lan truyền ngược để cập nhật các trọng số của mạng neural. Quá trình này lan truyền lỗi từ lớp đầu ra trở về lớp đầu vào, điều chỉnh các trọng số sao cho lỗi giảm dần qua mỗi lần cập nhật, giúp mạng tự điều chỉnh thông qua dữ liệu và cải thiện dự đoán của mình theo thời gian.
- Cập nhật trọng số: Các trọng số của mô hình được cập nhật dựa trên đạo hàm của hàm mất mát theo các trọng số và tốc độ học giúp điều chỉnh các tham số sao cho đầu ra dự đoán gần với nhãn thực tế nhất.
- Lặp lại quá trình: Các quá trình chuyển tiếp, tính toán lỗi, lan truyền ngược và cập nhật trọng số được lặp lại cho đến khi mô hình đạt được hiệu suất mong muốn hoặc khi điều kiện dừng được đáp ứng.

## 2.2. Các mô hình học sâu

Có nhiều mô hình học sâu phổ biến được sử dụng như: CNN, RNN, autoencoder,.... Mỗi mô hình có những đặc điểm riêng biệt và ứng dụng phù hợp cho từng loại dữ liệu và nhiệm vụ cụ thể.

Dưới đây trong từng mục, chúng ta sẽ tìm hiểu chi tiết về các mô hình học sâu này.

### 2.2.1. Mạng CNN

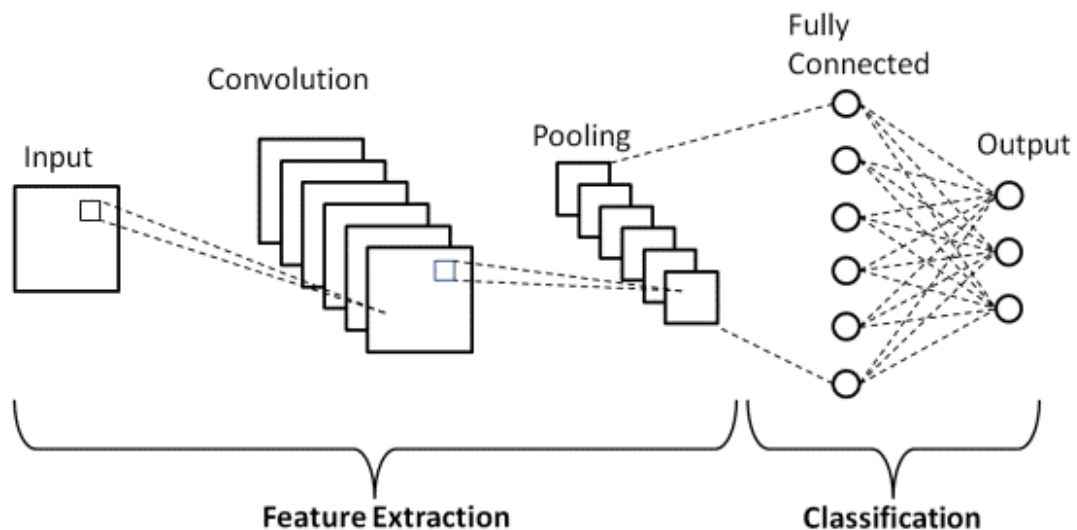
Mạng CNN là một trong những mô hình học sâu phổ biến nhất và mạnh mẽ nhất trong lĩnh vực thị giác máy tính. Nó được sử dụng rộng rãi trong các tác vụ liên quan đến hình ảnh và video. CNN có thể học được các đặc trưng của hình ảnh từ dữ liệu đầu vào, giúp cho việc nhận dạng và phân loại hình ảnh trở nên dễ dàng và chính xác.

Cấu trúc cơ bản của một mạng CNN có thể bao gồm các lớp sau [8]:

- Lớp tích chập (convolution): Lớp này sử dụng các bộ lọc (filters hoặc kernel) để trích xuất các đặc trưng từ dữ liệu đầu vào. Các bộ lọc này được áp dụng trên toàn bộ ảnh bằng cách trượt qua từng phần của ảnh để mã hóa thông tin có ích và loại bỏ thông tin không cần thiết. Kết quả của quá trình này là một bản đồ đặc trưng (feature

map) có thể chứa các đặc trưng cục bộ của ảnh như cạnh, góc, và các đặc điểm nổi bật khác.

- Lớp gộp (pooling): Lớp này thường được sử dụng sau các lớp tích chập để giảm kích thước kích thước không gian của đặc trưng và giảm độ phức tạp của mô hình bằng cách tổng hợp thông tin qua phép lấy mẫu. Phép lấy mẫu có thể là gộp theo giá trị lớn nhất (max pooling), trong đó giá trị lớn nhất trong mỗi vùng được chọn làm đại diện, hoặc là gộp theo giá trị trung bình, trong đó giá trị trung bình của mỗi vùng được tính toán.
- Hàm kích hoạt (activation function): Hàm kích hoạt được áp dụng sau mỗi lớp tích chập hoặc lớp gộp để tạo ra đầu ra phi tuyến tính của mô hình, cho phép mô hình học được các biểu diễn phức tạp hơn.. Hàm kích hoạt phổ biến nhất là ReLU, nhưng cũng có thể sử dụng các hàm khác như Tanh hoặc Sigmoid.
- Lớp kết nối đầy đủ (fully connected): Sau khi đã trích xuất các đặc trưng từ dữ liệu đầu vào, các đặc trưng này sẽ được đưa vào các lớp kết nối đầy đủ để phân loại. Các lớp này thường kết hợp với các lớp dropout để tránh quá khớp (overfitting) và cải thiện khả năng tổng quát hóa của mô hình.
- Lớp đầu ra (output): Lớp cuối cùng của mạng CNN, thường sử dụng hàm kích hoạt softmax để tính toán xác suất của các lớp đầu ra và dự đoán nhãn của ảnh đầu vào.



**Hình 2. Cấu tạo chung của mạng CNN**

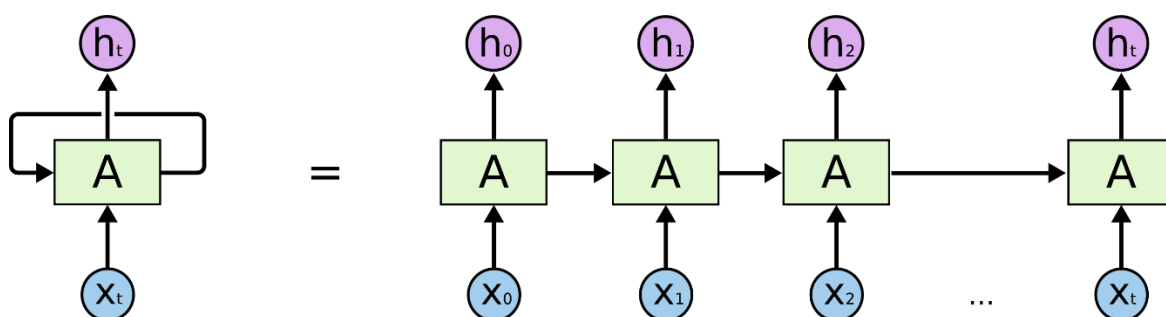
### 2.2.2. Mạng RNN

Mạng RNN là một loại mạng neural có khả năng xử lý dữ liệu chuỗi và có bộ nhớ ngắn hạn. RNN được thiết kế để nhận diện và tận dụng thông tin của các sự kiện trước đó

trong chuỗi dữ liệu. RNN hoạt động hiệu quả với các bài toán liên quan đến dữ liệu tuần tự như ngôn ngữ tự nhiên hoặc chuỗi thời gian.

Cấu trúc cơ bản của một mạng RNN bao gồm [9, 10]:

- Lớp đơn vị (cell): Đây là lớp cơ bản của mạng RNN, mỗi đơn vị trong lớp này có trạng thái ẩn (hidden state) được cập nhật sau mỗi bước thời gian dựa trên đầu vào hiện tại và trạng thái ẩn của bước thời gian trước đó. Trạng thái ẩn này chứa thông tin về lịch sử của dữ liệu chuỗi.
- Lớp đầu ra: Lớp này thường được đặt sau mỗi đơn vị RNN để tạo ra đầu ra dự đoán tại mỗi bước thời gian. Trong cấu trúc của RNN, các nút được kết nối vòng lại với chính chúng, cho phép thông tin được truyền từ bước thời gian này sang bước thời gian khác.



**Hình 3. Cấu tạo chung của một mạng RNN**

RNN hoạt động dựa trên nguyên lý truyền thông tin từ bước thời gian này sang bước thời gian khác, cho phép lưu giữ trạng thái tạm thời và "nhớ" thông tin từ quá khứ. Quá trình huấn luyện của RNN cũng sử dụng phương pháp lan truyền ngược qua thời gian để cập nhật trọng số của mạng.

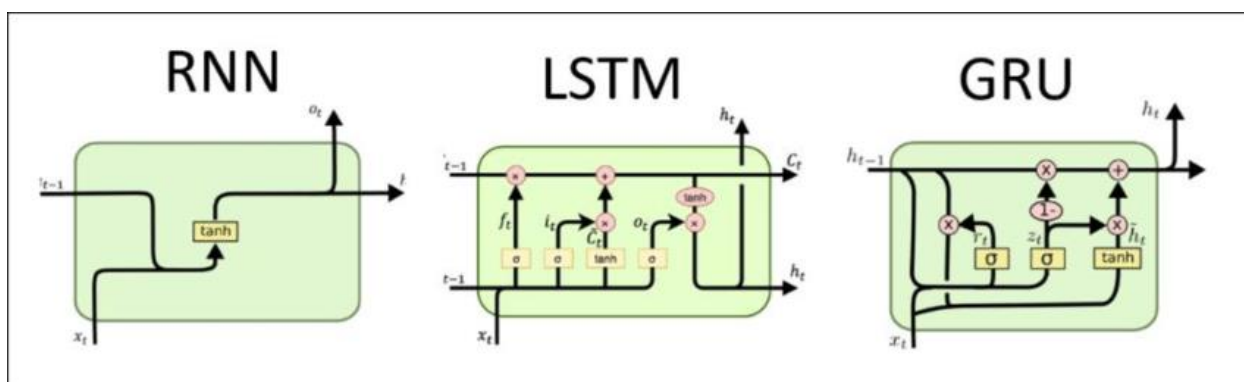
RNN được sử dụng trong nhiều ứng dụng khác nhau, từ dự đoán chuỗi thời gian trong tài chính, dự báo thời tiết, đến nhận dạng giọng nói, dịch máy, phân tích video và xử lý ngôn ngữ tự nhiên. Chúng cũng được áp dụng trong việc tạo ra văn bản tự động, nơi mà mạng có thể học cách tạo ra văn bản có cấu trúc và ngữ nghĩa tương tự như văn bản huấn luyện ban đầu.

