

**TRƯỜNG ĐẠI HỌC THỦY LỢI**  
**PHÂN HIỆU**



**ĐỒ ÁN MÔN HỌC: KHAI PHÁ DỮ LIỆU**

**ĐỀ TÀI:**

**PHÂN LOẠI ẢNH X-QUANG NHẬN BIẾT**  
**VIÊM PHỔI**

**Sinh viên thực hiện:**

Nguyễn Văn Phong – MSSV: 2251068230

Trần Đình Khải – MSSV: 2251068197

**Lớp: S25-K64CNTT**

**Giảng viên hướng dẫn: ThS. Vũ Thị Hạnh**

**TP HỒ CHÍ MINH 2025**

# MỤC LỤC

MỤC LỤC .....	2
DANH MỤC HÌNH ẢNH.....	5
CHƯƠNG 1. MỞ ĐẦU.....	7
1.1. Bối cảnh và tầm quan trọng.....	7
1.2. Vấn đề cần giải quyết.....	7
1.3. Mục tiêu của dự án.....	8
1.4. Cấu trúc báo cáo .....	9
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	10
2.1. Học sâu trong Chẩn đoán đoán hình ảnh y tế.....	10
2.1.1. Mạng Nơ-ron Tích chập (Convolutional Neural Network - CNN).....	10
2.1.2. Học chuyển giao (Transfer learning).....	10
2.2. Trí tuệ Nhân tạo có thể Diễn giải (Explainable AI - XAI).....	11
2.3. Đặc điểm X-quang Lồng ngực ở Bệnh nhân Nhi.....	12
2.3.1. Hình ảnh Bình thường và Các "Cạm bẫy" Chẩn đoán.....	12
2.3.2. Các dấu hiệu bệnh lý thường gặp.....	13
CHƯƠNG 3. PHƯƠNG PHÁP VÀ HỆ THỐNG THỰC NGHIỆM.....	15
3.1. Dữ liệu và phân tích khám phá.....	15
3.1.1. Mô tả bộ dữ liệu.....	15
3.1.2. Phân tích khám phá dữ liệu (EDA).....	18
3.2. Tiền xử lý và tăng cường dữ liệu.....	24
3.2.1. Tiền xử lý dữ liệu.....	24
3.2.2. Tăng cường dữ liệu.....	27
3.3. Thử nghiệm 1 – Huấn luyện mô hình CNN cơ bản với tập dữ liệu gốc.....	27
3.3.1. Mục tiêu.....	27
3.3.2. Cấu trúc mô hình.....	27
3.3.3. Cấu hình huấn luyện.....	29

3.3.4. Kết quả huấn luyện.....	29
3.3.5. Nhận xét và phân tích.....	30
3.3.6. Kết quả thử nghiệm.....	30
3.4. Thử nghiệm 2 - Huấn luyện mô hình CNN cơ bản với dữ liệu được tái cấu trúc..	31
3.4.1. Mục tiêu.....	31
3.4.2. Tái cấu trúc dữ liệu.....	31
3.4.3. Cấu trúc mô hình CNN.....	31
3.4.4. Cấu hình huấn luyện.....	33
3.4.5. Kết quả huấn luyện.....	33
3.4.6. Đánh giá trên tập kiểm tra.....	34
3.4.7. Nhận xét và phân tích.....	36
3.4.8. Kết quả thử nghiệm.....	36
3.5. Thử nghiệm 3 – Huấn luyện mô hình Transfer learning với DenseNet121 .....	37
3.5.1. Mục tiêu.....	37
3.5.2. Chuẩn bị dữ liệu.....	37
3.5.3. Cấu trúc mô hình.....	38
3.5.4. Giai đoạn 1 – Feature Extraction.....	39
3.5.5. Giai đoạn 2 Fine – tuning.....	40
3.5.6. Kết quả đánh giá trên tập test.....	41
3.5.7. Phân tích và nhận xét.....	43
3.5.8. Kết luận thử nghiệm.....	43
3.6. Huấn luyện với mô hình transfer learning ResNet50.....	44
3.6.1. Tiền xử lý dữ liệu.....	44
3.6.2. Xây dựng mô hình ResNet50.....	46
3.6.3. Huấn luyện mô hình (Feature Extraction).....	47
3.6.4. Fine – tuning.....	48
3.6.5. Đánh giá mô hình trên tập kiểm thử.....	49
3.6.6. Nhận xét kết quả.....	51
3.7. Huấn luyện mô hình EfficientNetB3 (Transfer Learning) .....	52
3.7.1. Tiền xử lý dữ liệu.....	52

3.7.2. Xây dựng mô hình EfficientNetB3.....	54
3.7.3. Giai đoạn 1 – Huấn luyện phân đầu ra (Feature Extraction).....	55
3.7.4. Giai đoạn 2 – Fine-tuning mô hình.....	56
3.7.5. Đánh giá mô hình trên tập kiểm thử.....	57
3.7.6. Nhận xét kết quả.....	58
3.8. Kết luận chung.....	59
3.8.1. Kết quả và so sánh giữa các mô hình.....	59
3.8.2. Kết luận chương và lựa chọn mô hình tối ưu.....	60
CHƯƠNG 4. XÂY DỰNG ỨNG DỤNG WEB ĐỂ HỖ TRỢ CHẨN ĐOÁN VIÊM PHỔI.....	62
4.1. Mục tiêu.....	62
4.2. Kiến trúc tổng thể của hệ thống.....	62
4.3. Thiết kế và xây dựng Frontend.....	62
4.4. Backend.....	64
4.5. Kết quả chạy thử nghiệm.....	64
CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	66
5.1. Kết luận.....	66
5.2. Hạn chế.....	66
5.3. Hướng phát triển.....	66
TÀI LIỆU THAM KHẢO.....	67

## DANH MỤC HÌNH ẢNH

Hình 1. Hình ảnh X-quang ngực bình thường của trẻ em với "dấu hiệu cánh buồm" của tuyến ức.....	8
Hình 2: Sơ đồ so sánh mô hình AI "hộp đen" và mô hình AI có thể giải thích bằng Grad-CAM.....	9
Hình 3: Ảnh chụp màn hình giao diện chính của ứng dụng web phân tích X-quang.....	10
Hình 4: Sơ đồ kiến trúc một mạng CNN cơ bản, minh họa các lớp Conv, Pool, và FC ...	11
Hình 5: Ảnh minh họa cơ chế của Grad-CAM, cho thấy ảnh gốc và ảnh đầu ra với heatmap.....	12
Hình 6: Ảnh X-quang điển hình có "dấu hiệu cánh buồm" (sail sign) của tuyến ức.....	13
Hình 7: Ví dụ viêm thùy trên phổi phải với dấu hiệu xóa bờ ở rãnh liên thùy nhỏ.....	14
Hình 8: Hình ảnh xẹp thùy trên phổi trái, khí quản bị kéo lệch về bên trái.....	15
Hình 9: Dấu hiệu góc sườn hoành tù do tràn dịch màng phổi.....	15
Hình 10: Ảnh mẫu thuộc lớp NORMAL (phổi bình thường).....	18
Hình 11: Ảnh mẫu thuộc lớp PNEUMONIA (phổi bị viêm phổi).....	19
Hình 12: Ảnh minh họa: Chế độ GPU và thông tin CUDA.....	21
Hình 13: Ảnh minh họa: Cấu trúc cây thư mục của bộ dữ liệu.....	22
Hình 14: Số lượng ảnh NORMAL và PNEUMONIA.....	23
Hình 15: Biểu đồ cột phân bố số lượng ảnh theo lớp trong từng tập (Train/Test/Validation).....	24
Hình 16: 5 ảnh NORMAL mẫu.....	25
Hình 17: 5 ảnh PNEUMONIA mẫu.....	25
Hình 18: : Biểu đồ Training/Validation Accuracy và Loss qua 10 epoch.....	35
Hình 19: Ma trận nhầm lẫn trên tập test.....	36
Hình 20: Classification Report – Precision, Recall, F1-score cho từng lớp.....	37
Hình 21: Biểu đồ Accuracy & Loss giai đoạn 1 – Feature Extraction.....	41
Hình 22: Biểu đồ Accuracy & Loss giai đoạn 2 – Fine-tuning.....	42
Hình 23: Ma trận nhầm lẫn trên tập test.....	43

Hình 24: Báo cáo phân loại chi tiết .....	43
Hình 25: Đồ thị Accuracy và Loss qua các epoch.....	49
Hình 26: Đồ thị Accuracy và Loss qua các epoch.....	50
Hình 27: Ma trận nhầm lẫn trên tập tesy .....	51
Hình 28: Báo cáo phân loại chi tiết .....	52
Hình 29: Đồ thị Accuracy và Loss trong giai đoạn Feature Extraction.....	56
Hình 30: Quá trình Fine-tuning EfficientNetB3 .....	58
Hình 31: Ma trận nhầm lẫn trên tập tets .....	59
Hình 32: Giao diện người dùng.....	64
Hình 33: Response trả về kết quả .....	65
Hình 34: Kết quả chạy thử nghiệm.....	66

# CHƯƠNG 1. MỞ ĐẦU

## 1.1. Bối cảnh và tầm quan trọng

Viêm phổi là một trong những bệnh lý nhiễm trùng phổ biến và nguy hiểm hàng đầu ở trẻ em, đặc biệt là nhóm tuổi từ 0-5. Trong bối cảnh này, X-quang ngực đóng vai trò là công cụ chẩn đoán hình ảnh cốt lõi, giúp các bác sĩ đưa ra quyết định điều trị kịp thời. Tuy nhiên, việc diễn giải phim X-quang nhi khoa đòi hỏi kinh nghiệm chuyên sâu do những khác biệt đáng kể về giải phẫu và kỹ thuật so với người lớn. Sự phát triển của Trí tuệ Nhân tạo (AI), đặc biệt là học sâu, mở ra tiềm năng to lớn trong việc xây dựng các hệ thống hỗ trợ, giúp nâng cao độ chính xác và giảm thiểu sai sót trong chẩn đoán.

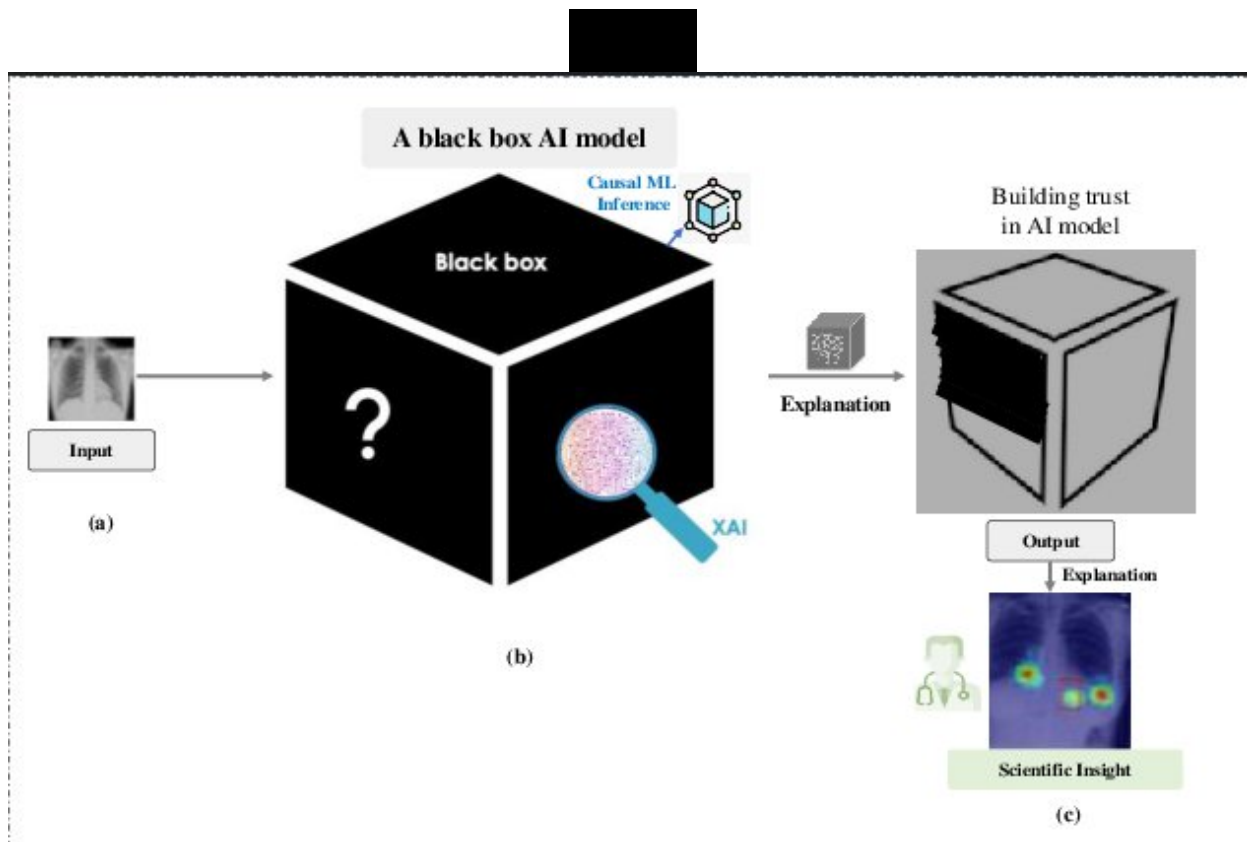
## 1.2. Vấn đề cần giải quyết

Việc áp dụng AI vào X-quang nhi khoa đối mặt với hai thách thức chính. Thứ nhất, về mặt chuyên môn, X-quang ngực trẻ em chứa đựng nhiều "cạm bẫy chẩn đoán". Cấu trúc tuyến ức lớn ở trẻ nhỏ có thể tạo ra bóng mờ dễ bị nhầm lẫn với viêm phổi. Việc nhận diện sai lầm này có thể dẫn đến chẩn đoán nhầm và điều trị không cần thiết, gây lo lắng cho gia đình.



*Hình 1. Hình ảnh X-quang ngực bình thường của trẻ em với "dấu hiệu cánh bướm" của tuyến ức*

**Thứ hai, về mặt công nghệ,** hầu hết các mô hình AI hoạt động như một "hộp đen", chỉ đưa ra kết quả cuối cùng mà không giải thích lý do. Điều này tạo ra rào cản lớn trong việc ứng dụng lâm sàng, vì bác sĩ cần phải hiểu và tin tưởng vào cơ sở quyết định của máy tính. Một dự đoán đơn thuần là không đủ; bác sĩ cần biết AI đã "nhìn" vào đâu để đưa ra kết luận đó.



Hình 2: Sơ đồ so sánh mô hình AI "hộp đen" và mô hình AI có thể giải thích bằng Grad-CAM

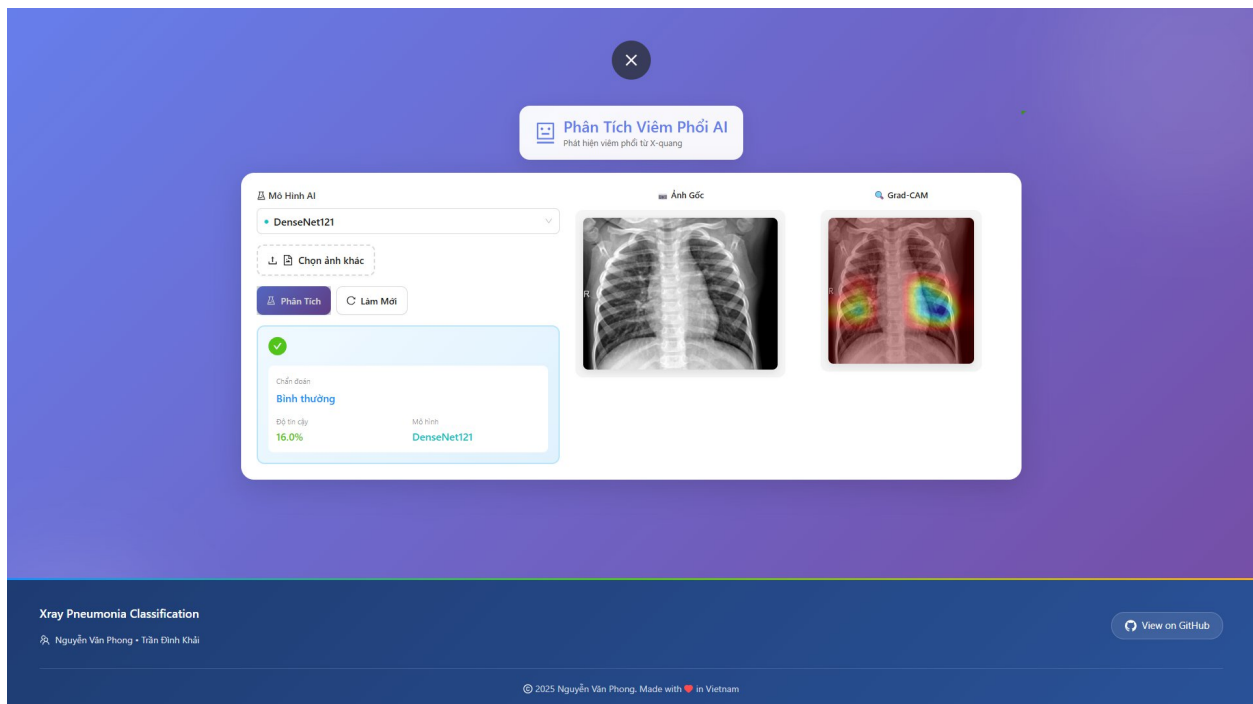
### 1.3. Mục tiêu của dự án

Để giải quyết các vấn đề trên, dự án này được thực hiện với các mục tiêu cụ thể sau:

1. **Xây dựng và đánh giá** các mô hình học sâu để phân loại chính xác ảnh X-quang viêm phổi ở trẻ em.



2. **Tích hợp kỹ thuật diễn giải AI (XAI)**, cụ thể là Grad-CAM, để cung cấp bằng chứng trực quan, làm sáng tỏ các vùng trên ảnh mà mô hình tập trung vào khi đưa ra dự đoán.
3. **Phát triển một ứng dụng web hoàn chỉnh**, cho phép người dùng tải ảnh X-quang lên và nhận về cả kết quả phân loại lẫn hình ảnh giải thích từ Grad-CAM một cách trực quan và dễ hiểu.



Hình 3: Ảnh chụp màn hình giao diện chính của ứng dụng web phân tích X-quang

#### 1.4. Cấu trúc báo cáo

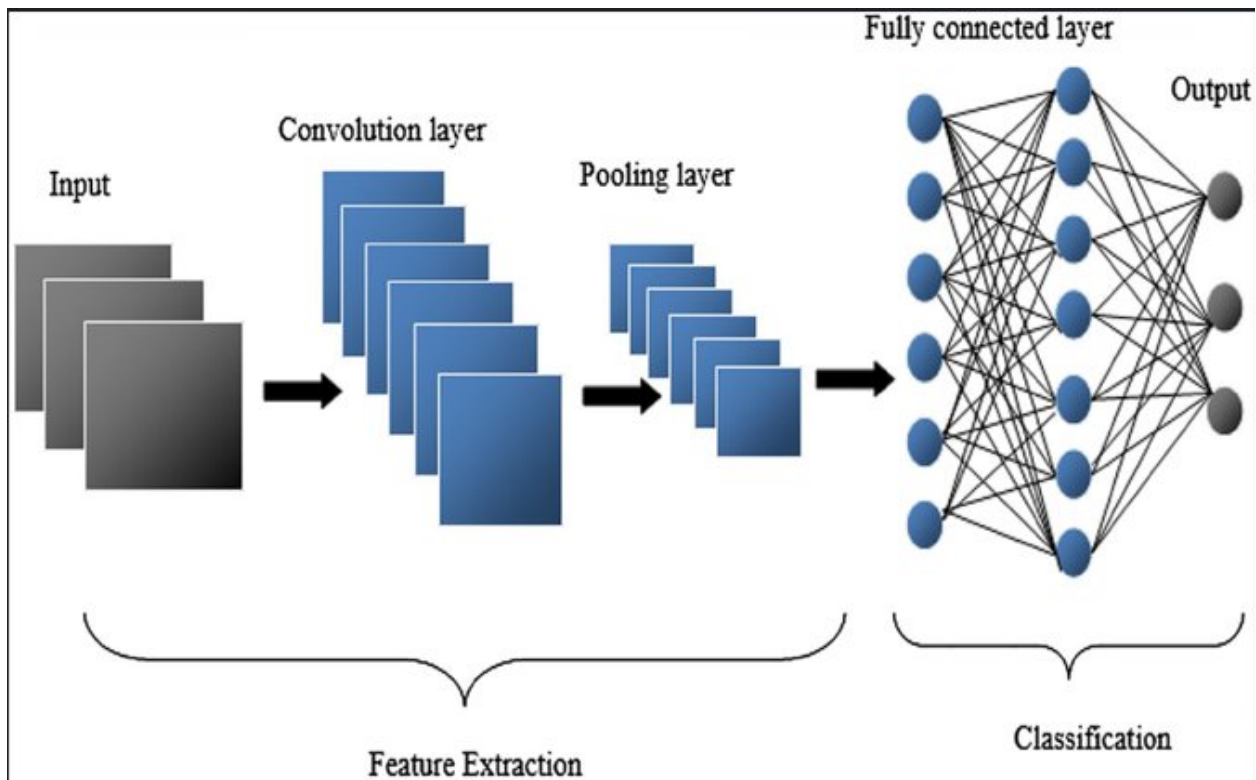
Báo cáo được trình bày qua 5 chương: **Chương 1** giới thiệu tổng quan về dự án. **Chương 2** trình bày các cơ sở lý thuyết về học sâu, kỹ thuật Grad-CAM và các đặc điểm X-quang nhi khoa. **Chương 3** mô tả phương pháp thực hiện, từ xử lý dữ liệu đến xây dựng ứng dụng web. **Chương 4** phân tích các kết quả đạt được. Cuối cùng, **Chương 5** đưa ra kết luận và đề xuất các hướng phát triển trong tương lai.

## CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

### 2.1. Học sâu trong Chẩn đoán đoán hình ảnh y tế

#### 2.1.1. Mạng Nơ-ron Tích chập (Convolutional Neural Network - CNN)

CNN là kiến trúc nền tảng cho hầu hết các bài toán phân tích hình ảnh. Mô hình này mô phỏng hệ thống thị giác của con người, sử dụng các lớp đặc biệt như lớp Tích chập (Convolutional Layer) để nhận diện các đặc trưng từ đơn giản (như cạnh, góc) đến phức tạp (như hình dạng tổng thể), và lớp Gộp (Pooling Layer) để giảm kích thước dữ liệu và tăng hiệu quả tính toán.



Hình 4: Sơ đồ kiến trúc một mạng CNN cơ bản, minh họa các lớp Conv, Pool, và FC

#### 2.1.2. Học chuyển giao (Transfer learning)

Huấn luyện một mô hình CNN từ đầu đòi hỏi một lượng dữ liệu khổng lồ và năng lực tính toán rất lớn. Học chuyển giao là một giải pháp hiệu quả, cho phép chúng ta tận dụng "kiến thức" từ một mô hình đã được huấn luyện trước trên một bộ dữ liệu lớn (như

ImageNet). Trong dự án này, chúng tôi đã sử dụng các kiến trúc tiên tiến như **DenseNet121** và **EfficientNetB3**, vốn đã học được cách nhận diện hàng ngàn loại đặc trưng hình ảnh khác nhau, và tinh chỉnh chúng cho nhiệm vụ cụ thể là phân loại ảnh X-quang.

## 2.2. Trí tuệ Nhân tạo có thể Diễn giải (Explainable AI - XAI)

**Grad-CAM (Gradient-weighted Class Activation Mapping):** Để AI được tin tưởng và ứng dụng trong y khoa, việc giải thích được quyết định của nó là yêu cầu bắt buộc. Grad-CAM là một kỹ thuật XAI phổ biến, cho phép chúng ta hình dung được các vùng trên ảnh đầu vào mà mô hình AI "chú ý" nhất khi đưa ra một dự đoán cụ thể. Grad-CAM tạo ra một "bản đồ nhiệt" (heatmap), trong đó các vùng màu nóng (đỏ, vàng) tương ứng với những khu vực có ảnh hưởng lớn nhất đến quyết định của mô hình. Trong bối cảnh y tế, bản đồ nhiệt này có thể được xem như "bằng chứng" trực quan, giúp bác sĩ xác thực xem AI có đang tập trung vào đúng vùng bệnh lý hay không.



*Hình 5: Ảnh minh họa cơ chế của Grad-CAM, cho thấy ảnh gốc và ảnh đầu ra với heatmap*

### 2.3. Đặc điểm X-quang Lồng ngực ở Bệnh nhân Nhi

Diễn giải X-quang ngực ở trẻ em từ 0-5 tuổi đòi hỏi kiến thức chuyên biệt do có nhiều điểm khác biệt so với người lớn. Việc nắm vững các đặc điểm này là nền tảng để xây dựng một mô hình AI chính xác và tránh các sai sót chẩn đoán.

#### 2.3.1. Hình ảnh Bình thường và Các "Cạm bẫy" Chẩn đoán

**Kỹ thuật chụp:** Do trẻ nhỏ không thể đứng yên, tư thế chụp phổ biến là **nằm ngửa (AP)**, khác với tư thế đứng (PA) ở người lớn. Tư thế này có thể làm bóng tim và các cấu trúc trung thất bị phóng đại nhẹ, một yếu tố mà mô hình AI cần phải học để không diễn giải sai.