Mạng RNN có khả năng mô hình hóa dữ liệu dạng chuỗi và mối quan hệ giữa các thành phần của chuỗi, giúp nắm bắt được thông tin liên quan đến thời gian trong dữ liệu. Tuy nhiên, một trong những hạn chế của RNN là khả năng xử lý dữ liệu dài với hiện tượng mất mát thông tin qua thời gian, hay còn gọi là vấn đề "mất mát đạo hàm" (vanishing

gradient) khiến cho việc học các phụ thuộc dài hạn trở nên khó khăn. Để giải quyết vấn đề này, các biến thể của RNN đã được phát triển và sử dụng [11], trong đó phổ biến nhất là:

- LSTM: LSTM có khả năng lưu giữ thông tin qua nhiều bước thời gian nhờ vào các cổng kiểm soát giúp điều chỉnh dòng thông tin.
- GRU: GRU là một phiên bản đơn giản hơn của LSTM, cũng sử dụng các cổng kiểm soát để điều chỉnh thông tin, nhưng có cấu trúc gọn gàng hơn.

Cả hai đều được thiết kế để có thể lưu giữ thông tin qua nhiều bước thời gian mà không gặp phải vấn đề mất mát đạo hàm.



**Hình 4. Cấu tạo của ba mạng RNN, LSTM và GRU**

Mạng RNN và các biến thể của nó đóng vai trò quan trọng trong việc mô hình hóa dữ liệu tuần tự và giải quyết các bài toán liên quan đến thời gian, góp phần quan trọng trong nhiều ứng dụng công nghệ hiện đại.

### 2.2.3. Mạng Autoencoder

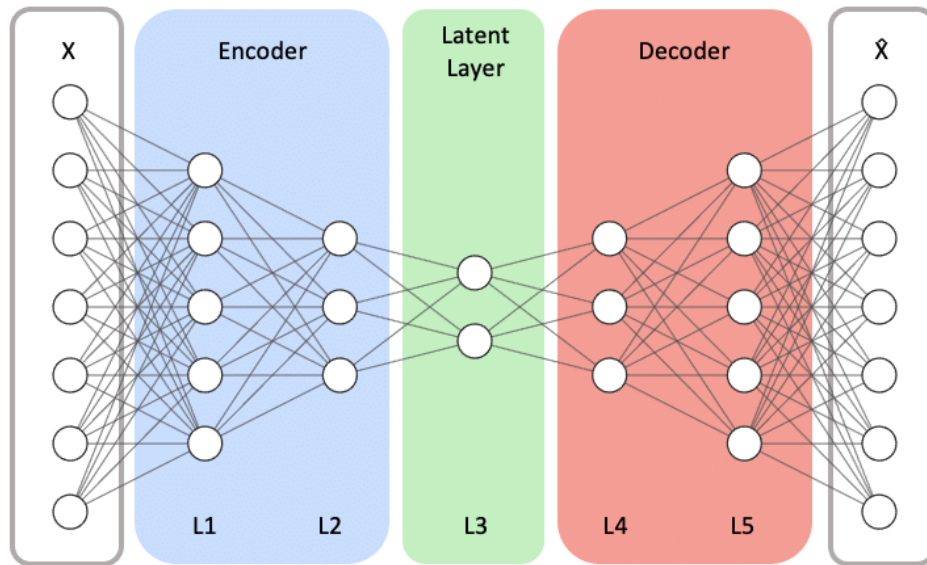
Mạng Autoencoder là một loại mạng neural được thiết kế để tự động học và trích xuất các đặc trưng quan trọng từ dữ liệu đầu vào bằng cách nén (encoding) và sau đó phục hồi (decoding) dữ liệu từ biểu diễn nén đó [12].

Cấu trúc của một mạng Autoencoder có ba phần chính [13]:

- Phần nén (encoder): Nhận đầu vào và biến đổi nó thành một biểu diễn nén có kích thước nhỏ hơn, thường được gọi là latent. Trong quá trình huấn luyện, phần này học cách trích xuất các đặc trưng quan trọng từ dữ liệu.
- Phần ẩn (latent layer): Đây là lớp ẩn giữa, nơi chứa biểu diễn nén của dữ liệu. Biểu diễn này mang thông tin quan trọng giúp mạng học được các đặc trưng từ dữ liệu.
- Phần giải nén (decoder): Nhận biểu diễn nén từ phần ẩn để tái tạo lại đầu vào ban đầu từ biểu diễn nén đó. Trong quá trình huấn luyện, phần giải nén học cách phục hồi lại dữ liệu từ biểu diễn nén và cải thiện chất lượng của dữ liệu tái tạo. Mục tiêu



của quá trình này là giảm sự khác biệt giữa dữ liệu đầu vào và dữ liệu được tái tạo.



**Hình 5. Cấu tạo của mạng Autoencoder**

Để huấn luyện một mạng Autoencoder, chúng ta cũng sử dụng thuật toán lan truyền ngược và hàm mất mát, tương tự như khi huấn luyện các mạng neural khác. Tuy nhiên, điểm khác biệt chính là thay vì huấn luyện mạng để dự đoán nhãn hoặc giá trị đầu ra, Autoencoder được huấn luyện để tái tạo lại chính dữ liệu đầu vào của nó; đòi hỏi mạng phải học cách nắm bắt và biểu diễn thông tin một cách hiệu quả trong code mà không cần đến sự giám sát từ nhãn dữ liệu. Khi mạng Autoencoder được huấn luyện đủ tốt, nó có thể tái tạo lại dữ liệu đầu vào một cách chính xác.

Autoencoder có nhiều ứng dụng trong học máy như phát hiện bất thường, giảm chiều dữ liệu, và khử nhiễu từ ảnh. Ngoài ra, Autoencoder còn có thể được sử dụng để tạo ra dữ liệu mới, có khả năng mô phỏng dữ liệu thực tế, thông qua quá trình tái tạo dữ liệu từ code.

#### **2.2.4. Mạng GAN**

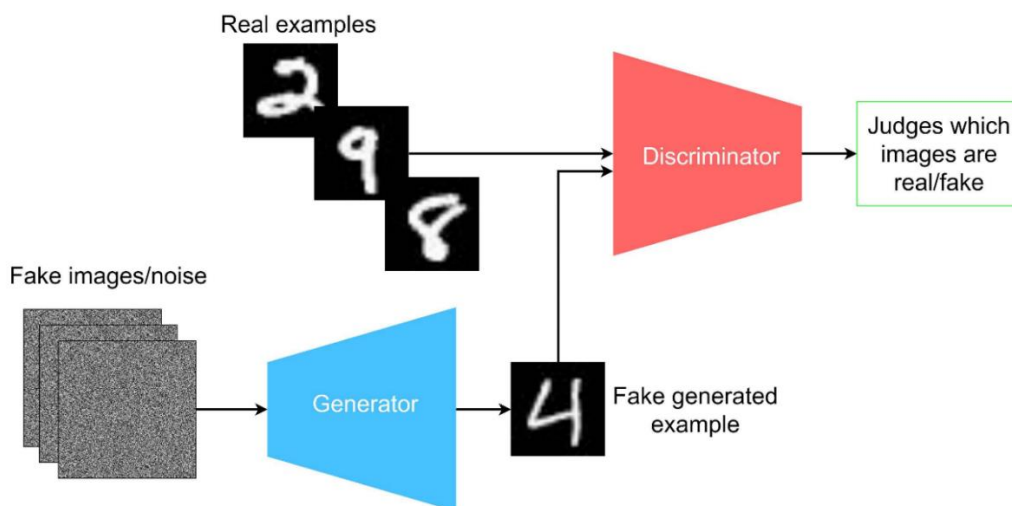
Mạng GAN là một dạng kiến trúc mạng neural đặc biệt được đề xuất bởi Ian Goodfellow và các đồng nghiệp vào năm 2014. Mạng GAN được thiết kế để tạo ra dữ liệu mới từ dữ liệu huấn luyện, có ứng dụng trong nhiều lĩnh vực như tạo ảnh, tạo âm thanh,....

Mạng GAN bao gồm hai thành phần chính [14]:

- Phần sinh (generator): có nhiệm vụ tạo ra dữ liệu giả mạo từ dữ liệu ban đầu. Nó thường được cấu tạo từ các lớp neural tích chập chuyển vị (transposed convolution hay còn gọi là deconvolutional) để chuyển đổi vector latent space thành dữ liệu giả

mạo có cấu trúc. Phần sinh cố gắng tạo ra dữ liệu sao cho giống với dữ liệu thật nhất có thể để đánh lừa phần nhận dạng (discriminator).

- Phần nhận dạng: Mạng này được huấn luyện để phân biệt giữa dữ liệu thật và dữ liệu giả mạo. Nó thường được cấu thành từ các lớp neural tích chập để trích xuất đặc trưng và các lớp kết nối đầy đủ để phân loại.



**Hình 6. Cấu tạo của mạng GAN**

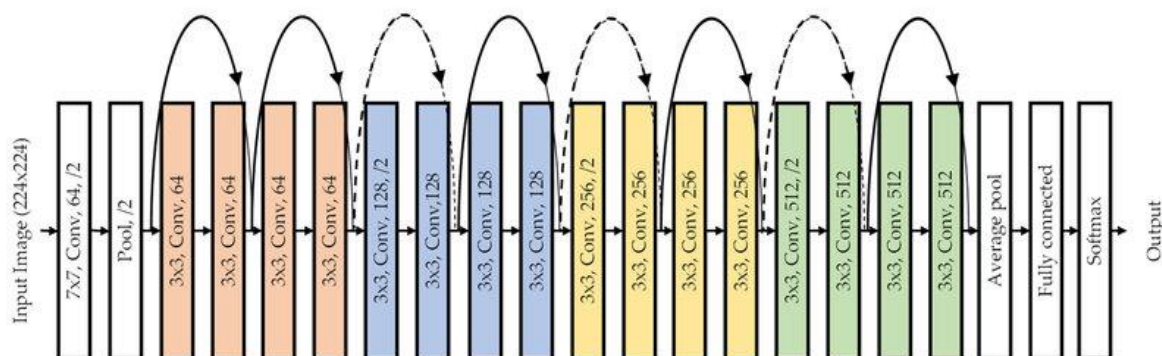
Quá trình huấn luyện của mạng GAN diễn ra như một trò chơi giữa phần sinh và phần nhận dạng: phần sinh cố gắng tạo ra dữ liệu giả mạo sao cho phần nhận dạng không thể phân biệt được, trong khi phần nhận dạng cố gắng tìm ra cách phân biệt giữa dữ liệu thật và giả mạo. Quá trình này liên tục lặp đi lặp lại, giúp cả hai phần cải thiện khả năng của chúng.

#### **2.2.5. Mạng ResNet**

Mạng ResNet là một kiến trúc mạng neural tích chập sâu được giới thiệu bởi Kaiming He và các cộng sự tại Microsoft Research vào năm 2015. Kiến trúc này đã giành chiến thắng trong cuộc thi ILSVRC (ImageNet Large Scale Visual Recognition Challenge) trong cùng năm đó. ResNet sử dụng các đơn vị khối phần dư (residual blocks), cho phép xây dựng mạng được đến hàng trăm hoặc thậm chí hàng nghìn tầng.

Cấu trúc cơ bản của ResNet bao gồm các khối phần dư. Mỗi khối phần dư bao gồm một loạt các lớp được nối với nhau theo cấu trúc "ánh xạ đồng nhất" (identity mapping). ResNet sử dụng các kết nối tắt (shortcut connection) hay còn gọi là kết nối phần dư, trong đó đầu vào của một lớp được cộng trực tiếp vào đầu ra của lớp sau một số lớp trung gian, cho phép đầu vào được truyền thẳng qua mạng. Quá trình này giúp giảm thiểu hiện tượng mất mát đạo hàm và giúp mạng học được biểu diễn sâu hơn [15].