**Tuyến ức:** Đây là một trong những thách thức lớn nhất trong X-quang nhi. Ở trẻ nhỏ, tuyến ức là một khối mô mềm lớn ở trung thất trước, tạo ra bóng mờ dễ bị nhầm với viêm phổi hoặc khối u. Hình ảnh đặc trưng nhất là "**dấu hiệu cánh buồm**" (**sail sign**), thường thấy ở bên phải. Một mô hình AI không được "dạy" về đặc điểm này rất có khả năng sẽ phân loại nhầm một lồng ngực khỏe mạnh thành có bệnh lý.



*Hình 6: Ảnh X-quang điển hình có "dấu hiệu cánh buồm" (sail sign) của tuyến ức*

### 2.3.2. Các dấu hiệu bệnh lý thường gặp

**Viêm Phổi (Pneumonia):** Bệnh lý này biểu hiện qua nhiều dấu hiệu, trong đó quan trọng nhất là **Tăng đậm độ** (vùng phổi bị viêm có màu trắng hơn) và **Dấu hiệu xóa bờ (Silhouette Sign)**. Dấu hiệu này xảy ra khi một vùng phổi bị đông đặc tiếp xúc với các cấu trúc có cùng đậm độ (như bờ tim), làm mất đi ranh giới sắc nét giữa chúng.



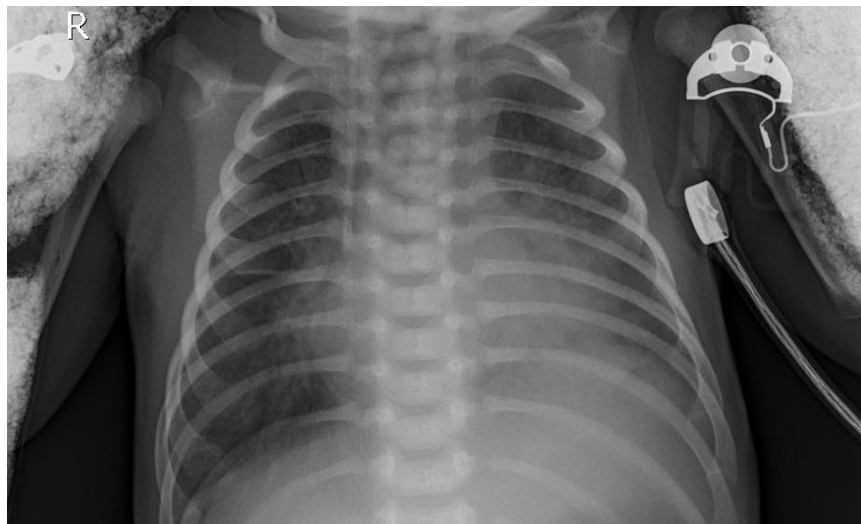
*Hình 7: Ví dụ viêm thùy trên phổi phải với dấu hiệu xóa bờ ở rãnh liên thùy nhỏ*

**Xẹp Phổi (Atelectasis):** Khác với viêm phổi thường gây hiệu ứng choán chỗ, xẹp phổi được đặc trưng bởi **dấu hiệu mất thể tích**. Điều này biểu hiện bằng việc các cấu trúc lân cận như khí quản hoặc rốn phổi bị **kéo lệch về phía** vùng phổi bị xẹp.



*Hình 8: Hình ảnh xẹp thùy trên phổi trái, khí quản bị kéo lệch về bên trái*

**Tràn dịch Màng phổi (Pleural Effusion):** Dấu hiệu kinh điển là sự thay đổi ở góc sườn hoành. Bình thường, góc này phải nhọn và sắc nét. Khi có dịch, nó sẽ lấp đầy và làm cho góc này trở nên **tù hoặc bị xóa mờ**.



*Hình 9: Dấu hiệu góc sườn hoành tù do tràn dịch màng phổi*

## CHƯƠNG 3. PHƯƠNG PHÁP VÀ HỆ THỐNG THỰC NGHIỆM

### 3.1. Dữ liệu và phân tích khám phá

#### 3.1.1. Mô tả bộ dữ liệu

##### Nguồn dữ liệu:

Bộ dữ liệu được sử dụng trong đề tài là "**Chest X-Ray Images (Pneumonia)**" do Paul Mooney công bố trên nền tảng [Kaggle](https://www.kaggle.com/paultattn/chest-xray-pneumonia). Bộ dữ liệu này được phát triển nhằm hỗ trợ nghiên cứu về **phát hiện bệnh viêm phổi (Pneumonia)** thông qua ảnh X-quang lồng ngực.

##### Mục tiêu:

Dữ liệu được thiết kế cho bài toán **phân loại nhị phân (binary classification)**, trong đó:

**NORMAL:** Ảnh X-quang phổi bình thường.

**PNEUMONIA:** Ảnh X-quang phổi của bệnh nhân bị viêm phổi (gồm cả do vi khuẩn và virus).

##### Thành phần của bộ dữ liệu:

Bộ dữ liệu được chia sẵn thành ba tập phục vụ cho quá trình huấn luyện và đánh giá mô hình:

**Tập huấn luyện (train):** 5216 ảnh

**Tập kiểm tra (test):** 624 ảnh

**Tập xác thực (val):** 16 ảnh

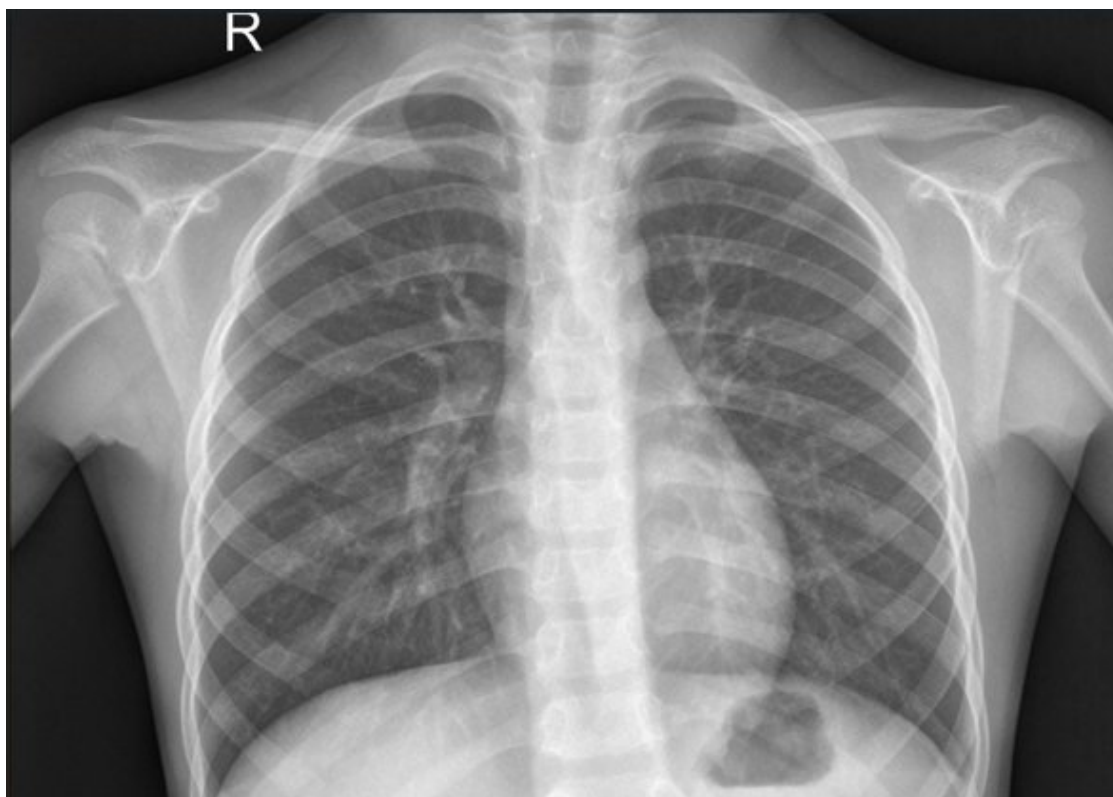
### **Tập dữ liệu NORMAL PNEUMONIA Tổng cộng**

Train	1341	3875	5216
Test	234	390	624
Val	8	8	16
<b>Tổng cộng</b>	<b>1583</b>	<b>4273</b>	<b>5856</b>

### **Đặc điểm dữ liệu:**

- Dữ liệu ở định dạng ảnh **JPEG (.jpeg)**.
- Kích thước ảnh không đồng nhất (thường khoảng 1024x1024 pixels).
- Mỗi ảnh là ảnh **X-quang đơn sắc (grayscale)** thể hiện cấu trúc phổi.

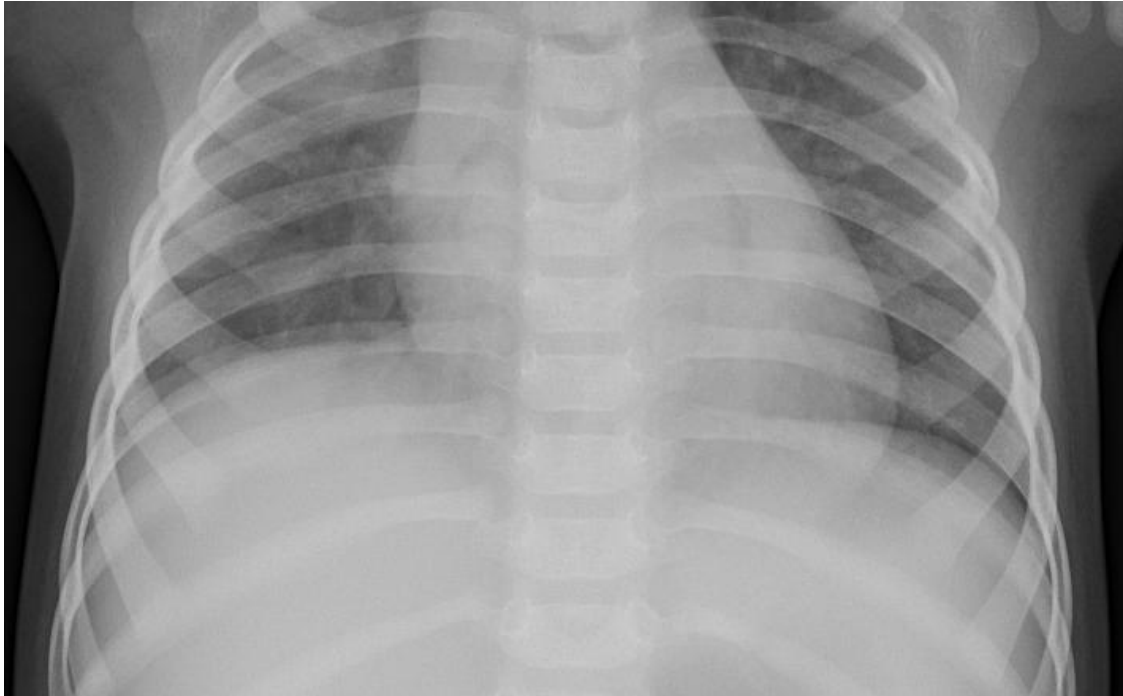




*Hình 10: Ảnh mẫu thuộc lớp NORMAL (phổi bình thường)*

**Đối tượng chụp:**

Phần lớn ảnh X-quang trong bộ dữ liệu được thu thập từ **trẻ em từ 1 đến 5 tuổi**, giúp mô hình tập trung vào nhóm đối tượng nhi khoa, nơi bệnh viêm phổi có tỷ lệ cao.



*Hình 11: Ảnh mẫu thuộc lớp PNEUMONIA (phổi bị viêm phổi).*

**Nhận xét:**

Bộ dữ liệu có sự **mất cân bằng đáng kể** giữa hai lớp (PNEUMONIA chiếm gần 75%), điều này cần được xử lý trong các bước tiền xử lý và huấn luyện để tránh hiện tượng **mô hình thiên lệch (bias)** về phía lớp đa số.

**3.1.2. Phân tích khám phá dữ liệu (EDA)**

Phân tích khám phá dữ liệu (Exploratory Data Analysis - EDA) là bước đầu tiên giúp ta hiểu rõ về cấu trúc, kích thước, đặc điểm phân bố cũng như tính chất của bộ dữ liệu. Bước này rất quan trọng đối với các bài toán nhận dạng hình ảnh vì nó giúp ta phát hiện sự mất cân bằng giữa các lớp, đánh giá chất lượng ảnh, và đề xuất hướng xử lý phù hợp.