ResNet có nhiều biến thể với số lượng lớp khác nhau như ResNet-18, ResNet-34, ResNet-50, ResNet-101, và ResNet-152. Số lượng lớp càng nhiều thì mô hình càng sâu và có khả năng học được nhiều đặc trưng phức tạp hơn, nhưng cũng đòi hỏi nhiều tài nguyên tính toán hơn.



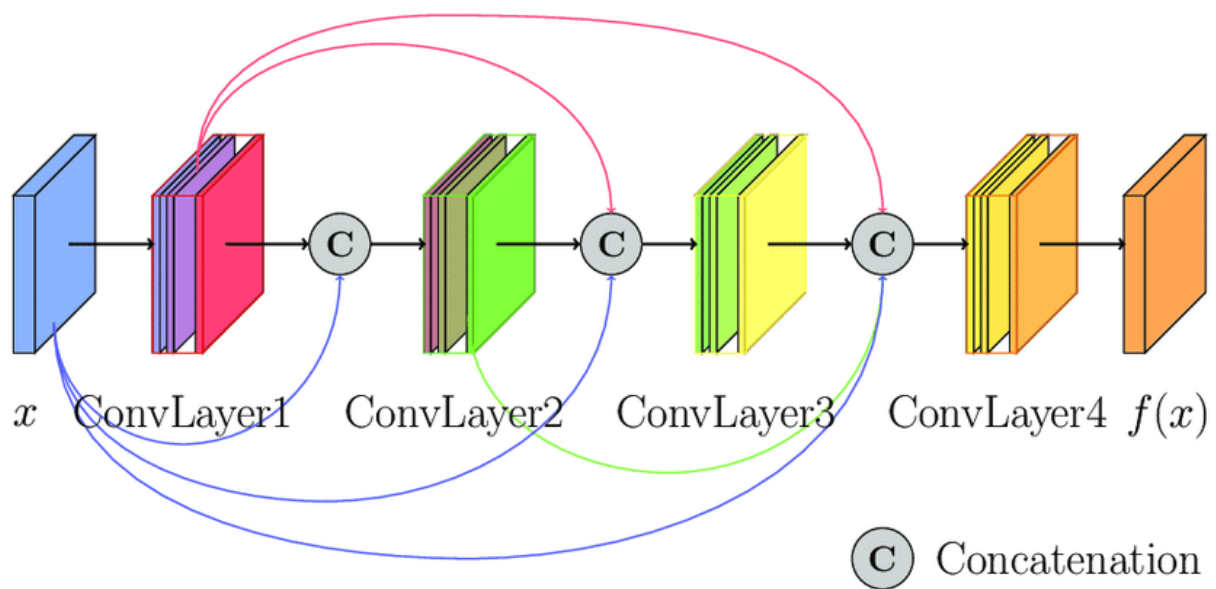
**Hình 7. Cấu tạo của một biến thể ResNet – ResNet18**

Nhờ vào khả năng học được các đặc trưng sâu, ResNet có thể giúp cải thiện hiệu suất phát hiện bất thường và giảm thiểu việc mất mát thông tin trong quá trình lan truyền ngược.

#### **2.2.6. Mạng DenseNet**

Mạng DenseNet là một kiến trúc mạng neural sâu được giới thiệu bởi Gao Huang và cộng sự vào năm 2017.. DenseNet đã giải quyết vấn đề mất mát đạo hàm trong quá trình huấn luyện các mạng neural sâu bằng cách sử dụng kỹ thuật gọi là “kết nối dày đặc” (dense connection). Kết nối dày đặc là một kỹ thuật trong đó đầu ra của mỗi lớp được kết nối đến tất cả các lớp sau đó. Điều này cho phép gradient được truyền thẳng qua mạng, giúp cải thiện hiệu suất huấn luyện.

Mạng DenseNet được xây dựng từ các khối dày đặc (dense block). Một khối dày đặc bao gồm một loạt các lớp tích chập được kết nối với nhau, mỗi lớp sau đó được theo sau bởi một hàm kích hoạt ReLU. Các đầu ra của các lớp tích chập được nối lại với nhau bằng cách sử dụng phép nối (concatenate), tạo thành một đầu ra dày đặc (dense output) giúp cải thiện hiệu suất huấn luyện bằng cách tận dụng lại thông tin từ tất cả các lớp trước đó [16].



**Hình 8. Cấu tạo của mạng DenseNet**

Nhờ vào khả năng của nó trong việc tận dụng lại thông tin từ tất cả các lớp, mạng DenseNet có thể học được biểu diễn phức tạp của dữ liệu hình ảnh và tăng cường hiệu suất phát hiện.

### 2.3. Quy trình huấn luyện và đánh giá một mô hình học sâu

Các bước thực hiện chính như sau:

- Thu thập dữ liệu: Nguồn hình ảnh y khoa có thể được thu thập từ các nguồn khác nhau như máy chụp cắt lớp hay hình ảnh X-quang.
- Tiền xử lý dữ liệu: có thể bao gồm chuẩn hóa, loại bỏ nhiễu, tách tập dữ liệu hoặc tăng cường dữ liệu.
- Xây dựng mô hình học sâu bằng cách xem xét và lựa chọn các mô hình học sâu như CNN, ResNet, DenseNet,..., sau đó tiến hành cài đặt mô hình và thiết lập các tham số của mô hình như số lượng lớp, kích thước lớp, hàm kích hoạt.
- Huấn luyện mô hình: Sử dụng dữ liệu đã chuẩn bị để huấn luyện mô hình. Quá trình huấn luyện bao gồm việc tối ưu hóa các tham số của mô hình thông qua các thuật toán học sâu để tối ưu hóa hiệu suất.
- Đánh giá mô hình: Sau khi huấn luyện, mô hình sẽ được đánh giá trên tập dữ liệu kiểm tra. Các chỉ số như độ chính xác, F1-score hoặc độ nhạy sẽ được sử dụng để đo lường hiệu suất của mô hình. Nếu cần thiết, dựa trên kết quả đánh giá, điều chỉnh các siêu tham số hoặc cấu trúc của mô hình để cải thiện hiệu suất.

## **CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU**

### **3.1. Mô tả bài toán**

Phát hiện bất thường trong ảnh là một phần quan trọng trong lĩnh vực y học hình ảnh, nhằm nhận diện các khu vực hoặc đối tượng trong hình ảnh y khoa mà không tuân theo mẫu chuẩn hoặc có sự biến đổi bất thường so với trạng thái bình thường. Bài toán này đòi hỏi độ chính xác cao do mức độ nghiêm trọng của việc chẩn đoán sai lệch. Do đó, việc thu thập và xử lý dữ liệu cần phải được thực hiện một cách cẩn thận, đảm bảo tính toàn vẹn và độ tin cậy của dữ liệu.

Đầu vào của bài toán là ảnh y khoa, các ảnh này có kích thước lớn và độ phức tạp cao, với các đối tượng và biến đổi bất thường xuất hiện ở nhiều hình dạng và kích thước khác nhau. Sau đó các ảnh sẽ được đưa vào các mô hình để nhận dạng. Đầu ra của bài toán cho biết hình ảnh đầu vào là bình thường hay bất thường.

Trong phạm vi của khoá luận này, bài toán sẽ đề cập về viêm phổi. Viêm phổi là một bệnh nhiễm trùng phổi gây ra bởi vi khuẩn, virus hoặc nấm; bệnh làm viêm các túi khí trong phổi, gây ra triệu chứng như ho, sốt, khó thở và đau ngực. Viêm phổi có thể ảnh hưởng nghiêm trọng, đặc biệt ở người già, trẻ em và những người có hệ miễn dịch yếu. Phát hiện sớm viêm phổi là rất quan trọng để điều trị kịp thời và giảm thiểu biến chứng. Các phương pháp chẩn đoán truyền thống như chụp X-quang và xét nghiệm máu có thể mất thời gian và không phải lúc nào cũng chính xác. Do đó, việc áp dụng các mô hình học sâu để phát hiện viêm phổi từ ảnh y khoa đang trở thành một hướng nghiên cứu tiềm năng. Các mô hình học sâu có thể tự động phát hiện các dấu hiệu của viêm phổi từ ảnh X-quang với độ chính xác cao. Điều này không chỉ giúp cải thiện tốc độ chẩn đoán mà còn hỗ trợ các bác sĩ trong việc đưa ra quyết định điều trị.

### **3.2. Những kết quả đã đạt được trước đó**

Các nghiên cứu trước đây đã tiếp cận vấn đề này thông qua nhiều phương pháp khác như phân tích thành phần chính và máy vector hỗ trợ [17]. Một số nghiên cứu đã tập trung vào việc mô hình hóa nền để phát hiện sự bất thường dựa trên sự khác biệt so với nền thông thường, trong khi những nghiên cứu khác lại hướng đến việc phát hiện bất thường thông qua việc phân tích ảnh dư, nơi mà nhiễu và bất thường là chủ đạo [18]. Các phương pháp này đã được chứng minh là có hiệu quả trong việc phát hiện các loại bất thường khác nhau, từ những thay đổi nhỏ trong kết cấu mô đến những khối u lớn.

Tuy nhiên, việc phát hiện bất thường trong ảnh y khoa vẫn còn nhiều thách thức,

bao gồm việc xác định những gì là "bình thường" và "bất thường", sự đa dạng của các loại bất thường, và khả năng của mô hình trong việc tổng quát hóa từ dữ liệu huấn luyện đến dữ liệu thực tế.

Nghiên cứu này tập trung vào việc xem xét và phát triển các mô hình học sâu để giải quyết những hạn chế của các phương pháp hiện tại, và có khả năng thích ứng tốt hơn với sự đa dạng của dữ liệu y khoa và cải thiện độ chính xác trong việc phát hiện bất thường.

### **3.3. Đề xuất cách giải quyết bài toán**

Các mô hình học sâu đã được chứng minh là hiệu quả trong việc xử lý dữ liệu không cân xứng, giúp giảm thiểu sự thiên vị về phía nhóm dữ liệu lớn hơn, điều này rất quan trọng trong xử lý ảnh y khoa vì số lượng ảnh cho trường hợp bình thường thường nhiều hơn so với trường hợp bất thường.

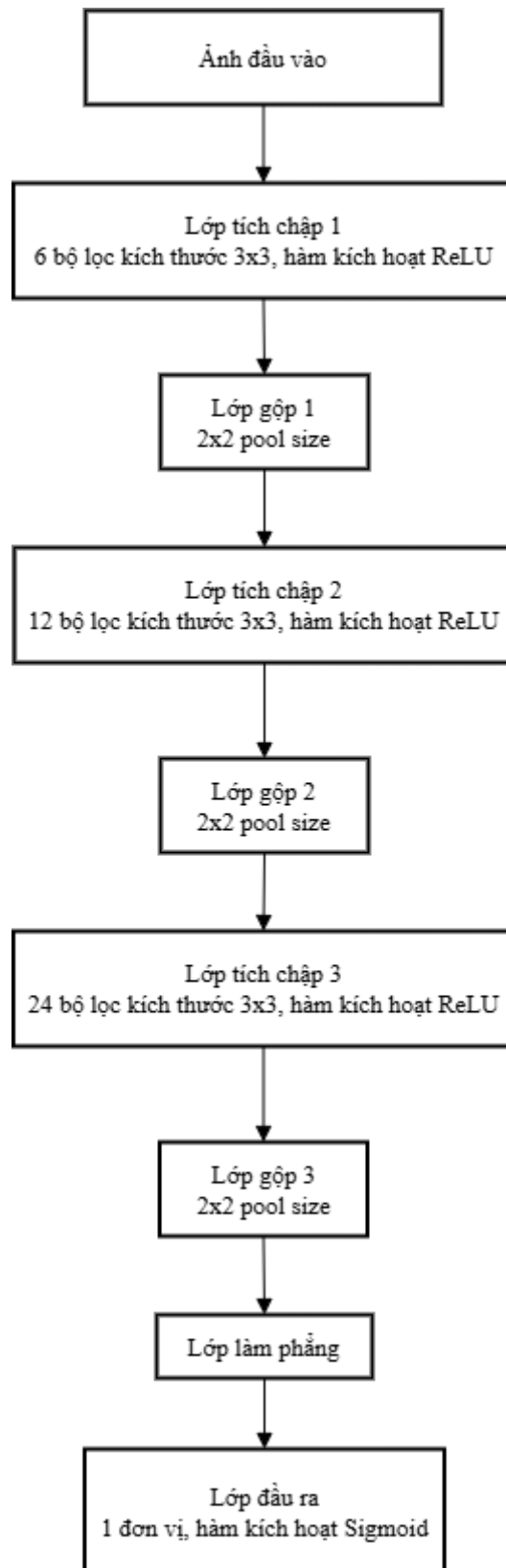
Vì vậy, trong khoá luận này, em sử dụng các mô hình học sâu bao gồm mạng CNN, mạng ResNet, và mạng DenseNet. Các mô hình này đặc biệt phù hợp với việc trích xuất đặc trưng từ ảnh và học các đặc trưng phức tạp và quan trọng để phân biệt giữa các vùng bình thường và bất thường trong ảnh y khoa.

Ngoài ra, các mô hình này còn sử dụng các lớp phi tuyến tính với các hàm kích hoạt như ReLU và Sigmoid để tăng khả năng học của mạng. Phương pháp tối ưu hóa Adam được chọn để điều chỉnh trọng số của mạng, giúp mô hình hội tụ nhanh và hiệu quả hơn.

Sau đây là kiến trúc của từng mô hình được đề xuất.

#### **3.3.1. Kiến trúc của CNN**

Mô hình CNN được đề xuất như sau:



**Hình 9. Sơ đồ khối kiến trúc CNN được đề xuất**

- Ảnh đầu vào với kích thước và số kênh màu của ảnh.
- Lớp tích chập 1: sử dụng 6 bộ lọc với kích thước mỗi bộ lọc là 3x3 và hàm kích hoạt ReLU. Mục tiêu của lớp này là trích xuất các đặc trưng cơ bản từ ảnh đầu vào. như

cạnh và góc cạnh từ ảnh. Hàm kích hoạt ReLU giúp mô hình học các đặc trưng phi tuyến tính.

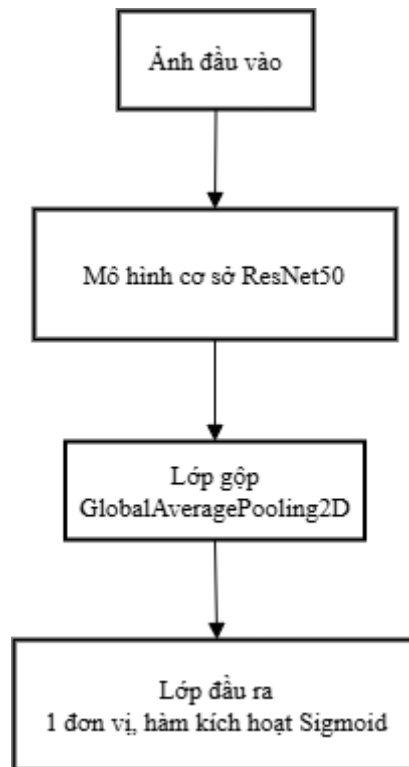
- Lớp gộp 1: sử dụng phép gộp theo giá trị lớn nhất với kích thước  $2 \times 2$  để giảm kích thước của đặc trưng, giữ lại các đặc trưng quan trọng nhất và giảm thiểu tính toán cho các lớp tiếp theo.
- Lớp tích chập 2: sử dụng 12 bộ lọc với kích thước mỗi bộ lọc là  $3 \times 3$  và hàm kích hoạt ReLU. Lớp này giúp trích xuất các đặc trưng phức tạp hơn từ ảnh.
- Lớp gộp 2: sử dụng phép gộp theo giá trị lớn nhất với kích thước  $2 \times 2$  để tiếp tục trích xuất và giảm kích thước các đặc trưng phức tạp hơn.
- Lớp tích chập 3: sử dụng 24 bộ lọc với kích thước bộ lọc là  $3 \times 3$  và hàm kích hoạt ReLU, giúp trích xuất các đặc trưng chi tiết hơn nữa.
- Lớp gộp 3: sử dụng phép gộp theo giá trị lớn nhất với kích thước  $2 \times 2$ . Lớp này tiếp tục giảm kích thước đầu ra.
- Lớp làm phẳng: biến đổi đầu ra từ lớp gộp 3 thành một vector một chiều để đưa vào lớp đầu ra
- Lớp đầu ra: 1 đơn vị với hàm kích hoạt sigmoid để phân loại ảnh đầu vào thành bình thường hoặc bất thường dựa trên các đặc trưng đã học.

Mô hình CNN được biên dịch sử dụng thuật toán tối ưu hoá Adam, hàm mất mát `binary_crossentropy` và đánh giá mô hình bằng loss và accuracy trên tập dữ liệu kiểm tra.

### **3.3.2. Kiến trúc của ResNet**

Mô hình ResNet được đề xuất dựa trên kiến trúc ResNet50 gồm các phần chính sau:



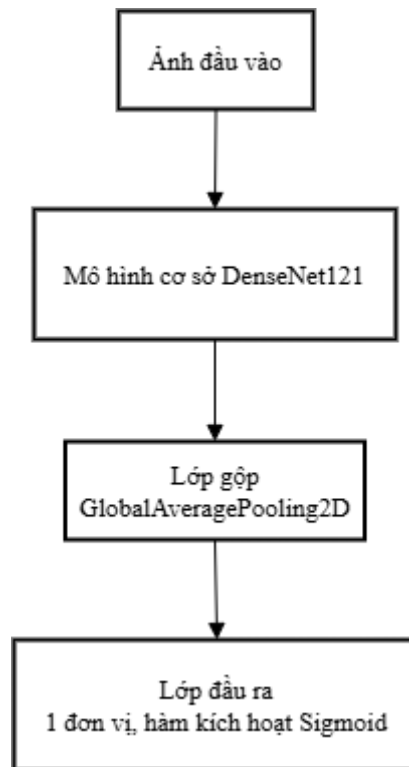


**Hình 10. Sơ đồ khối kiến trúc ResNet được đề xuất**

- Ảnh đầu vào với kích thước và số kênh màu của ảnh.
  - Mô hình cơ sở ResNet50: được lấy từ thư viện Keras, đã được huấn luyện trước trên tập dữ liệu ImageNet.
  - Lớp gộp trung bình chung: giảm kích thước của hình ảnh xuống còn một vector duy nhất cho mỗi kênh màu.
  - Lớp đầu ra: cũng là 1 đơn vị với hàm kích hoạt là sigmoid để phân loại.
- Thuật toán tối ưu hoá, hàm mất mát và chỉ số đánh giá mô hình tương tự như trong mục 3.3.1.

### **3.3.3. Kiến trúc của DenseNet**

Mô hình DenseNet, cũng tương tự ResNet, có cấu trúc chi tiết như sau:



**Hình 11. Sơ đồ khối kiến trúc DenseNet được đề xuất**

- Ảnh đầu vào với kích thước và số kênh màu của ảnh.
- Mô hình cơ sở DenseNet121: cũng được lấy từ thư viện Keras và đã được huấn luyện trước trên tập dữ liệu ImageNet.
- Lớp gộp trung bình chung: tương tự ResNet, giảm kích thước của hình ảnh xuống một vector duy nhất tạo ra một đầu ra.
- Lớp đầu ra: cũng với 1 đơn vị và hàm kích hoạt sigmoid.

Thuật toán tối ưu hoá, hàm mất mát và chỉ số đánh giá mô hình tương tự như trong mục 3.3.1.

### **3.4. Giới thiệu tập dữ liệu**

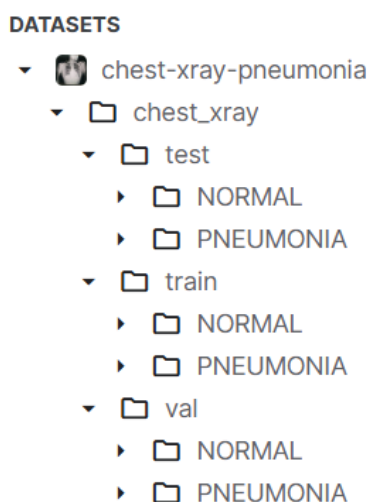
Tập dữ liệu được sử dụng trong khoá luận là tập dữ liệu hình ảnh X-quang phổi (Chest X-Ray Images (Pneumonia)).

Dữ liệu được cung cấp bởi bài báo “Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification” trên kho lưu trữ Mendeley Data [19] và được tổng hợp trên Kaggle: <https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia>.

Tập dữ liệu bao gồm tổng cộng 5,863 hình ảnh, được chia làm hai loại danh mục hình ảnh, bao gồm hình ảnh phổi bình thường và hình ảnh viêm phổi. Tập dữ liệu này

được sử dụng để hỗ trợ việc phân loại bệnh viêm phổi thông qua hình ảnh X-quang, đồng thời là nguồn dữ liệu quan trọng cho các nghiên cứu và phát triển mô hình học máy trong lĩnh vực y tế.

Dữ liệu được tổ chức thành ba tập thư mục con: tập huấn luyện, tập kiểm tra và tập đánh giá. Mỗi thư mục con lại có thêm hai thư mục con nhỏ nữa tương ứng với hai nhãn bình thường và viêm phổi.



**Hình 12. Cây thư mục của tập dữ liệu**

### 3.5. Môi trường cài đặt

Để triển khai và huấn luyện mô hình, khoá luận sử dụng Kaggle, một nền tảng trực tuyến nổi tiếng trong cộng đồng khoa học dữ liệu và học máy. Được thành lập vào năm 2010 và hiện là một phần của Google, Kaggle cung cấp một môi trường làm việc toàn diện cho các chuyên gia và những người đam mê trong lĩnh vực này để thực hành, cạnh tranh và học hỏi từ nhau.