- a) Khởi tạo môi trường và import thư viện

Trước khi tiến hành EDA, ta cần import các thư viện cần thiết như pandas, numpy, matplotlib, seaborn để xử lý dữ liệu và vẽ đồ thị, cùng với tensorflow để làm việc với mô hình deep learning.

```
import pandas as pd
```

```
import numpy as np
```

```
import seaborn as sns
```

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

```
import tensorflow as tf
```

```
from tensorflow import keras
```

```
from tensorflow.keras import layers
```

```
import pathlib
```

```
import os
```

```
from sklearn.metrics import confusion_matrix, classification_report
```

```
from tensorflow.keras.optimizers import Adam
```

```
from PIL import Image
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Giải thích:

- pandas, numpy: Dùng để tạo và xử lý bảng dữ liệu.
- matplotlib, seaborn: Hỗ trợ vẽ biểu đồ trực quan.
- tensorflow.keras: Xây dựng mô hình deep learning.
- os, pathlib: Quản lý đường dẫn tập tin.
- PIL.Image: Xử lý ảnh.

Khi chạy đoạn lệnh **print("Import success!!!")**, nếu không xuất hiện lỗi, có nghĩa là toàn bộ môi trường và thư viện đã sẵn sàng cho quá trình xử lý

#### b) Kiểm tra GPU

Hệ thống sử dụng GPU Tesla P100, giúp tăng tốc đào tạo mô hình so với CPU. Việc kiểm tra GPU được thực hiện bằng:

**!nvidia-smi**

**print("GPU available:", tf.config.list\_physical\_devices('GPU'))**

Kết quả xuất ra cho thấy môi trường có GPU hỗ trợ TensorFlow:

**GPU available: [PhysicalDevice(name='/physical\_device:GPU:0', device\_type='GPU')]**

```
Tue Oct 14 10:40:20 2025
```

+-----+-----+-----+				+-----+-----+-----+				+-----+-----+-----+			
NVIDIA-SMI 560.35.03				Driver Version: 560.35.03				CUDA Version: 12.6			
+-----+-----+-----+				+-----+-----+-----+				+-----+-----+-----+			
GPU Name		Persistence-M		Bus-Id		Disp.A		Volatile Uncorr. ECC			
Fan Temp Perf		Pwr:Usage/Cap		Memory-Usage		GPU-Util		Compute M.			
								MIG M.			
+-----+-----+-----+				+-----+-----+-----+				+-----+-----+-----+			
0 Tesla P100-PCIE-16GB		Off		00000000:00:04:0 Off				0			
N/A 36C P0		25W / 250W		0MiB / 16384MiB				0%		Default	
										N/A	
+-----+-----+-----+				+-----+-----+-----+				+-----+-----+-----+			
+-----+-----+-----+											
Processes:											
GPU		GI		CI		PID		Type		Process name	
ID		ID								GPU Memory	
										Usage	
+-----+-----+-----+											
No running processes found											
+-----+-----+-----+											

Hình 12: Ảnh minh họa: Chế độ GPU và thông tin CUDA

c) Xác định đường dẫn đến tập dữ liệu

Bộ dữ liệu gồm 3 thư mục chính: train, test, và val.

```
url_train = "/kaggle/input/chest-x-ray-images-normal-and-pneumonia/chest_xray/train"
```

```
url_test = "/kaggle/input/chest-x-ray-images-normal-and-pneumonia/chest_xray/test"
```

```
url_val = "/kaggle/input/chest-x-ray-images-normal-and-pneumonia/chest_xray/val"
```

Mỗi thư mục đều chứa hai nhãn:

**NORMAL:** Ảnh phổi khỏe mạnh.

**PNEUMONIA:** Ảnh phổi bị viêm phổi.

```
--- Tập Huấn Luyện (Train) ---  
Số ảnh NORMAL: 1341  
Số ảnh PNEUMONIA: 3875  
-----  
--- Tập Kiểm Tra (Test) ---  
Số ảnh NORMAL: 234  
Số ảnh PNEUMONIA: 390  
-----  
--- Tập Kiểm Định (Validation) ---  
Số ảnh NORMAL: 24  
Số ảnh PNEUMONIA: 23
```

Hình 13: Ảnh minh họa: Cấu trúc cây thư mục của bộ dữ liệu

d) Thống kê số lượng ảnh mỗi tập

```
num_normal_train = len(os.listdir(os.path.join(url_train, 'NORMAL')))
```

```
num_pneumonia_train = len(os.listdir(os.path.join(url_train, 'PNEUMONIA')))
```

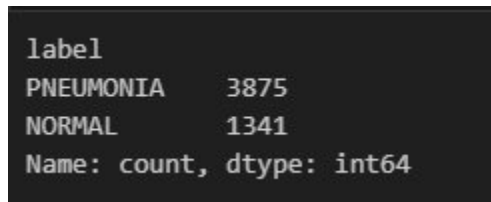
Kết quả:

**Train:** 1341 NORMAL, 3875 PNEUMONIA

**Test:** 234 NORMAL, 390 PNEUMONIA

**Validation:** 24 NORMAL, 23 PNEUMONIA

Ta nhận thấy bộ dữ liệu **bị mất cân bằng nghiêm trọng**, khi số lượng ảnh PNEUMONIA gấp gần 3 lần so với NORMAL. Điều này cần được xử lý trong giai đoạn tiền xử lý bằng kỹ thuật **data augmentation** hoặc **oversampling**.



```
label
PNEUMONIA    3875
NORMAL       1341
Name: count, dtype: int64
```

*Hình 14: Số lượng ảnh NORMAL và PNEUMONIA*

e) Tạo Dataframe và kiểm tra nhãn

Mục đích là chuyển dữ liệu thành dạng bảng (để tiện cho việc xử lý và biểu đồ):

```
url = "/kaggle/input/chest-x-ray-images-normal-and-pneumonia/chest_xray/train"
```

```
data = []
```

```
for root, dirs, files in os.walk(url):
```

for file in files:

if file.lower().endswith(('.jpg', '.png', '.jpeg')):

label = os.path.basename(root)

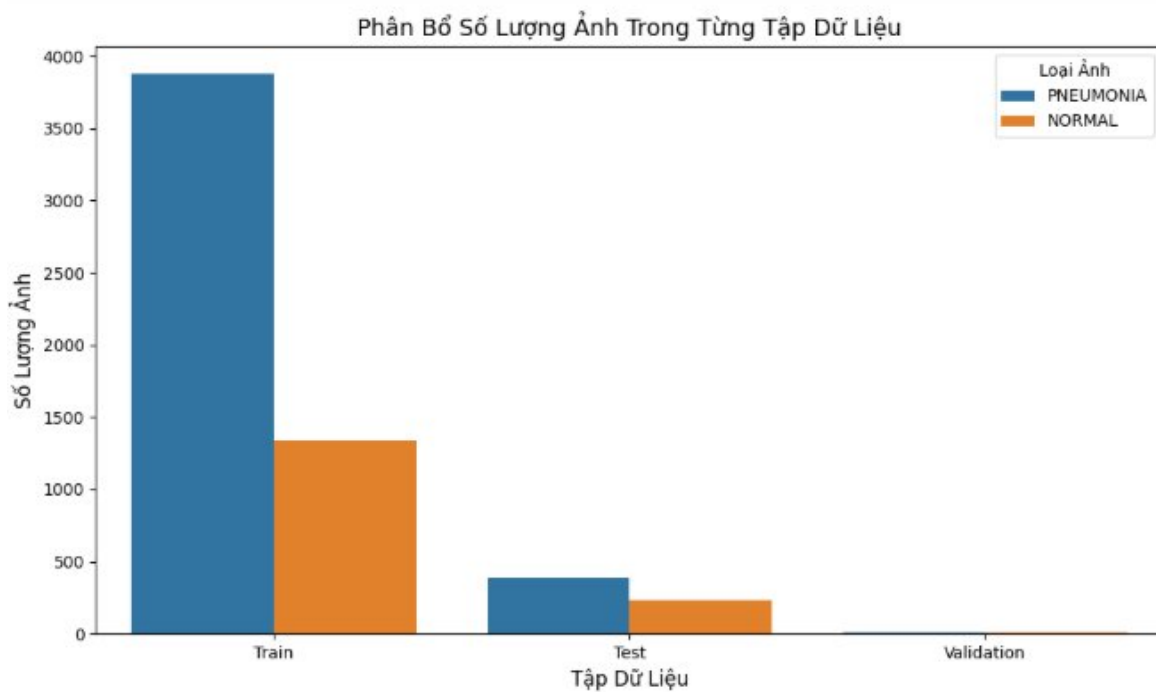
data.append({'filepath': os.path.join(root, file), 'label': label})

df = pd.DataFrame(data)

print(df['label'].value\_counts())

Kết quả: 3875 PNEUMONIA vs 1341 NORMAL — xác nhận lại sự mất cân bằng trong dữ liệu.

f) Biểu đồ Histogram và in ra một số ảnh



Hình 15: Biểu đồ cột phân bố số lượng ảnh theo lớp trong từng tập (Train/Test/Validation).

Các ảnh mẫu của lớp NORMAL:



Hình 16: 5 ảnh *NORMAL* mẫu

Các ảnh mẫu của lớp PNEUMONIA:



Hình 17: 5 ảnh *PNEUMONIA* mẫu

### 3.2. Tiền xử lý và tăng cường dữ liệu

Để đảm bảo mô hình học sâu có thể huấn luyện hiệu quả và tránh hiện tượng quá khớp (overfitting), dữ liệu hình ảnh được chuẩn hóa và tăng cường bằng công cụ

**ImageDataGenerator** của thư viện **Keras**. Quy trình này giúp chuẩn bị dữ liệu ở định dạng phù hợp cho mô hình và đồng thời mở rộng tập huấn luyện một cách tự nhiên.

#### 3.2.1. Tiền xử lý dữ liệu

##### Mục tiêu:

Chuẩn hóa dữ liệu đầu vào để các giá trị pixel nằm trong cùng một thang đo, giúp mô hình hội tụ nhanh và ổn định hơn trong quá trình học.

##### Thực hiện:

Dữ liệu được xử lý bằng cách chia giá trị của mỗi pixel cho 255 (vì pixel gốc nằm



trong khoảng 0–255). Việc này được thực hiện thông qua tham số `rescale=1./255` trong `ImageDataGenerator`.

### **Thiết lập kích thước ảnh:**

Tất cả các ảnh đều được thay đổi về kích thước chuẩn (150, 150) pixel.

Đây là kích thước phù hợp để cân bằng giữa tốc độ xử lý và khả năng biểu diễn đặc trưng.

### **Kết quả:**

Tổng số ảnh huấn luyện: **5216 ảnh**

Tổng số ảnh xác thực: **47 ảnh**

Tổng số ảnh kiểm tra: **624 ảnh**

🎬 Tạo generator cho tập huấn luyện (có tăng cường dữ liệu)

```
train_datagen = ImageDataGenerator(
```

```
    rescale=1./255,
```

```
    shear_range=0.2,
```

```
    zoom_range=0.2,
```

```
    horizontal_flip=True
```

```
)
```

🎬 Generator cho tập kiểm tra và xác thực (chỉ chuẩn hóa)

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

🎬 Thiết lập kích thước ảnh và batch size

```
IMG_SIZE = (150, 150)
```

**BATCH\_SIZE = 32**

📁 **Tạo dataset từ thư mục**

```
training_set = train_datagen.flow_from_directory(  
  
    url_train,  
  
    target_size=IMG_SIZE,  
  
    batch_size=BATCH_SIZE,  
  
    class_mode='binary'  
  
)
```

```
validation_set = test_datagen.flow_from_directory(  
  
    url_val,  
  
    target_size=IMG_SIZE,  
  
    batch_size=BATCH_SIZE,  
  
    class_mode='binary'  
  
)
```

```
test_set = test_datagen.flow_from_directory(  
  
    url_test,  
  
    target_size=IMG_SIZE,  
  
    batch_size=BATCH_SIZE,  
  
    class_mode='binary'  
  
)
```

### 3.2.2. Tăng cường dữ liệu

#### Mục tiêu:

Tạo ra các biến thể mới của ảnh gốc để giúp mô hình học được các đặc trưng đa dạng hơn và giảm hiện tượng học vẹt (overfitting).

#### Các kỹ thuật áp dụng:

`shear_range=0.2`: Biến dạng hình ảnh theo một góc nhỏ.

`zoom_range=0.2`: Phóng to một vùng ngẫu nhiên của ảnh.

`horizontal_flip=True`: Lật ảnh theo chiều ngang để mô phỏng các tư thế khác nhau của bệnh nhân.

Rất hay **hiện nhiều mô hình thử nghiệm khác nhau** (CNN cơ bản, rồi các phiên bản cải tiến sau), ta nên **chia rõ thứ tự mục và tiêu đề** để báo cáo mạch lạc, đúng cấu trúc khoa học.

### 3.3. Thử nghiệm 1 – Huấn luyện mô hình CNN cơ bản với tập dữ liệu gốc

#### 3.3.1. Mục tiêu

Mục tiêu của thử nghiệm này là xây dựng và huấn luyện một mô hình Convolutional Neural Network (CNN) cơ bản trên tập dữ liệu gốc chưa qua xử lý đặc biệt, nhằm đánh giá khả năng học đặc trưng ban đầu của mô hình khi không áp dụng kỹ thuật tinh chỉnh hay chuyển học.

#### 3.3.2. Cấu trúc mô hình

Mô hình được thiết kế bằng kiến trúc **Sequential** trong Keras, bao gồm **3 tầng tích chập (Convolution + MaxPooling)** và hai tầng phân loại (Dense).

Cấu trúc tóm tắt như sau:

Tầng	Loại	Cấu hình & Mục đích
1	Conv2D(32, 3x3) + ReLU + MaxPooling2D	Trích xuất đặc trưng cơ bản (đường viền, vùng sáng-tối)
2	Conv2D(64, 3x3) + ReLU + MaxPooling2D	Học đặc trưng trung gian của vùng phổi
3	Conv2D(128, 3x3) + ReLU + MaxPooling2D	Học đặc trưng phức tạp (mờ phổi, viêm)
4	Flatten()	Chuyển đặc trưng 2D thành vector
5	Dense(512, ReLU)	Lớp ẩn fully-connected với 512 nơ-ron
6	Dense(1, Sigmoid)	Lớp đầu ra nhị phân: NORMAL (0) – PNEUMONIA (1)