Kaggle cho phép người dùng viết và chạy mã trực tiếp trong trình duyệt mà không cần thiết lập môi trường cục bộ. Kaggle cung cấp một môi trường tính toán đám mây, giúp người dùng thực hiện các dự án khoa học dữ liệu và xây dựng mô hình học máy mà không cần cấu hình máy tính quá mạnh. Kaggle hỗ trợ Python và R, hai ngôn ngữ phổ biến nhất trong khoa học dữ liệu. Kaggle cung cấp sẵn các tài nguyên tính toán cần thiết, bao gồm GPU và TPU, giúp tăng tốc quá trình xử lý và huấn luyện mô hình.

Kaggle còn cho phép khả năng kết nối trực tiếp với các bộ dữ liệu trên Kaggle, giúp quá trình tải và xử lý dữ liệu trở nên đơn giản và nhanh chóng.



**Hình 13. Logo của Kaggle**

## CHƯƠNG 4. KẾT QUẢ NGHIÊN CỨU

### 4.1. Chuẩn bị thư viện và dữ liệu

Khoá luận sử dụng các thư viện của Python để xây dựng và thiết lập các mô hình học sâu như thể hiện trong hình 15 sau:

```
import tensorflow as tf
import keras
from keras import datasets, layers, models
from keras.layers import Input, Dense, Conv2D, MaxPooling2D, UpSampling2D, concatenate
from keras.models import Model
import numpy as np
```

```
from keras.applications.resnet50 import ResNet50
from keras.layers import Dense, GlobalAveragePooling2D
from keras.optimizers import Adam
```

```
import matplotlib.pyplot as plt
```

**Hình 14. Các thư viện cho việc xây dựng các mô hình**

TensorFlow và Keras là hai thư viện phổ biến trong lĩnh vực học sâu. TensorFlow cung cấp một nền tảng mạnh mẽ cho việc xây dựng và huấn luyện các mô hình học sâu. Keras, một giao diện Python cho TensorFlow, giúp đơn giản hóa quá trình xây dựng mô hình bằng cách cung cấp các lớp và hàm tiện ích.

Trong thư viện Keras, Layers chứa các lớp cơ bản được sử dụng để xây dựng mô hình học sâu, bao gồm các lớp như Dense (lớp kết nối đầy đủ), Conv2D (lớp tích chập 2 chiều), MaxPooling2D (lớp gộp theo giá trị lớn nhất 2 chiều), và nhiều lớp khác. Các lớp này giúp xây dựng kiến trúc của mô hình học sâu.

Numpy và Matplotlib là hai thư viện quan trọng trong Python. Numpy hỗ trợ các phép toán khoa học và đại số tuyến tính, trong khi matplotlib hỗ trợ việc vẽ đồ thị và trực quan hóa dữ liệu.

Ngoài ra, do số lượng hình ảnh trong mỗi thư mục chưa được chuẩn (xem hình 18), nên cần thêm các thư viện sau để sắp xếp lại tập dữ liệu:

```
import shutil
import random
```

**Hình 15. Các thư viện dùng cho việc sắp xếp dữ liệu**

Thư viện shutil (shell utilities) cung cấp một số hàm tiện ích cao cấp để thao tác với các tệp và thư mục, như sao chép, di chuyển, hoặc xóa các tệp và thư mục. Nó cũng hỗ trợ việc sao chép quyền truy cập và thời gian của các tệp, cũng như toàn bộ cây thư mục.

Thư viện random được sử dụng để tạo ra các số ngẫu nhiên. Nó cung cấp các hàm như random(), randint(), uniform(), và choice() để tạo ra các số ngẫu nhiên theo các phân phối khác nhau.

Tiếp theo chúng ta khai báo một số biến dựa trên cấu trúc của tập dữ liệu:

```
dataset_path = '/kaggle/input/chest-xray-pneumonia/chest_xray'
subfolders = ['train', 'val', 'test']
labels = ['NORMAL', 'PNEUMONIA']
total_images = 0
```

### Hình 16. Các biến dựa trên cấu trúc của tập dữ liệu

Kiểm tra số lượng hình ảnh trong từng thư mục:

```
def count_images_in_folder(folder_path):
    return len([f for f in os.listdir(folder_path) if os.path.isfile(os.path.join(folder_path, f))])
```

```
for subfolder in subfolders:
    for label in labels:
        folder_path = os.path.join(dataset_path, subfolder, label)
        num_images = count_images_in_folder(folder_path)
        total_images += num_images
        print(f'Thư mục {folder_path} có {num_images} hình ảnh')

print(f'Tổng số hình ảnh trong tập dữ liệu: {total_images}')
```

```
Thư mục /kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL có 1341 hình ảnh
Thư mục /kaggle/input/chest-xray-pneumonia/chest_xray/train/PNEUMONIA có 3875 hình ảnh
Thư mục /kaggle/input/chest-xray-pneumonia/chest_xray/val/NORMAL có 8 hình ảnh
Thư mục /kaggle/input/chest-xray-pneumonia/chest_xray/val/PNEUMONIA có 8 hình ảnh
Thư mục /kaggle/input/chest-xray-pneumonia/chest_xray/test/NORMAL có 234 hình ảnh
Thư mục /kaggle/input/chest-xray-pneumonia/chest_xray/test/PNEUMONIA có 390 hình ảnh
Tổng số hình ảnh trong tập dữ liệu: 5856
```

### Hình 17. Số lượng hình ảnh trong từng thư mục

Như chúng ta có thể thấy, có sự không tương xứng giữa số lượng hình ảnh của các thư mục với nhau. Do đó trong trường hợp này, chúng ta sẽ sắp xếp và chia lại số lượng hình ảnh giữa các tập dữ liệu theo tỷ lệ là train:val:test = 70%:20%:10%.

Để làm điều đó, chúng ta sẽ thực hiện tạo một thư mục mới chứa tập dữ liệu theo tỷ lệ nêu trên. Các đoạn mã nguồn sau đây thực hiện công việc đó:

```
output_dataset_path = '/kaggle/working/chest_xray'
os.makedirs(output_dataset_path, exist_ok=True)
```

### Hình 18. Khai báo vị trí của thư mục mới cho việc sắp xếp dữ liệu

```
for subfolder in subfolders:
    for label in labels:
        folder_path = os.path.join(dataset_path, subfolder, label)
        images = [f for f in os.listdir(folder_path) if os.path.isfile(os.path.join(folder_path,
f)))]

        random.seed(42)
        random.shuffle(images)
        num_train = int(0.7 * len(images))
        num_val = int(0.1 * len(images))
        num_test = len(images) - num_train - num_val

        train_images = images[:num_train]
        val_images = images[num_train:num_train + num_val]
        test_images = images[num_train + num_val:]
```

### Hình 19. Thực hiện việc xáo trộn và phân chia hình ảnh theo tỷ lệ

```
for split in subfolders:
    os.makedirs(os.path.join(output_dataset_path, split, label), exist_ok=True)

    # Di chuyển hình ảnh vào thư mục mới
    for image in train_images:
        shutil.copy(os.path.join(folder_path, image), os.path.join(output_dataset_path, 'train',
label, image))
    for image in val_images:
        shutil.copy(os.path.join(folder_path, image), os.path.join(output_dataset_path, 'val',
label, image))
    for image in test_images:
        shutil.copy(os.path.join(folder_path, image), os.path.join(output_dataset_path, 'test',
label, image))
```

### Hình 20. Sao chép lần lượt hình ảnh vào các thư mục mới

Cuối cùng, kiểm tra lại thư mục vừa mới tạo cùng với dữ liệu được sắp xếp:

```
total_images = 0
for subfolder in subfolders:
    for label in labels:
        folder_path = os.path.join(output_dataset_path, subfolder, label)
        num_images = count_images_in_folder(folder_path)
        total_images += num_images
        print(f'Thư mục đã chia {folder_path} có {num_images} hình ảnh')

print(f'Tổng số hình ảnh trong tập dữ liệu: {total_images}')
```

Thư mục đã chia /kaggle/working/chest\_xray/train/NORMAL có 1106 hình ảnh  
 Thư mục đã chia /kaggle/working/chest\_xray/train/PNEUMONIA có 2990 hình ảnh  
 Thư mục đã chia /kaggle/working/chest\_xray/val/NORMAL có 157 hình ảnh  
 Thư mục đã chia /kaggle/working/chest\_xray/val/PNEUMONIA có 426 hình ảnh  
 Thư mục đã chia /kaggle/working/chest\_xray/test/NORMAL có 320 hình ảnh  
 Thư mục đã chia /kaggle/working/chest\_xray/test/PNEUMONIA có 857 hình ảnh  
 Tổng số hình ảnh trong tập dữ liệu: 5856

**Hình 21. Kiểm tra lại số lượng hình ảnh sau khi chia**

## 4.2. Xây dựng các mô hình

- Mô hình CNN:

```
model_CNN = models.Sequential()
model_CNN.add(layers.Conv2D(6, (3, 3), activation='relu', input_shape=image_size + (3,)))
model_CNN.add(layers.MaxPooling2D((2, 2)))
model_CNN.add(layers.Conv2D(12, (3, 3), activation='relu'))
model_CNN.add(layers.MaxPooling2D((2, 2)))
model_CNN.add(layers.Conv2D(24, (3, 3), activation='relu'))
model_CNN.add(layers.MaxPooling2D((2, 2)))
model_CNN.add(layers.Flatten())
model_CNN.add(layers.Dense(1, activation='sigmoid'))
```

**Hình 22. Mã nguồn kiến trúc mô hình CNN**

- Mô hình ResNet

```
base_model = ResNet50(weights='imagenet', include_top=False)
# Thêm lớp phân loại cuối cùng
x = base_model.output
x = GlobalAveragePooling2D()(x)
predictions = Dense(1, activation='sigmoid')(x) # 2 lớp đầu ra: "bình thường" và "bất thường"
# Tạo mô hình mới
model_resnet = Model(inputs=base_model.input, outputs=predictions)
for layer in base_model.layers:
    layer.trainable = False

model_resnet.layers[-1].trainable = True
```

**Hình 23. Mã nguồn kiến trúc mô hình ResNet**

- Mô hình DenseNet



```
from keras.applications import DenseNet121
```

```
base_model = DenseNet121(weights='imagenet', include_top=False)
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1, activation='sigmoid')(x)

model_densenet = Model(inputs=base_model.input, outputs=x)
```

**Hình 24. Mã nguồn kiến trúc mô hình DenseNet**

### 4.3. Huấn luyện các mô hình

Trong khoá luận này, các tham số chính được sử dụng trong quá trình huấn luyện các mô hình là tương tự nhau, bao gồm:

- Thuật toán tối ưu hoá: Adam
- Learning rate: 0.001
- Số lần học (epochs): 50
- Hàm mất mát: binary\_crossentropy (do bài toán là phân loại nhị phân)
- Chỉ số đánh giá: Accuracy

Tập huấn luyện (val\_ds) được sử dụng để điều chỉnh các trọng số của mô hình thông qua quá trình lan truyền ngược và tối ưu hóa.

```
model_CNN.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])
```

```
epochs = 50
hist_CNN = model_CNN.fit(train_ds, epochs = epochs, validation_data = val_ds)
```

**Hình 25. Biên dịch và huấn luyện mô hình CNN**

```
model_resnet.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])
```

```
hist_resnet = model_resnet.fit(train_ds, epochs = epochs, validation_data = val_ds)
```

**Hình 26. Biên dịch và huấn luyện mô hình ResNet**

```
model_densenet.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=[ 'accuracy' ])
```

```
hist_densenet = model_densenet.fit(train_ds, epochs = epochs, validation_data = val_ds)
```

### Hình 27. Biên dịch và huấn luyện mô hình DenseNet

Các mô hình được huấn luyện trong tổng cộng 50 epochs. Đối với hai mô hình CNN và ResNet, ngoại trừ epoch đầu tiên, mỗi epoch còn lại mất khoảng 9 đến 12 giây để hoàn thành. Với mô hình DenseNet, khoảng thời gian cần thiết để huấn luyện là gấp đôi so với hai mô hình kia. Quá trình huấn luyện được theo dõi và ghi nhận để đánh giá các chỉ số hiệu suất như độ chính xác và hàm mất mát theo thời gian.

```
epochs = 50  
hist_CNN = model_CNN.fit(train_ds, epochs = epochs, validation_data = val_ds)
```

Epoch 1/50  
2/64 ————— 5s 90ms/step - accuracy: 0.6406 - loss: 48.0187

WARNING: All log messages before absl::InitializeLog() is called are written to STDERR  
I0000 00:00:1719808527.921541 114 device\_compiler.h:186] Compiled cluster using XLA! This line is logged at most once for the lifetime of the process.

64/64 ————— 20s 208ms/step - accuracy: 0.6501 - loss: 26.4327 - val\_accuracy: 0.9177 - val\_loss: 0.2326  
Epoch 2/50  
64/64 ————— 13s 199ms/step - accuracy: 0.9100 - loss: 0.2427 - val\_accuracy: 0.9400 - val\_loss: 0.1620  
Epoch 3/50  
64/64 ————— 12s 192ms/step - accuracy: 0.9283 - loss: 0.1816 - val\_accuracy: 0.9348 - val\_loss: 0.1489  
Epoch 4/50  
64/64 ————— 12s 191ms/step - accuracy: 0.9439 - loss: 0.1490 - val\_accuracy: 0.9297 - val\_loss: 0.1431  
Epoch 5/50  
64/64 ————— 13s 199ms/step - accuracy: 0.9539 - loss: 0.1164 - val\_accuracy: 0.9434 - val\_loss: 0.1237

### Hình 28. Minh hoạ việc huấn luyện mô hình CNN

```
hist_resnet = model_resnet.fit(train_ds, epochs = epochs, validation_data = val_ds)
```

```
Epoch 1/50
64/64 ————— 39s 345ms/step - accuracy: 0.7779 - loss: 0.4886 - val_accuracy: 0.9280 - val_loss: 0.1794
Epoch 2/50
64/64 ————— 13s 207ms/step - accuracy: 0.9268 - loss: 0.1907 - val_accuracy: 0.9331 - val_loss: 0.1649
Epoch 3/50
64/64 ————— 14s 211ms/step - accuracy: 0.9456 - loss: 0.1537 - val_accuracy: 0.9503 - val_loss: 0.1430
Epoch 4/50
64/64 ————— 13s 204ms/step - accuracy: 0.9554 - loss: 0.1358 - val_accuracy: 0.9554 - val_loss: 0.1274
Epoch 5/50
64/64 ————— 12s 193ms/step - accuracy: 0.9584 - loss: 0.1243 - val_accuracy: 0.9571 - val_loss: 0.1208
Epoch 6/50
64/64 ————— 13s 200ms/step - accuracy: 0.9641 - loss: 0.1100 - val_accuracy: 0.9623 - val_loss: 0.1207
Epoch 7/50
64/64 ————— 12s 192ms/step - accuracy: 0.9651 - loss: 0.1101 - val_accuracy: 0.9657 - val_loss: 0.1133
Epoch 8/50
64/64 ————— 13s 198ms/step - accuracy: 0.9678 - loss: 0.1018 - val_accuracy: 0.9657 - val_loss: 0.1113
Epoch 9/50
64/64 ————— 12s 191ms/step - accuracy: 0.9677 - loss: 0.0974 - val_accuracy: 0.9588 - val_loss: 0.1188
```

**Hình 29. Minh họa việc huấn luyện mô hình ResNet**

```
hist_densenet = model_densenet.fit(train_ds, epochs = epochs, validation_data = val_ds)
```

```
Epoch 1/50
64/64 ————— 211s 681ms/step - accuracy: 0.8875 - loss: 0.2873 - val_accuracy: 0.7307 - val_loss: 17.6924
Epoch 2/50
64/64 ————— 23s 349ms/step - accuracy: 0.9695 - loss: 0.0835 - val_accuracy: 0.8971 - val_loss: 0.4912
Epoch 3/50
64/64 ————— 41s 344ms/step - accuracy: 0.9754 - loss: 0.0679 - val_accuracy: 0.9554 - val_loss: 0.1405
Epoch 4/50
64/64 ————— 23s 357ms/step - accuracy: 0.9835 - loss: 0.0538 - val_accuracy: 0.8851 - val_loss: 0.4123
Epoch 5/50
64/64 ————— 40s 345ms/step - accuracy: 0.9849 - loss: 0.0442 - val_accuracy: 0.9022 - val_loss: 0.3187
Epoch 6/50
64/64 ————— 24s 367ms/step - accuracy: 0.9897 - loss: 0.0285 - val_accuracy: 0.9708 - val_loss: 0.1079
Epoch 7/50
64/64 ————— 22s 343ms/step - accuracy: 0.9889 - loss: 0.0282 - val_accuracy: 0.8542 - val_loss: 0.8243
Epoch 8/50
64/64 ————— 22s 342ms/step - accuracy: 0.9958 - loss: 0.0151 - val_accuracy: 0.9177 - val_loss: 0.3235
Epoch 9/50
64/64 ————— 22s 344ms/step - accuracy: 0.9914 - loss: 0.0266 - val_accuracy: 0.8696 - val_loss: 0.4955
```

**Hình 30. Minh họa việc huấn luyện mô hình DenseNet**

#### 4.4. Đánh giá các mô hình

Các mô hình được đánh giá bằng cách sử dụng chỉ số độ chính xác (accuracy) là tỷ lệ dự đoán đúng trên tổng số mẫu kiểm tra.

```
loss, accuracy = model_CNN.evaluate(test_ds)
print(f'Test loss: {loss}, Test accuracy: {accuracy}')
```

```
19/19 ————— 4s 211ms/step - accuracy: 0.9407 - loss: 0.2264
Test loss: 0.3050755560398102, Test accuracy: 0.9405267834663391
```

**Hình 31. Chỉ số độ chính xác trên tập kiểm thử của mô hình CNN**

```
loss, accuracy = model_resnet.evaluate(test_ds)
print(f'Test loss: {loss}, Test accuracy: {accuracy}')
```

19/19 ————— 9s 472ms/step - accuracy: 0.9545 - loss: 0.1206  
Test loss: 0.11202114820480347, Test accuracy: 0.9549702405929565

**Hình 32. Chỉ số độ chính xác trên tập kiểm thử của mô hình ResNet**

```
loss, accuracy = model_densenet.evaluate(test_ds)
print(f'Test loss: {loss}, Test accuracy: {accuracy}')
```

19/19 ————— 17s 942ms/step - accuracy: 0.9758 - loss: 0.2022  
Test loss: 0.14186137914657593, Test accuracy: 0.9787595868110657

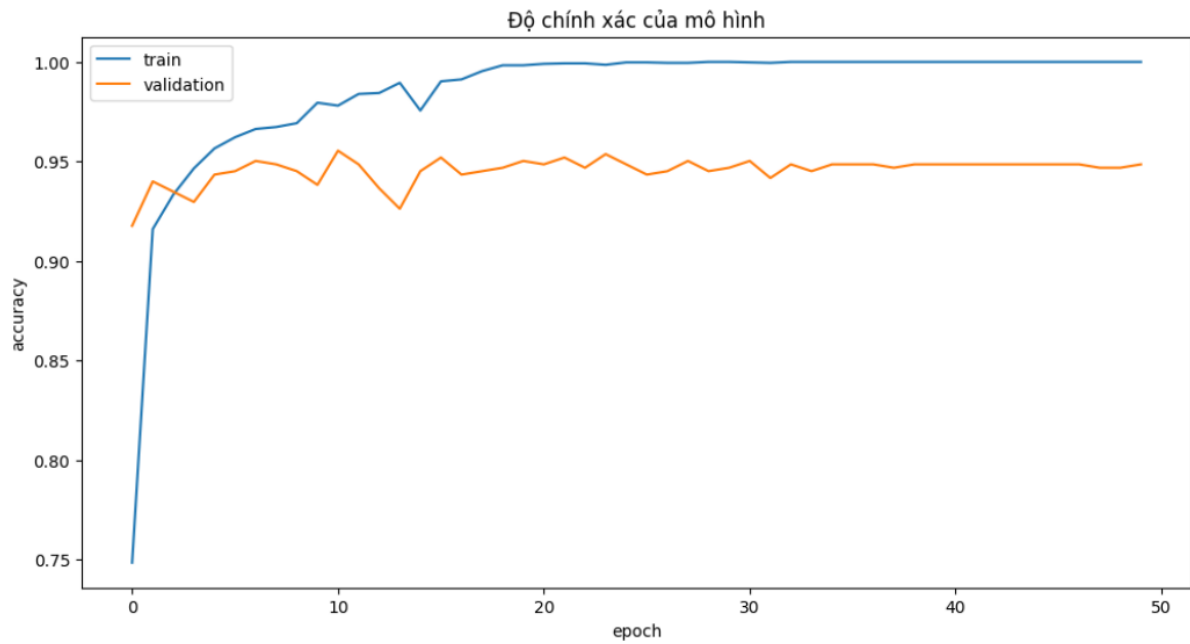
**Hình 33. Chỉ số độ chính xác trên tập kiểm thử của mô hình DenseNet**

Tiếp theo, có thể trực quan hoá các độ chính xác của các mô hình trên hai tập huấn luyện và đánh giá như trong hình 32 sau:

```
def plot_hist(history):
    plt.figure(figsize=(12, 6))
    plt.plot(history.history["accuracy"], label='train')
    plt.plot(history.history["val_accuracy"], label='validation')
    plt.title("Độ chính xác của mô hình")
    plt.ylabel("accuracy")
    plt.xlabel("epoch")
    plt.legend()
    plt.show()
```