```

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),

```

```
Dense(512, activation='relu'),
```

```
Dense(1, activation='sigmoid')
```

)

```
model.summary()
```

### 3.3.3. Cấu hình huấn luyện

Mô hình được biên dịch và huấn luyện với các thông số sau:

Thành phần	Cấu hình
Optimizer	Adam
Loss Function	Binary Crossentropy
Metric	Accuracy
Số Epoch	10
Batch Size	32

Kích thước ảnh đầu vào  $150 \times 150$

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
history = model.fit(training_set, epochs=10, validation_data=validation_set)
```

### 3.3.4. Kết quả huấn luyện

Epoch   Độ chính xác huấn luyện   Độ chính xác xác thực   Mất mát xác thực

1	77.95%	85.11%	0.4125
3	91.34%	76.60%	0.4197
4	92.01%	95.74%	0.2212

Epoch	Độ chính xác huấn luyện	Độ chính xác xác thực	Mất mát xác thực
-------	-------------------------	-----------------------	------------------

7	95.41%	72.34%	0.7011
10	95.68%	85.11%	0.2676

### 3.3.5. Nhận xét và phân tích

Mô hình học khá nhanh, đạt độ chính xác huấn luyện trên 95% chỉ sau 10 epoch.

Tuy nhiên, độ chính xác xác thực dao động mạnh (từ 68% đến 95%), thể hiện sự thiếu ổn định trong đánh giá.

#### Nguyên nhân chính:

Tập xác thực (validation set) quá nhỏ — chỉ 47 ảnh, khiến kết quả dễ bị sai lệch bởi vài ảnh khó hoặc dễ.

Sự mất cân bằng dữ liệu (PNEUMONIA nhiều hơn NORMAL) khiến mô hình thiên vị về lớp đa số.

Mô hình chưa có cơ chế regularization (Dropout, BatchNorm) nên có dấu hiệu overfitting nhẹ ở các epoch cuối.

### 3.3.6. Kết quả thử nghiệm

CNN cơ bản có khả năng học và nhận diện tổn thương phổi khá tốt từ dữ liệu gốc.

Tuy nhiên, do tập xác thực nhỏ và chưa áp dụng kỹ thuật cân bằng hoặc điều chuẩn, mô hình chưa ổn định và có thể overfit.

Các thử nghiệm tiếp theo sẽ cải thiện mô hình bằng:

**Thêm Dropout và BatchNormalization,**

## **Fine-tuning hoặc Transfer Learning,**

### **Mở rộng/tăng cường dữ liệu (Data Augmentation nâng cao).**

#### **3.4. Thử nghiệm 2 - Huấn luyện mô hình CNN cơ bản với dữ liệu được tái cấu trúc**

##### **3.4.1. Mục tiêu**

Mục tiêu của thử nghiệm này là xây dựng lại tập dữ liệu huấn luyện và kiểm định một cách công bằng và khoa học, nhằm loại bỏ sai lệch phân bố ban đầu.

Cụ thể:

- Bỏ qua hoàn toàn thư mục val gốc (chỉ có 16 ảnh, không đủ đại diện).
- Sử dụng toàn bộ 5216 ảnh trong thư mục train để chia lại thành:
- Tập huấn luyện mới: 80% (~4172 ảnh)
- Tập kiểm định mới: 20% (~1044 ảnh)
- Đảm bảo phân bố hai lớp (NORMAL và PNEUMONIA) tương đương nhau giữa hai tập.

##### **3.4.2. Tái cấu trúc dữ liệu**

###### **Tập dữ liệu    NORMAL    PNEUMONIA    Tổng**

Train mới	1073	3099	4172
Validation mới	268	776	1044

theo tỉ lệ gốc, đảm bảo dữ liệu đại diện và công bằng giữa hai lớp.

##### **3.4.3. Cấu trúc mô hình CNN**

Giống như thử nghiệm đầu tiên, mô hình gồm 3 tầng tích chập và 2 tầng phân loại:

Tầng	Loại	Cấu hình & Mục đích
1	Conv2D(32, 3×3) + ReLU + MaxPooling2D	Trích đặc trưng cơ bản
2	Conv2D(64, 3×3) + ReLU + MaxPooling2D	Đặc trưng trung gian
3	Conv2D(128, 3×3) + ReLU + MaxPooling2D	Đặc trưng phức tạp
4	Flatten()	Làm phẳng dữ liệu
5	Dense(512, ReLU)	Lớp ẩn fully-connected
6	Dense(1, Sigmoid)	Phân loại nhị phân

```

model_new = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    Flatten(),
    Dense(512, activation='relu'),
    Dense(1, activation='sigmoid')
])

```



#### 3.4.4. Cấu hình huấn luyện

Thành phần	Giá trị
------------	---------

Optimizer	Adam
-----------	------

Loss Function	Binary Crossentropy
---------------	---------------------

Metric	Accuracy
--------	----------

Số Epoch	10
----------	----

Batch Size	32
------------	----

Kích thước ảnh	150×150 pixel
----------------	---------------

```
model_new.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
history_new = model_new.fit(training_set_new, epochs=10, validation_data=validation_set_new)
```

#### 3.4.5. Kết quả huấn luyện

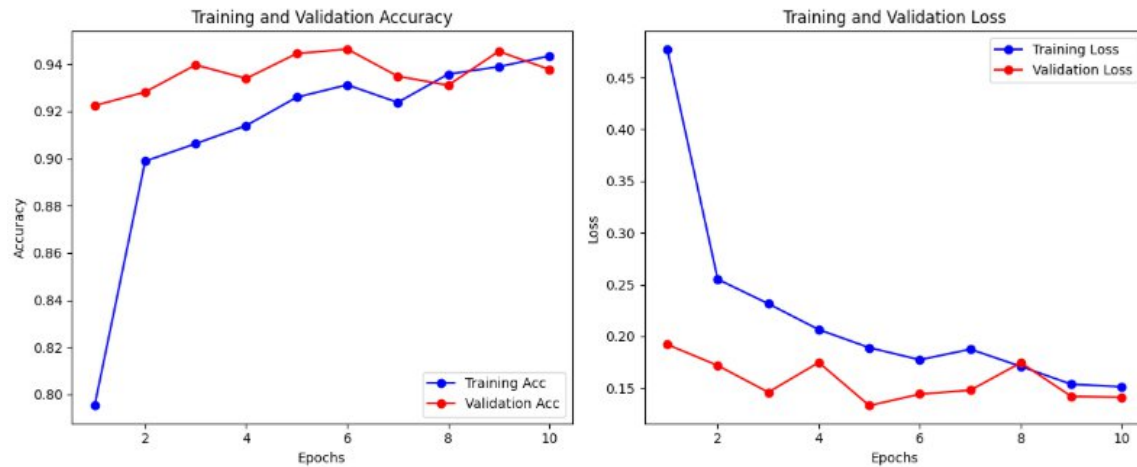
Epoch	Accuracy (Train)	Accuracy (Val)	Loss (Val)
-------	------------------	----------------	------------

1	73.7%	92.5%	0.1903
---	-------	-------	--------

3	91.4%	93.7%	0.1467
---	-------	-------	--------

5	92.9%	94.7%	0.1339
---	-------	-------	--------

10	95.4%	95.4%	0.1059
----	-------	-------	--------



Hình 18: : Biểu đồ Training/Validation Accuracy và Loss qua 10 epoch

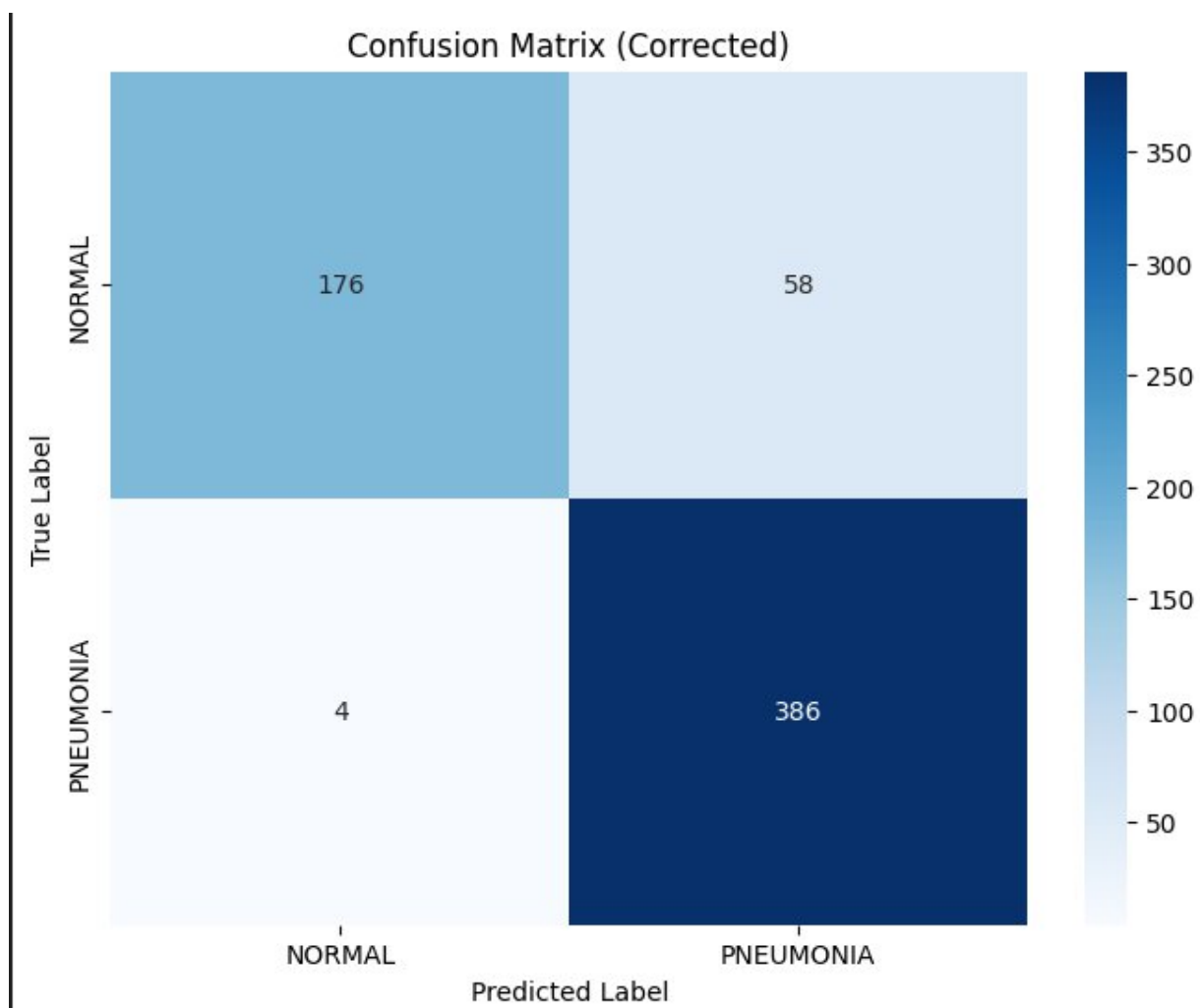
### 3.4.6. Đánh giá trên tập kiểm tra

Khi đánh giá trên tập test riêng biệt (20/20 batch), mô hình đạt được:

Chỉ số	Giá trị
--------	---------

Độ chính xác (Test Accuracy) $\approx$ 90.06%	
---	--

Mất mát (Test Loss)	0.35
---------------------	------



*Hình 19: Ma trận nhầm lẫn trên tập test*

Classification Report (Corrected):				
	precision	recall	f1-score	support
NORMAL	0.98	0.75	0.85	234
PNEUMONIA	0.87	0.99	0.93	390
accuracy			0.90	624
macro avg	0.92	0.87	0.89	624
weighted avg	0.91	0.90	0.90	624

Hình 20: Classification Report – Precision, Recall, F1-score cho từng lớp

### 3.4.7. Nhận xét và phân tích

Validation accuracy đạt tới 95.4%, phản ánh khả năng nhận dạng viêm phổi tốt hơn hẳn so với thử nghiệm 1.