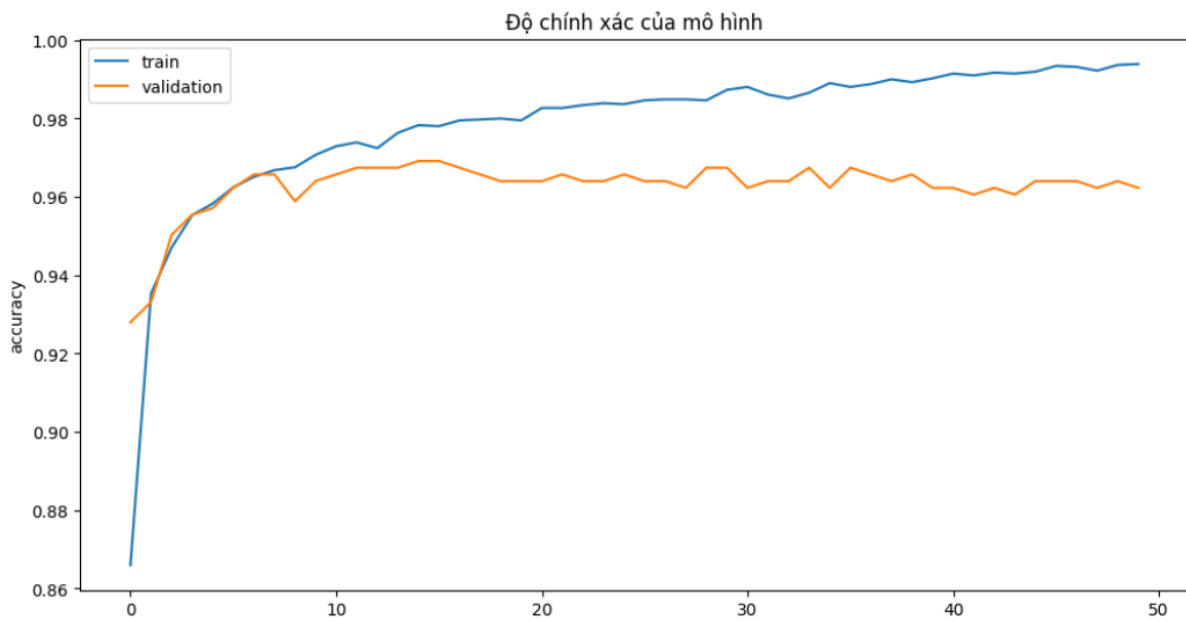
**Hình 34. Hàm vẽ biểu đồ thể hiện độ chính xác khi huấn luyện của các mô hình**

```
plot_hist(hist_CNN)
```



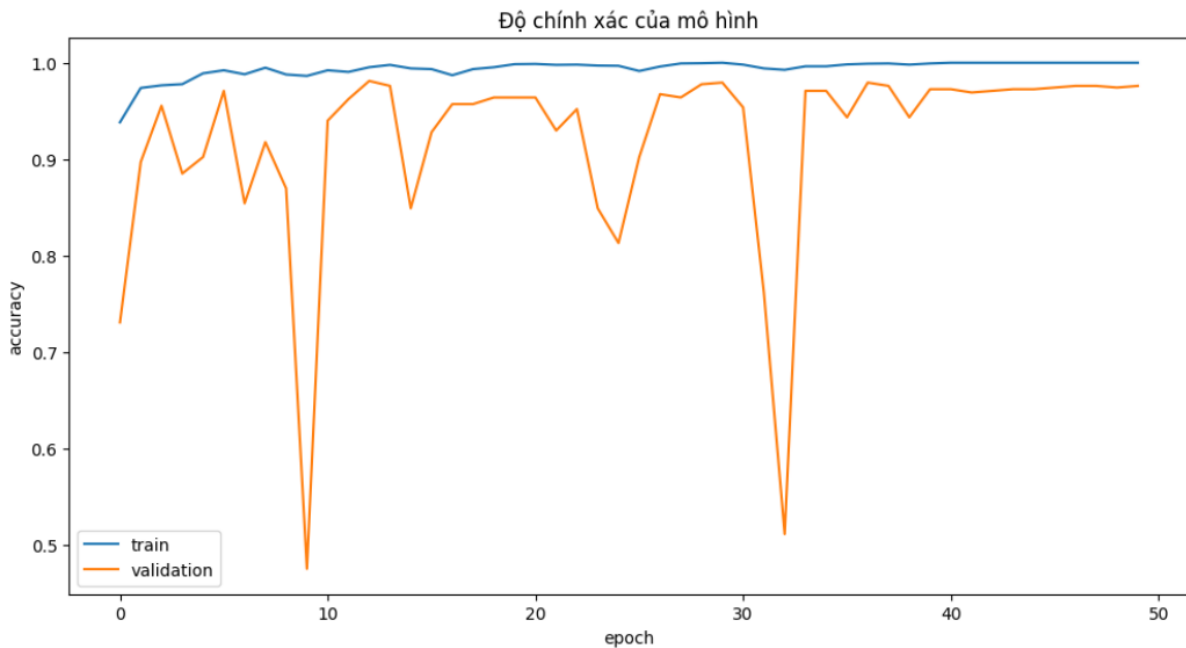
**Hình 35. Biểu đồ thể hiện độ chính xác khi huấn luyện mô hình CNN**

```
plot_hist(hist_resnet)
```



**Hình 36. Biểu đồ thể hiện độ chính xác khi huấn luyện mô hình ResNet**

```
plot_hist(hist_densenet)
```



**Hình 37. Biểu đồ thể hiện độ chính xác khi huấn luyện mô hình DenseNet**

#### 4.5. Xây dựng giao diện phát hiện bất thường trong ảnh

Để thực hiện việc kiểm thử độ hiệu quả của mô hình, khoá luận sử dụng Flask để xây dựng hệ thống đơn giản trực quan. Người dùng đưa hình ảnh lên hệ thống, sau đó hình ảnh sẽ được xử lý và dự đoán thông qua các mô hình đã được huấn luyện sẵn, và cuối cùng hệ thống sẽ đưa ra nhãn bình thường hoặc bất thường cho hình ảnh đầu vào đó.

Cấu trúc thư mục hệ thống như sau:

```
> .env  
> .vscode  
> models  
> static  
> templates  
📁 app.py
```

**Hình 38. Các thư mục của hệ thống**

- .env: thư mục môi trường ảo, chứa các thư viện và cấu hình để chạy ứng dụng
- .vscode: thiết lập cài đặt cho trình biên dịch của Python
- models: thư mục chứa các tệp mô hình đã được huấn luyện
- static: thư mục chứa các tệp tin phương tiện do người dùng tải lên hoặc để kiểm thử
- templates: thư mục chứa tệp tin giao diện dưới dạng html

Ngoài ra, tệp app.py khởi động giao diện của hệ thống và xử lý việc dự đoán nhân từ hình ảnh người dùng tải lên.


Giao diện của hệ thống có dạng:

### Nhận dạng hình ảnh

Chọn hình ảnh

Choose File

No file chosen

 Preview Image

Dự đoán

### Hình 39. Giao diện chính

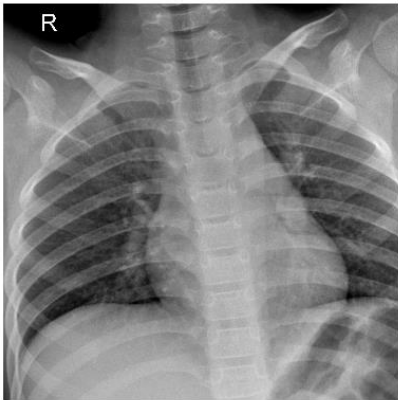
Với hình ảnh bình thường:

### Nhận dạng hình ảnh

Chọn hình ảnh

Choose File

No file chosen



NORMAL2-IM-0312-0001.jpeg

Dự đoán

Kết quả: NORMAL

### Hình 40. Kết quả hiển thị ảnh bình thường

Với hình ảnh bất thường:

## Nhận dạng hình ảnh

Chọn hình ảnh

Choose File No file chosen



person15\_virus\_46.jpeg

Dự đoán

Kết quả: PNEUMONIA

**Hình 41. Kết quả hiển thị ảnh bất thường**

### 4.6. Nhận xét

Kết quả đánh giá cho thấy mô hình đạt độ chính xác cao, chứng minh tính hiệu quả của mô hình trong việc phát hiện bất thường trong ảnh y khoa.

CNN là mô hình đơn giản hơn, nhưng không đủ mạnh để cạnh tranh với các mô hình sâu hơn như ResNet và DenseNet. DenseNet có khả năng tái sử dụng các đặc trưng từ các lớp trước, giúp cải thiện độ chính xác, nhưng gặp vấn đề lớn với hiện tượng quá khớp. Mô hình ResNet thể hiện tính ổn định và hiệu quả cao hơn so với hai mô hình còn lại, với độ chính xác của tập huấn luyện và tập đánh giá gần như không có sự khác biệt lớn.

Tuy nhiên, các mô hình trên vẫn còn một số hạn chế nhất định, nhất là hiện tượng quá khớp xảy ra đáng kể trên mô hình DenseNet. Điều này khiến cho chúng gặp trở ngại lớn khi tổng quát hoá trên tập dữ liệu đánh giá, gây khó khăn cho việc phát hiện hình ảnh bất thường nếu có hình ảnh mới đưa vào. Cần xem xét và cải tiến cho các mô hình trên để cải thiện hiệu suất và giảm thiểu hiện tượng quá khớp.

### 4.7. Hướng dẫn sử dụng ứng dụng

**Bước 1:** Cài Python phiên bản 3.9.11

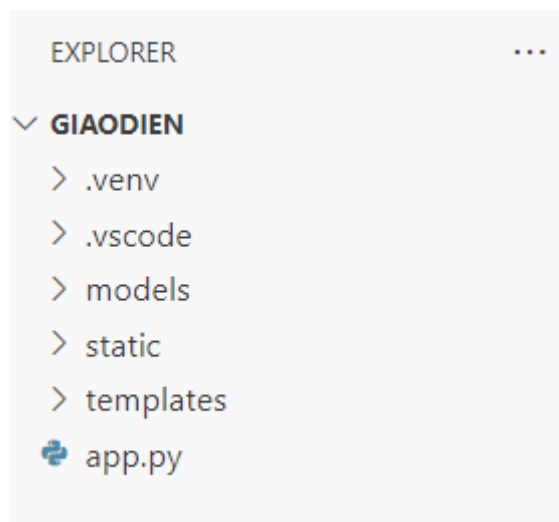
Tải về tại: <https://www.python.org/downloads/release/python-3911/>



Files					
Version	Operating System	Description	MD5 Sum	File Size	GPG
<a href="#">Gzipped source tarball</a>	Source release		daca49063ced330eb933a0fb437dee50	25.1 MB	<a href="#">SIG</a>
<a href="#">XZ compressed source tarball</a>	Source release		3c8dde3ebd6da005e969b83b5f0c1886	18.8 MB	<a href="#">SIG</a>
<a href="#">macOS 64-bit Intel-only installer</a>	macOS	for macOS 10.9 and later, deprecated	99e519a1e8387f692da6c5a0e6177243	29.5 MB	<a href="#">SIG</a>
<a href="#">macOS 64-bit universal2 installer</a>	macOS	for macOS 10.9 and later	655c5ca728aafc5f64f74b94397593ad	37.0 MB	<a href="#">SIG</a>
<a href="#">Windows installer (64-bit)</a>	Windows	Recommended	fef52176a572efd48b7148f006b25801	27.8 MB	<a href="#">SIG</a>
<a href="#">Windows installer (32-bit)</a>	Windows		4210652b14a030517046cdf111c09c1e	26.7 MB	<a href="#">SIG</a>
<a href="#">Windows help file</a>	Windows		dbc3c10a40ebccc1d0b47616a2d4503f	8.5 MB	<a href="#">SIG</a>
<a href="#">Windows embeddable package (64-bit)</a>	Windows		9bc1e9dd44c1c1c68838e5b4ce9f2248	8.1 MB	<a href="#">SIG</a>
<a href="#">Windows embeddable package (32-bit)</a>	Windows		b5c293e11564f11cb23f9c4e4e97cbf8	7.3 MB	<a href="#">SIG</a>