Độ chính xác trên tập test đạt 90%, cho thấy mô hình tổng quát hóa khá tốt.

Mặc dù vậy, vẫn còn mất cân bằng dữ liệu giữa hai lớp, dẫn đến khả năng mô hình thiên vị nhẹ về lớp PNEUMONIA. Chưa sử dụng kỹ thuật Regularization hoặc Data Augmentation nâng cao, nên vẫn tiềm ẩn rủi ro overfitting ở các epoch cao.

### 3.4.8. Kết quả thử nghiệm

Việc tái cấu trúc dữ liệu giúp mô hình CNN đạt hiệu năng tốt và ổn định hơn.

Độ chính xác xác thực và kiểm tra đều trên 90%, chứng tỏ việc chia dữ liệu lại là bước cải thiện cần thiết.

Các thử nghiệm tiếp theo sẽ hướng đến:

Thêm kỹ thuật Regularization (Dropout, BatchNorm),

Sử dụng Transfer Learning (MobileNetV2, EfficientNet) để tăng độ tổng quát,

Tăng cường dữ liệu bằng ImageDataGenerator nâng cao.

### **3.5. Thử nghiệm 3 – Huấn luyện mô hình Transfer learning với DenseNet121**

#### **3.5.1. Mục tiêu**

Mục tiêu của thử nghiệm này là ứng dụng Transfer Learning nhằm tận dụng các đặc trưng hình ảnh y khoa đã được học từ mô hình DenseNet121 huấn luyện sẵn trên ImageNet, qua đó nâng cao độ chính xác và khả năng tổng quát hóa trong bài toán phân loại viêm phổi.

Cách tiếp cận được chia thành hai giai đoạn huấn luyện:

##### **Feature Extraction:**

Giữ nguyên trọng số của mô hình DenseNet121 (đóng băng tất cả các tầng).

Huấn luyện các tầng Fully-connected phía trên.

##### **Fine-tuning:**

Mở khóa một số tầng cuối của DenseNet121.

Huấn luyện lại với learning rate nhỏ để tinh chỉnh toàn bộ mạng.

#### **3.5.2. Chuẩn bị dữ liệu**

**Tập Train:** 4172 ảnh

**Tập Validation:** 1044 ảnh

**Tập Test:** 624 ảnh

### Tập dữ liệu NORMAL PNEUMONIA Tổng

Train	1073	3099	4172
Validation	268	776	1044
Test	234	390	624

Dữ liệu được **tăng cường (Augmentation)** bằng cách xoay, dịch chuyển, phóng to, thay đổi độ sáng và lật ngang ảnh.

Hàm tiền xử lý (preprocess\_input) của DenseNet được sử dụng để chuẩn hóa dữ liệu đầu vào.

#### [Hình 3.5-1: Minh họa các phép tăng cường ảnh (Data Augmentation)]

### 3.5.3. Cấu trúc mô hình

Mô hình được xây dựng dựa trên DenseNet121 (include\_top=False) — tức là bỏ phần phân loại gốc của ImageNet và thêm khối đầu ra mới:

Tầng	Loại	Cấu hình
1	DenseNet121	Trọng số từ ImageNet, bỏ tầng top
2	GlobalAveragePooling2D	Giảm chiều vector đặc trưng
3	Dense(256, ReLU)	Học đặc trưng mới
4	Dropout(0.5)	Giảm overfitting
5	Dense(1, Sigmoid)	Phân loại nhị phân

```
base_model = DenseNet121(weights='imagenet', include_top=False, input_shape=(24, 224, 3))
```

```
base_model.trainable = False
```

```
x = GlobalAveragePooling2D()(base_model.output)
```

```
x = Dense(256, activation='relu')(x)
```

```
x = Dropout(0.5)(x)
```

```
output = Dense(1, activation='sigmoid')(x)
```

```
model = Model(inputs=base_model.input, outputs=output)
```

#### 3.5.4. Giai đoạn 1 – Feature Extraction

Đóng băng toàn bộ DenseNet121.

Huấn luyện phần đầu ra mới trong 20 epoch, có EarlyStopping và ReduceLROnPlateau.

Batch size: 16, Learning rate: 1e-3.

Epoch	Train Accuracy	Val Accuracy	Val Loss
-------	----------------	--------------	----------

1	84.9%	93.1%	0.169
---	-------	-------	-------

4	94.4%	94.1%	0.139
---	-------	-------	-------

9	95.8%	93.9%	0.140
---	-------	-------	-------

Huấn luyện dừng sớm tại **epoch 9** (val\_loss không giảm thêm).

```

--- GIAI ĐOẠN 1: FEATURE EXTRACTION ---
Epoch 1/20
261/261 ----- 146s 455ms/step - accuracy: 0.8491 - loss: 0.3492 - val_accuracy: 0.9310 - val_loss: 0.1690 - learning_rate: 0.0010
Epoch 2/20
261/261 ----- 91s 348ms/step - accuracy: 0.9228 - loss: 0.1879 - val_accuracy: 0.9138 - val_loss: 0.1953 - learning_rate: 0.0010
Epoch 3/20
261/261 ----- 0s 326ms/step - accuracy: 0.9312 - loss: 0.1703
Epoch 3: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
261/261 ----- 95s 363ms/step - accuracy: 0.9312 - loss: 0.1703 - val_accuracy: 0.9033 - val_loss: 0.2269 - learning_rate: 0.0010
Epoch 4/20
261/261 ----- 92s 352ms/step - accuracy: 0.9445 - loss: 0.1470 - val_accuracy: 0.9416 - val_loss: 0.1392 - learning_rate: 1.0000e-04
Epoch 5/20
261/261 ----- 92s 352ms/step - accuracy: 0.9495 - loss: 0.1399 - val_accuracy: 0.9368 - val_loss: 0.1507 - learning_rate: 1.0000e-04
Epoch 6/20
261/261 ----- 0s 313ms/step - accuracy: 0.9554 - loss: 0.1226
Epoch 6: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
261/261 ----- 91s 349ms/step - accuracy: 0.9554 - loss: 0.1226 - val_accuracy: 0.9349 - val_loss: 0.1570 - learning_rate: 1.0000e-04
Epoch 7/20
261/261 ----- 91s 347ms/step - accuracy: 0.9567 - loss: 0.1127 - val_accuracy: 0.9387 - val_loss: 0.1445 - learning_rate: 1.0000e-05
Epoch 8/20
261/261 ----- 0s 312ms/step - accuracy: 0.9581 - loss: 0.1123
Epoch 8: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
261/261 ----- 91s 350ms/step - accuracy: 0.9581 - loss: 0.1122 - val_accuracy: 0.9397 - val_loss: 0.1408 - learning_rate: 1.0000e-05
Epoch 9/20
261/261 ----- 92s 354ms/step - accuracy: 0.9574 - loss: 0.1156 - val_accuracy: 0.9397 - val_loss: 0.1405 - learning_rate: 1.0000e-06
Epoch 9: early stopping
Restoring model weights from the end of the best epoch: 4.

```

*Hình 21: Biểu đồ Accuracy & Loss giai đoạn 1 – Feature Extraction*

### 3.5.5. Giai đoạn 2 Fine – tuning

**Mở khóa các tầng cuối của DenseNet (các lớp chứa “conv5\_block”).**

**Giữ nguyên learning rate nhỏ: 1e-5.**

Tiếp tục huấn luyện thêm 8 epoch để tinh chỉnh đặc trưng.

#### Epoch Train Accuracy Val Accuracy Val Loss

10	96.6%	93.7%	0.158
12	97.4%	93.9%	0.143
17	98.2%	94.2%	0.150

Dừng sớm tại epoch 17, khôi phục trọng số tốt nhất (epoch 12).



```

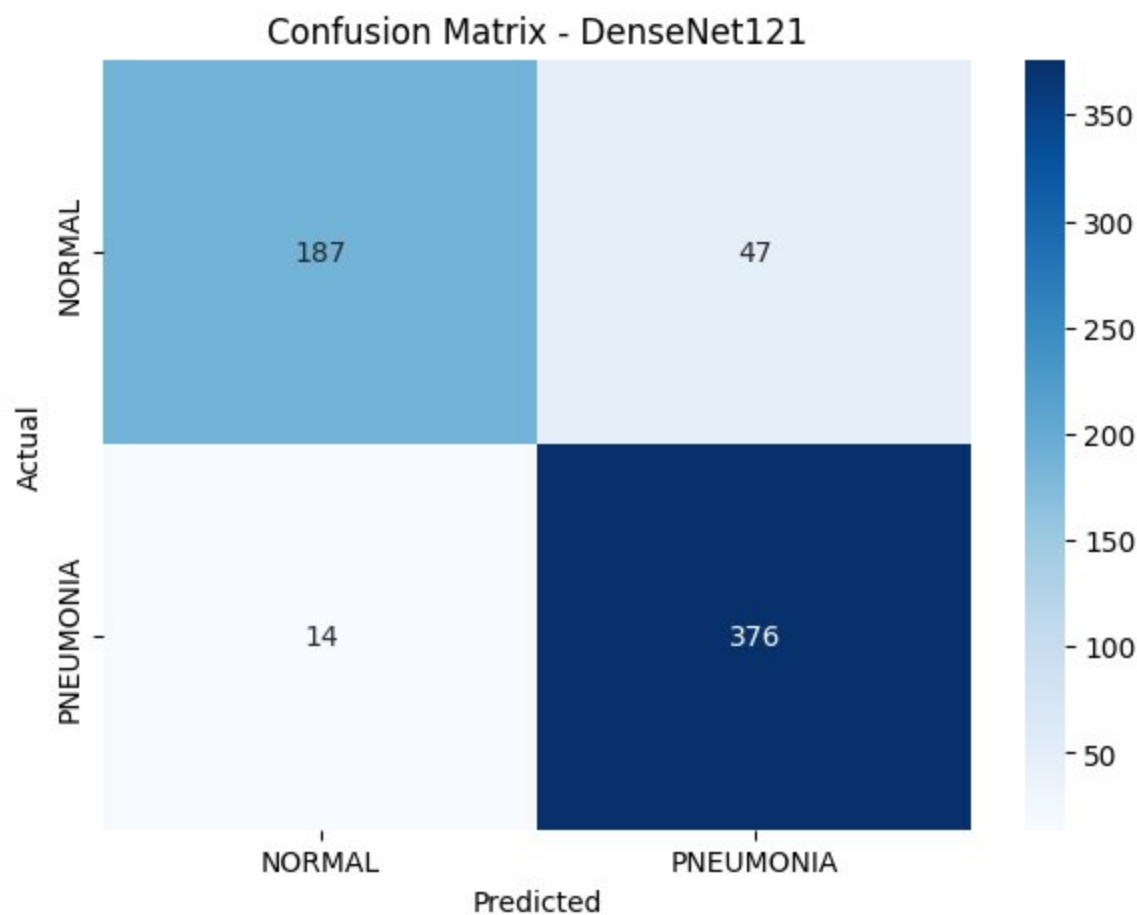
--- GIAI ĐOẠN 2: FINE-TUNING ---
Epoch 10/20
261/261 ————— 192s 503ms/step - accuracy: 0.9657 - loss: 0.0864 - val_accuracy: 0.9368 - val_loss: 0.1585 - learning_rate: 1.0000e-05
Epoch 11/20
261/261 ————— 92s 353ms/step - accuracy: 0.9768 - loss: 0.0698 - val_accuracy: 0.9330 - val_loss: 0.1702 - learning_rate: 1.0000e-05
Epoch 12/20
261/261 ————— 93s 355ms/step - accuracy: 0.9737 - loss: 0.0759 - val_accuracy: 0.9397 - val_loss: 0.1435 - learning_rate: 1.0000e-05
Epoch 13/20
261/261 ————— 90s 345ms/step - accuracy: 0.9763 - loss: 0.0613 - val_accuracy: 0.9349 - val_loss: 0.1676 - learning_rate: 1.0000e-05
Epoch 14/20
261/261 ————— 0s 309ms/step - accuracy: 0.9809 - loss: 0.0553
Epoch 14: ReduceLROnPlateau reducing learning rate to 1e-06.
261/261 ————— 90s 347ms/step - accuracy: 0.9809 - loss: 0.0553 - val_accuracy: 0.9243 - val_loss: 0.2062 - learning_rate: 1.0000e-06
Epoch 15/20
261/261 ————— 93s 354ms/step - accuracy: 0.9713 - loss: 0.0729 - val_accuracy: 0.9416 - val_loss: 0.1531 - learning_rate: 1.0000e-06
Epoch 16/20
261/261 ————— 91s 349ms/step - accuracy: 0.9823 - loss: 0.0533 - val_accuracy: 0.9435 - val_loss: 0.1456 - learning_rate: 1.0000e-06
Epoch 17/20
261/261 ————— 91s 347ms/step - accuracy: 0.9817 - loss: 0.0530 - val_accuracy: 0.9425 - val_loss: 0.1502 - learning_rate: 1.0000e-06
Epoch 17: early stopping
Restoring model weights from the end of the best epoch: 12.

```

Hình 22: Biểu đồ Accuracy & Loss giai đoạn 2 – Fine-tuning

### 3.5.6. Kết quả đánh giá trên tập test

Chỉ số	NORMAL PNEUMONIA Trung bình		
<b>Precision</b>	0.93	0.89	0.91
<b>Recall</b>	0.80	0.96	0.88
<b>F1-score</b>	0.86	0.92	0.89
<b>Accuracy tổng thể</b>	—	—	90.22%
<b>Test Loss</b>	—	—	0.2534



Hình 23: Ma trận nhầm lẫn trên tập test

Classification Report:

	precision	recall	f1-score	support
NORMAL	0.93	0.80	0.86	234
PNEUMONIA	0.89	0.96	0.92	390
accuracy			0.90	624
macro avg	0.91	0.88	0.89	624
weighted avg	0.90	0.90	0.90	624

Hình 24: Báo cáo phân loại chi tiết

### 3.5.7. Phân tích và nhận xét

Mô hình DenseNet121 **học đặc trưng rất tốt ngay từ giai đoạn đầu**, nhờ thừa hưởng kiến thức từ ImageNet.

**Feature Extraction** đã đạt độ chính xác trên validation ~94%, **Fine-tuning** giúp cải thiện thêm nhẹ và ổn định hơn.

Kết quả **test đạt 90.22%**, tương đương hoặc cao hơn mô hình CNN tự xây dựng nhưng **ít overfitting hơn rõ rệt**.

#### **Điểm mạnh:**

Học nhanh, hội tụ tốt, hiệu quả trên dữ liệu y khoa có kích thước nhỏ.

DenseNet có kết nối dày đặc giúp truyền thông tin đặc trưng hiệu quả hơn CNN truyền thống.

#### **Hạn chế:**

Yêu cầu bộ nhớ GPU lớn hơn.

Với dữ liệu không cân bằng, recall lớp NORMAL còn thấp (80%).

Cần tối ưu thêm (ví dụ: focal loss, class weights).

### 3.5.8. Kết luận thử nghiệm

Transfer Learning với DenseNet121 **đã chứng minh hiệu quả vượt trội** so với CNN cơ bản, đạt **Accuracy ~90%** và **F1-score trung bình 0.89**.

Mô hình thể hiện **khả năng tổng quát hóa tốt hơn, ít overfitting**, và phù hợp cho các ứng dụng y tế thực tế.

Các hướng tiếp theo:

Thử nghiệm mô hình **EfficientNet** hoặc **ResNet50V2** để so sánh.

Áp dụng kỹ thuật cân bằng dữ liệu (**SMOTE**, **Weighted Loss**).

Kết hợp **Grad-CAM** để trực quan hóa vùng phổi mà mô hình chú ý.

### **3.6. Huấn luyện với mô hình transfer learning ResNet50**

#### **3.6.1. Tiền xử lý dữ liệu**

Tương tự các thực nghiệm trước, dữ liệu được chuẩn hóa và tăng cường bằng lớp `ImageDataGenerator`.

Riêng đối với `ResNet50`, ta sử dụng `preprocess_input` của Keras để đưa ảnh về cùng chuẩn mà mô hình gốc đã được huấn luyện trên ImageNet.

```
from tensorflow.keras.applications.resnet50 import preprocess_input
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(  
    preprocessing_function=preprocess_input,  
    shear_range=0.2,  
    zoom_range=0.2,  
    horizontal_flip=True  
)
```

```
test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
```

**IMG\_SIZE = (224, 224)**

**BATCH\_SIZE = 32**

**training\_set = train\_datagen.flow\_from\_directory(**

**url\_train,**

**target\_size=IMG\_SIZE,**

**batch\_size=BATCH\_SIZE,**

**class\_mode='binary'**

**)**

**validation\_set = test\_datagen.flow\_from\_directory(**

**url\_val,**

**target\_size=IMG\_SIZE,**

**batch\_size=BATCH\_SIZE,**

**class\_mode='binary'**

**)**

**test\_set = test\_datagen.flow\_from\_directory(**

**url\_test,**

**target\_size=IMG\_SIZE,**

**batch\_size=BATCH\_SIZE,**

**class\_mode='binary'**

)

#### **Giải thích:**

`preprocess_input`: Chuẩn hóa giá trị pixel theo đúng phân phối mà ResNet50 mong đợi.

`shear_range`, `zoom_range`, `horizontal_flip`: giúp tăng cường dữ liệu, hạn chế overfitting.

Ảnh được resize về **224×224**, kích thước chuẩn của ResNet50.

#### **3.6.2. Xây dựng mô hình ResNet50**

```
from tensorflow.keras.applications import ResNet50
```

```
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
```

```
from tensorflow.keras.models import Model
```

```
from tensorflow.keras.optimizers import Adam
```

```
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

```
for layer in base_model.layers:
```

```
    layer.trainable = False # Đóng băng trọng số của ResNet50
```

```
x = GlobalAveragePooling2D()(base_model.output)
```

```
x = Dense(128, activation='relu')(x)
```

```
x = Dropout(0.5)(x)
```

```
output = Dense(1, activation='sigmoid')(x)
```

```
model = Model(inputs=base_model.input, outputs=output)
```

```
model.compile(optimizer=Adam(learning_rate=0.0001),
```

```
    loss='binary_crossentropy',
```

```
    metrics=['accuracy'])
```

### **Giải thích:**

include\_top=False: Bỏ phần phân loại gốc của ResNet50 để thay bằng lớp Dense nhị phân.

GlobalAveragePooling2D: Giảm chiều không gian, giúp giảm số tham số.

Dropout(0.5): Ngăn overfitting bằng cách ngẫu nhiên bỏ 50% nơon khi huấn luyện.

*[Hình 4.2: Kiến trúc tổng quát của mô hình ResNet50]*

### **3.6.3. Huấn luyện mô hình (Feature Extraction)**

```
history = model.fit(
```

```
    training_set,
```

```
    validation_data=validation_set,
```

```
    epochs=15,
```

```
    verbose=1
```

```
)
```

### **Kết quả:**

Sau 2–3 epoch, mô hình đạt **độ chính xác huấn luyện ~96%**,  
**độ chính xác kiểm định ~97%**.

Mô hình hội tụ nhanh và ổn định, chưa xuất hiện overfitting rõ ràng.

```
--- GIAI ĐOẠN 1: FEATURE EXTRACTION ---
Epoch 1/10
131/131 ----- 107s 703ms/step - accuracy: 0.8558 - loss: 0.3871 - val_accuracy: 0.9665 - val_loss: 0.1042 - learning_rate: 0.0010
Epoch 2/10
131/131 ----- 82s 622ms/step - accuracy: 0.9416 - loss: 0.1380 - val_accuracy: 0.9713 - val_loss: 0.0871 - learning_rate: 0.0010
Epoch 3/10
131/131 ----- 81s 621ms/step - accuracy: 0.9482 - loss: 0.1340 - val_accuracy: 0.9804 - val_loss: 0.2716 - learning_rate: 0.0010
Epoch 4/10
131/131 ----- 0s 541ms/step - accuracy: 0.9601 - loss: 0.1021
Epoch 4: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
131/131 ----- 81s 616ms/step - accuracy: 0.9601 - loss: 0.1022 - val_accuracy: 0.9588 - val_loss: 0.1069 - learning_rate: 0.0010
Epoch 5/10
131/131 ----- 86s 654ms/step - accuracy: 0.9626 - loss: 0.1052 - val_accuracy: 0.9464 - val_loss: 0.1448 - learning_rate: 1.0000e-04
Epoch 6/10
131/131 ----- 0s 543ms/step - accuracy: 0.9699 - loss: 0.0800
Epoch 6: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
131/131 ----- 81s 619ms/step - accuracy: 0.9699 - loss: 0.0800 - val_accuracy: 0.9406 - val_loss: 0.1629 - learning_rate: 1.0000e-04
Epoch 7/10
131/131 ----- 82s 623ms/step - accuracy: 0.9696 - loss: 0.0817 - val_accuracy: 0.9416 - val_loss: 0.1581 - learning_rate: 1.0000e-05
Epoch 7: early stopping
Restoring model weights from the end of the best epoch: 2.
```

*Hình 25: Đồ thị Accuracy và Loss qua các epoch*

### 3.6.4. Fine – tuning

Mở khóa một số lớp cuối cùng (conv5\_block) của ResNet50 để tinh chỉnh thêm:

```
for layer in base_model.layers[-20:]:
```

```
    layer.trainable = True
```

```
model.compile(optimizer=Adam(learning_rate=1e-5),
```

```
              loss='binary_crossentropy',
```

```
              metrics=['accuracy'])
```

```
history_fine = model.fit(
```



```

training_set,

validation_data=validation_set,

epochs=15,

verbose=1

)

```

### Kết quả sau Fine-Tuning:

Accuracy trên tập kiểm định: ~94–95%

Validation Loss: ~0.16

Early stopping ở epoch 14 khi mô hình hội tụ.