**Hình 42. Vị trí tệp Python cần tải**

**Bước 2:** Mở thư mục chứa ứng dụng lên bằng ứng dụng Visual Studio Code (hoặc Command Prompt)



**Hình 43. Cấu trúc thư mục của ứng dụng**

**Bước 3:** Mở cửa sổ dòng lệnh (terminal) lên, gõ lệnh:

`.\.venv\Scripts\activate`

Lệnh này để kích hoạt môi trường ảo nhằm cài đặt các thư viện riêng biệt tránh xung đột.

```
PS C:\Users\ADMIN\KhoaLuanTotNghiep\GiaoDien> .\.venv\Scripts\activate
(.venv) PS C:\Users\ADMIN\KhoaLuanTotNghiep\GiaoDien>
```

**Hình 44. Kích hoạt môi trường ảo**

**Bước 4:** Chạy ứng dụng bằng câu lệnh:

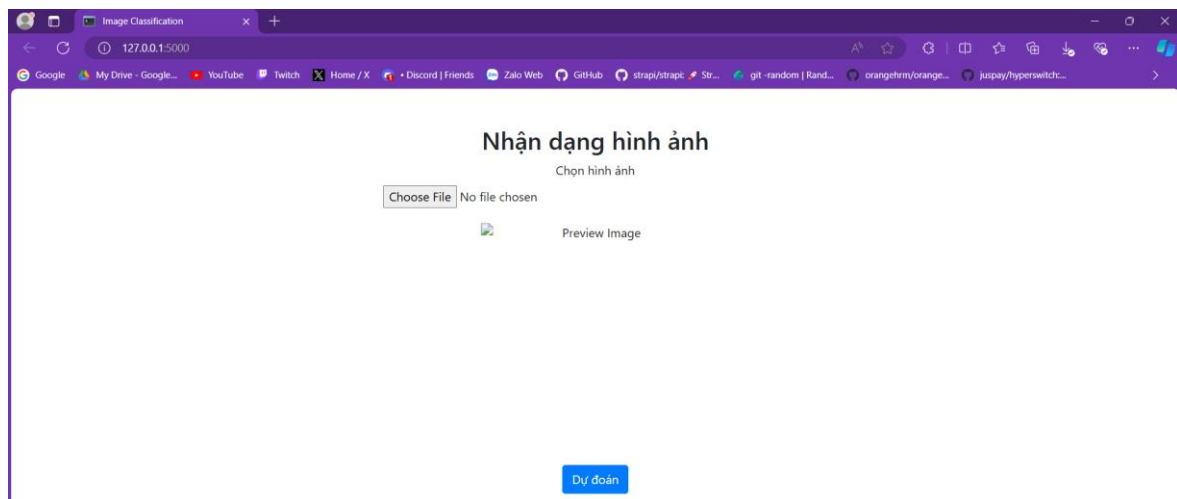
`python app.py`

Khi đó màn hình cửa sổ dòng lệnh hiển thị:

```
* Serving Flask app 'app'
* Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
```

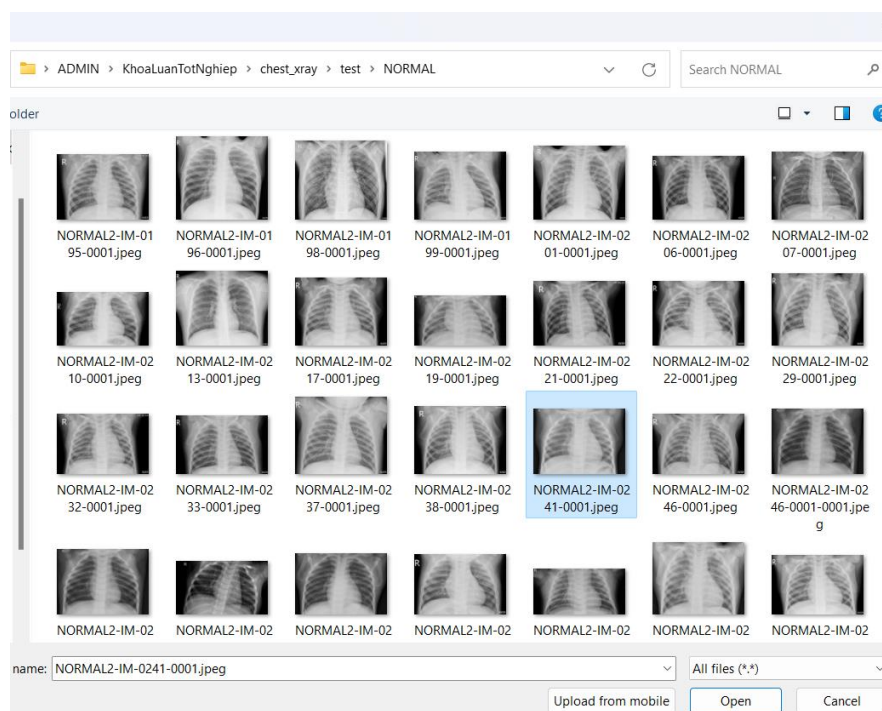
**Hình 45. Thông báo chạy giao diện ứng dụng**

**Bước 5:** Mở trình duyệt web lên, sao chép địa chỉ `http://127.0.0.1:5000` vào thanh địa chỉ và nhấn Enter (Hoặc giữ phím Ctrl và nhấn chuột trái trực tiếp vào địa chỉ đó trong cửa sổ dòng lệnh). Màn hình trình duyệt có dạng:



**Hình 46. Giao diện ứng dụng trên trình duyệt**

**Bước 6:** Nhấn chuột trái vào nút Choose File và chọn tệp hình ảnh tải lên



**Hình 47. Chọn tệp hình ảnh cần thử**

Kết quả sau khi chọn:

## Nhận dạng hình ảnh

Chọn hình ảnh

Choose File

NORMAL2-IM-0241-0001.jpeg



NORMAL2-IM-0241-0001.jpeg

Dự đoán

### Hình 48. Giao diện sau khi chọn

**Bước 7:** Sau khi hình ảnh được tải lên, nhấn chuột vào nút “Dự đoán”. Kết quả dự đoán sẽ được hiển thị trên màn hình

## Nhận dạng hình ảnh

Chọn hình ảnh

Choose File

No file chosen



NORMAL2-IM-0241-0001.jpeg

Dự đoán

**Kết quả: NORMAL**

### Hình 49. Kết quả nhận dạng “bình thường”

**Bước 8:** Lặp lại các bước 6, 7 nếu muốn thử với các hình khác.

## **CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

### **5.1. Kết luận**

Khóa luận đã thành công trong việc phát triển và ứng dụng các mô hình học sâu để phát hiện bất thường trong ảnh y khoa. Các mô hình đã được huấn luyện và đánh giá trên bộ dữ liệu cho thấy hiệu suất cao. Kết quả này chứng tỏ rằng các mô hình học sâu được đề xuất có khả năng phân loại chính xác các ảnh bình thường và bất thường.

### **5.2. Hướng phát triển**

Để cải thiện và mở rộng các kết quả đã đạt được, các hướng phát triển có thể được xem xét như:

- Nghiên cứu và tích hợp thêm các mô hình học tăng cường để cải thiện khả năng tự học và hiệu suất của mô hình nếu đối mặt với dữ liệu chưa từng thấy trước đó.
- Mở rộng nghiên cứu sang các loại dữ liệu hình ảnh y khoa khác như hình ảnh MRI hoặc siêu âm để kiểm tra và nâng cao khả năng tổng quát hóa và độ tin cậy của mô hình.
- Xây dựng và phát triển một hệ thống cho phép kiểm tra các ảnh y khoa một cách tiện lợi, và bổ sung các tính năng mới cho hệ thống, như chẩn đoán tự động và phân tích xu hướng theo thời gian, để hỗ trợ tốt hơn cho các quyết định lâm sàng.

## DANH MỤC TÀI LIỆU THAM KHẢO

- [1] Y. B. Y. & H. G. LeCun, “Deep Learning,” *Nature*, tập 521, số 7553, pp. 436-444, 2015.
- [2] G. E. O. S. & T. Y. W. Hinton, “A fast learning algorithm for deep belief nets,” *Neural Computation*, tập 18, số 7, pp. 1527-1554, 2006.
- [3] A. M. A.-r. & H. G. Graves, “Speech recognition with deep recurrent neural networks,” trong *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.
- [4] A. S. I. & H. G. E. Krizhevsky, “Imagenet classification with deep convolutional neural networks,” trong *Advances in Neural Information Processing Systems*, 2012.
- [5] “CS231n Convolutional Neural Networks for Visual Recognition,” [Trực tuyến]. Available: <https://cs231n.github.io/neural-networks-1/>. [Đã truy cập 10 June 2024].
- [6] I. Sarker, “Deep Learning: a Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions,” *SN Computer Science*, tập 2, số 6, 2021.
- [7] G. X. a. B. Y., “Understanding the difficulty of training deep feedforward neural networks,” trong *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.
- [8] R. K. R. K. O. a. M. A. Bouthina, “Palm Tree Diseases Detection Using Deep Learning: A Short Review,” pp. 1-8, 2024.
- [9] C. Olah, “Understanding LSTM Networks,” [Trực tuyến]. Available: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. [Đã truy cập 30 May 2024].
- [10] R. M. Schmidt, “Recurrent Neural Networks (RNNs): A gentle Introduction and Overview,” 2019. [Trực tuyến]. Available: <https://arxiv.org/abs/1912.05911>. [Đã truy cập 14 June 2024].
- [11] T. P. R. C. R. Z. S. L. Y. a. C. R.-C. Toharudin, “Employing Long Short-Term Memory and Facebook Prophet Model in Air Temperature Forecasting,” *Communication in Statistics- Simulation and Computation*, 2021.

- [12] B. e. a. Zong, “Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection,” 2018. [Trực tuyến]. Available: <https://openreview.net/forum?id=BJJLHbb0->. [Đã truy cập 19 June 2024].
- [13] Y. H. S. a. C. Y.-G. Song, “Analysis of Autoencoders for Network Intrusion Detection,” *Sensors*, tập 21, p. 4294, 2021.
- [14] I. Developer, “Generative Adversarial Networks Explained,” 2021. [Trực tuyến]. Available: <https://developer.ibm.com/articles/generative-adversarial-networks-explained/>. [Đã truy cập 10 June 2024].
- [15] J. G. Z. a. R. N. Brown, “CNN Based Image Classification of Malicious UAVs,” *Applied Sciences*, tập 13, p. 240, 2022.
- [16] A. S. F. R. P. a. S. G. Mazza, “TanDEM-X Forest Mapping using Convolutional Neural Networks,” *Remote Sensing*, tập 11, 2019.
- [17] H. a. W. L. J. Abdi, “Principal component analysis,” *Wiley Interdisciplinary Reviews: Computational Statistics*, tập 2, số 4, p. 433–459, 2010.
- [18] O. M. P. a. M. D. Rippel, “Modeling the Distribution of Normal Data in Pre-Trained Deep Features for Anomaly Detection,” January 2021. [Trực tuyến]. Available: <https://ieeexplore.ieee.org/abstract/document/9412109>. [Đã truy cập 17 June 2024].
- [19] D. Z. K. a. G. M. Kermany, *Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification*, Mendeley Data, 2018.