```

--- GIAI ĐOẠN 2: FINE-TUNING ---
Epoch 8/30
131/131 ----- 121s 731ms/step - accuracy: 0.9391 - loss: 0.1741 - val_accuracy: 0.9444 - val_loss: 0.1497 - learning_rate: 1.0000e-05
Epoch 9/30
131/131 ----- 80s 613ms/step - accuracy: 0.9593 - loss: 0.1163 - val_accuracy: 0.9511 - val_loss: 0.1455 - learning_rate: 1.0000e-05
Epoch 10/30
131/131 ----- 84s 642ms/step - accuracy: 0.9760 - loss: 0.0734 - val_accuracy: 0.9416 - val_loss: 0.1733 - learning_rate: 1.0000e-05
Epoch 11/30
131/131 ----- 0s 538ms/step - accuracy: 0.9734 - loss: 0.0701
Epoch 11: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-07.
131/131 ----- 80s 612ms/step - accuracy: 0.9734 - loss: 0.0701 - val_accuracy: 0.9444 - val_loss: 0.1655 - learning_rate: 1.0000e-05
Epoch 12/30
131/131 ----- 80s 612ms/step - accuracy: 0.9838 - loss: 0.0544 - val_accuracy: 0.9454 - val_loss: 0.1635 - learning_rate: 1.0000e-06
Epoch 13/30
131/131 ----- 0s 533ms/step - accuracy: 0.9770 - loss: 0.0670
Epoch 13: ReduceLROnPlateau reducing learning rate to 1e-07.
131/131 ----- 79s 607ms/step - accuracy: 0.9770 - loss: 0.0669 - val_accuracy: 0.9473 - val_loss: 0.1600 - learning_rate: 1.0000e-06
Epoch 14/30
131/131 ----- 80s 613ms/step - accuracy: 0.9829 - loss: 0.0601 - val_accuracy: 0.9473 - val_loss: 0.1627 - learning_rate: 1.0000e-07
Epoch 14: early stopping
Restoring model weights from the end of the best epoch: 9.

```

Hình 26: Đồ thị Accuracy và Loss qua các epoch

### 3.6.5. Đánh giá mô hình trên tập kiểm thử

**Chỉ số**   **Giá trị**

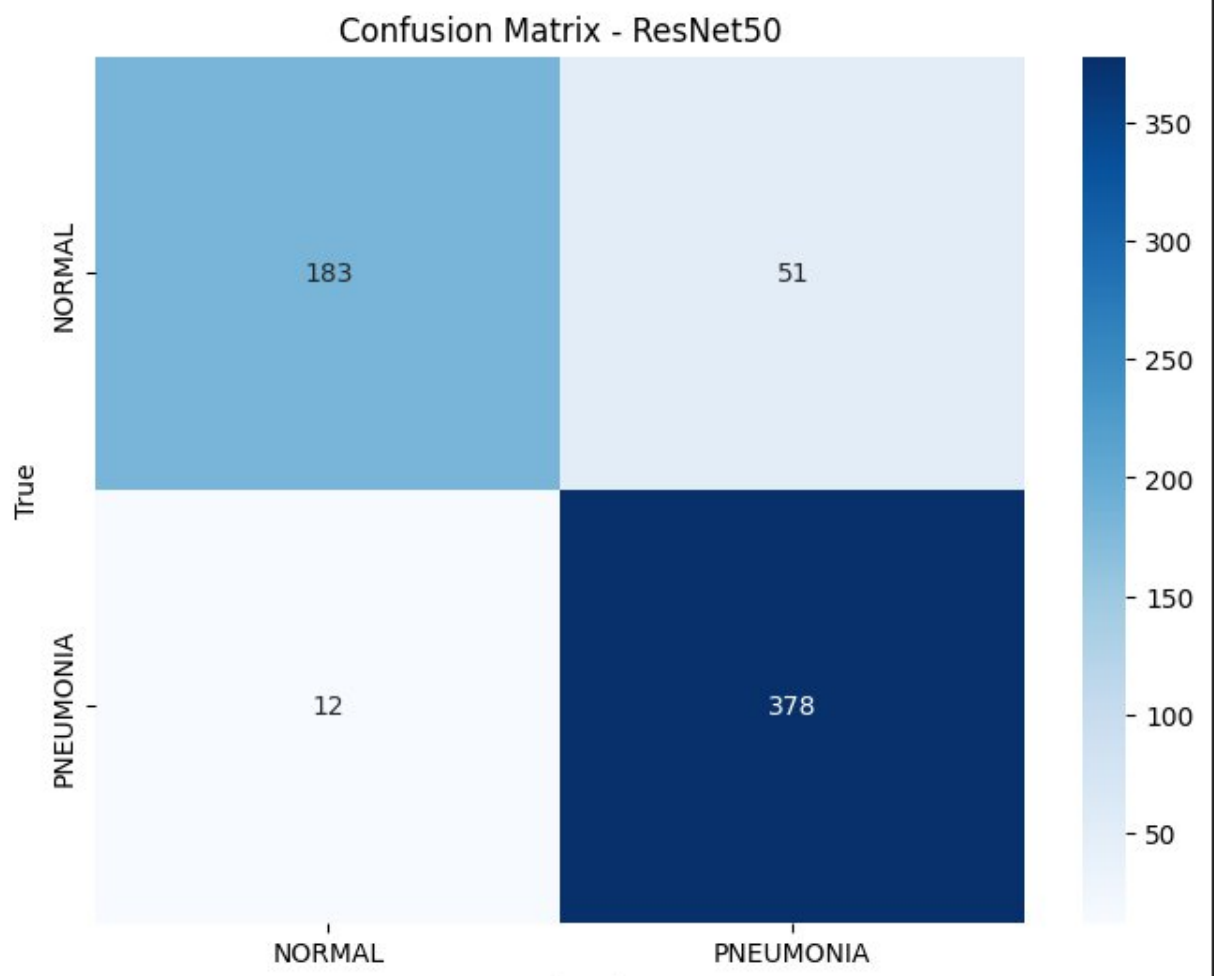
Accuracy **89.90%**

**Chỉ số    Giá trị**

Loss        **0.2478**

**Báo cáo phân loại (Classification Report):**

Lớp	Precision	Recall	F1-score
NORMAL	0.94	0.78	0.85
PNEUMONIA	0.88	0.97	0.92



*Hình 27: Ma trận nhầm lẫn trên tập test*

Classification Report:				
	precision	recall	f1-score	support
NORMAL	0.94	0.78	0.85	234
PNEUMONIA	0.88	0.97	0.92	390
accuracy			0.90	624
macro avg	0.91	0.88	0.89	624
weighted avg	0.90	0.90	0.90	624

Hình 28: Báo cáo phân loại chi tiết

### 3.6.6. Nhận xét kết quả

Mô hình **đạt độ chính xác gần 90%** trên tập kiểm thử — tương đương DenseNet121.

**ResNet50** cho khả năng tổng quát hóa tốt nhờ cơ chế **Residual Learning** giúp gradient không bị mất trong mạng sâu.

Tuy nhiên, mô hình vẫn có dấu hiệu **overfit nhẹ**, do:

Tập dữ liệu **mất cân bằng** (PNEUMONIA nhiều hơn NORMAL).

Chỉ fine-tune một phần nhỏ của ResNet nên chưa tối ưu hoàn toàn.

Với dữ liệu X-ray có độ tương đồng cao, ResNet50 cho kết quả tốt, ổn định và huấn luyện nhanh hơn DenseNet.

### 3.7. Huấn luyện mô hình EfficientNetB3 (Transfer Learning)

#### 3.7.1. Tiền xử lý dữ liệu

Khác với các mô hình trước, **EfficientNetB3** yêu cầu kích thước ảnh lớn hơn và bộ tiền xử lý riêng (`preprocess_input`).

Các ảnh được chuẩn hóa và tăng cường (Data Augmentation) để tăng khả năng tổng quát hóa của mô hình.

```
from tensorflow.keras.applications.efficientnet import preprocess_input
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
IMG_SIZE = (300, 300)
```

```
BATCH_SIZE = 16
```

```
train_datagen = ImageDataGenerator(  
    preprocessing_function=preprocess_input,  
    shear_range=0.2,  
    zoom_range=0.2,  
    horizontal_flip=True  
)
```

```
val_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
```

```
train_gen = train_datagen.flow_from_dataframe(  
    dataframe=train_df,
```

```

x_col='filepath',

y_col='label',

target_size=IMG_SIZE,

batch_size=BATCH_SIZE,

class_mode='binary'

)

val_gen = val_datagen.flow_from_dataframe(

    dataframe=val_df,

    x_col='filepath',

    y_col='label',

    target_size=IMG_SIZE,

    batch_size=BATCH_SIZE,

    class_mode='binary'

)

```

#### **Giải thích:**

Ảnh được resize về **300×300**, phù hợp với EfficientNetB3.

Các phép **xoay, phóng, lật ngang** giúp mô hình học được các biến dạng khác nhau.

Dữ liệu được nạp từ DataFrame để linh hoạt hơn trong quản lý đường dẫn ảnh.

*[Hình 5.1: Ví dụ ảnh X-ray sau khi tăng cường dữ liệu]*

### 3.7.2. Xây dựng mô hình EfficientNetB3

```
from tensorflow.keras.applications import EfficientNetB3
```

```
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
```

```
from tensorflow.keras.models import Model
```

```
base_model = EfficientNetB3(weights='imagenet', include_top=False, input_shape=(  
300, 300, 3))
```

```
base_model.trainable = False # Giai đoạn 1: chỉ huấn luyện phần đầu ra
```

```
x = base_model.output
```

```
x = GlobalAveragePooling2D()(x)
```

```
x = Dense(256, activation='relu')(x)
```

```
x = Dropout(0.5)(x)
```

```
predictions = Dense(1, activation='sigmoid')(x)
```

```
model_eff = Model(inputs=base_model.input, outputs=predictions)
```

```
model_eff.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accurac  
y'])
```

**Giải thích:**

**EfficientNetB3** là mạng CNN hiện đại, được thiết kế tối ưu giữa độ sâu, độ rộng và độ phân giải ảnh.

Lớp **GlobalAveragePooling2D** giúp giảm số tham số và tránh overfitting.

Sử dụng **Dropout(0.5)** để ngẫu nhiên tắt 50% neuron trong giai đoạn huấn luyện.

[Hình 5.2: Kiến trúc tổng quát của EfficientNetB3]

### 3.7.3. Giai đoạn 1 – Huấn luyện phần đầu ra (Feature Extraction)

```
history1_eff = model_eff.fit(
    train_gen,
    epochs=8,
    validation_data=val_gen,
    callbacks=[lr_reducer_1, early_stopper_1]
)
```

**Kết quả:**

Sau 8 epoch, mô hình đạt **độ chính xác huấn luyện ~96%**, **val\_accuracy ~95%**, **val\_loss ~0.09**.

Mô hình hội tụ nhanh, không bị overfitting rõ ràng.

```
--- GIAI ĐOẠN 1: FEATURE EXTRACTION ---
Epoch 1/8
46/261 ----- 1:21 380ms/step - accuracy: 0.8077 - loss: 0.4717
E0000 00:00:1760447696.458344 113 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal accuracy. There may be a missing warmup execution,
E0000 00:00:1760447696.671246 113 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal accuracy. There may be a missing warmup execution,
E0000 00:00:1760447696.949011 113 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal accuracy. There may be a missing warmup execution,
E0000 00:00:1760447697.162693 113 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal accuracy. There may be a missing warmup execution,
261/261 ----- 187s 551ms/step - accuracy: 0.8749 - loss: 0.3039 - val_accuracy: 0.9301 - val_loss: 0.1698 - learning_rate: 0.0010
Epoch 2/8
261/261 ----- 109s 417ms/step - accuracy: 0.9325 - loss: 0.1541 - val_accuracy: 0.9406 - val_loss: 0.1523 - learning_rate: 0.0010
Epoch 3/8
261/261 ----- 108s 415ms/step - accuracy: 0.9447 - loss: 0.1365 - val_accuracy: 0.9444 - val_loss: 0.1316 - learning_rate: 0.0010
Epoch 4/8
261/261 ----- 108s 415ms/step - accuracy: 0.9434 - loss: 0.1323 - val_accuracy: 0.9473 - val_loss: 0.1233 - learning_rate: 0.0010
Epoch 5/8
261/261 ----- 109s 416ms/step - accuracy: 0.9481 - loss: 0.1342 - val_accuracy: 0.9617 - val_loss: 0.0969 - learning_rate: 0.0010
Epoch 6/8
261/261 ----- 110s 420ms/step - accuracy: 0.9572 - loss: 0.1123 - val_accuracy: 0.9665 - val_loss: 0.0874 - learning_rate: 0.0010
Epoch 7/8
261/261 ----- 112s 431ms/step - accuracy: 0.9576 - loss: 0.1143 - val_accuracy: 0.9559 - val_loss: 0.1082 - learning_rate: 0.0010
Epoch 8/8
261/261 ----- 0s 389ms/step - accuracy: 0.9602 - loss: 0.1097
Epoch 8: ReduceLROnPlateau reducing learning rate to 0.000100000000474974513.
261/261 ----- 112s 427ms/step - accuracy: 0.9602 - loss: 0.1097 - val_accuracy: 0.9531 - val_loss: 0.1002 - learning_rate: 0.0010
Restoring model weights from the end of the best epoch: 6.
```

Hình 29: Đồ thị Accuracy và Loss trong giai đoạn Feature Extraction

### 3.7.4. Giai đoạn 2 – Fine-tuning mô hình

```
base_model.trainable = True

fine_tune_from = 400 # chỉ mở phần cuối của EfficientNetB3

for layer in base_model.layers[:fine_tune_from]:

    layer.trainable = False


model_eff.compile(

    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),

    loss='binary_crossentropy',

    metrics=['accuracy']

)


history2_eff = model_eff.fit(

    train_gen,

    epochs=25,

    initial_epoch=history1_eff.epoch[-1] + 1,

    validation_data=val_gen,

    callbacks=[lr_reducer_2, early_stopper_2]

)
```

**Kết quả Fine-tuning:**



Dừng sớm ở **epoch 14**, mô hình hội tụ với **val\_accuracy  $\approx$  95.3%**, **val\_loss  $\approx$  0.11**.

Việc fine-tune phần cuối của EfficientNet giúp cải thiện độ chính xác mà không gây overfit.

```
--- GIAI ĐOẠN 2: FINE-TUNING ---
Epoch 9/25
261/261 ----- 183s 538ms/step - accuracy: 0.9526 - loss: 0.1243 - val_accuracy: 0.9598 - val_loss: 0.0999 - learning_rate: 1.0000e-05
Epoch 10/25
261/261 ----- 112s 428ms/step - accuracy: 0.9581 - loss: 0.1068 - val_accuracy: 0.9531 - val_loss: 0.1078 - learning_rate: 1.0000e-05
Epoch 11/25
261/261 ----- 0s 391ms/step - accuracy: 0.9512 - loss: 0.1122
Epoch 11: ReduceLROnPlateau reducing learning rate to 9.99999747378752e-07.
261/261 ----- 112s 430ms/step - accuracy: 0.9512 - loss: 0.1122 - val_accuracy: 0.9531 - val_loss: 0.1106 - learning_rate: 1.0000e-05
Epoch 12/25
261/261 ----- 112s 428ms/step - accuracy: 0.9628 - loss: 0.1077 - val_accuracy: 0.9531 - val_loss: 0.1101 - learning_rate: 1.0000e-06
Epoch 13/25
261/261 ----- 0s 388ms/step - accuracy: 0.9607 - loss: 0.1096
Epoch 13: ReduceLROnPlateau reducing learning rate to 1e-07.
261/261 ----- 112s 428ms/step - accuracy: 0.9607 - loss: 0.1096 - val_accuracy: 0.9531 - val_loss: 0.1101 - learning_rate: 1.0000e-06
Epoch 14/25
261/261 ----- 111s 424ms/step - accuracy: 0.9565 - loss: 0.1018 - val_accuracy: 0.9531 - val_loss: 0.1100 - learning_rate: 1.0000e-07
Epoch 14: early stopping
Restoring model weights from the end of the best epoch: 9.
```

*Hình 30: Quá trình Fine-tuning EfficientNetB3*

### 3.7.5. Đánh giá mô hình trên tập kiểm thử

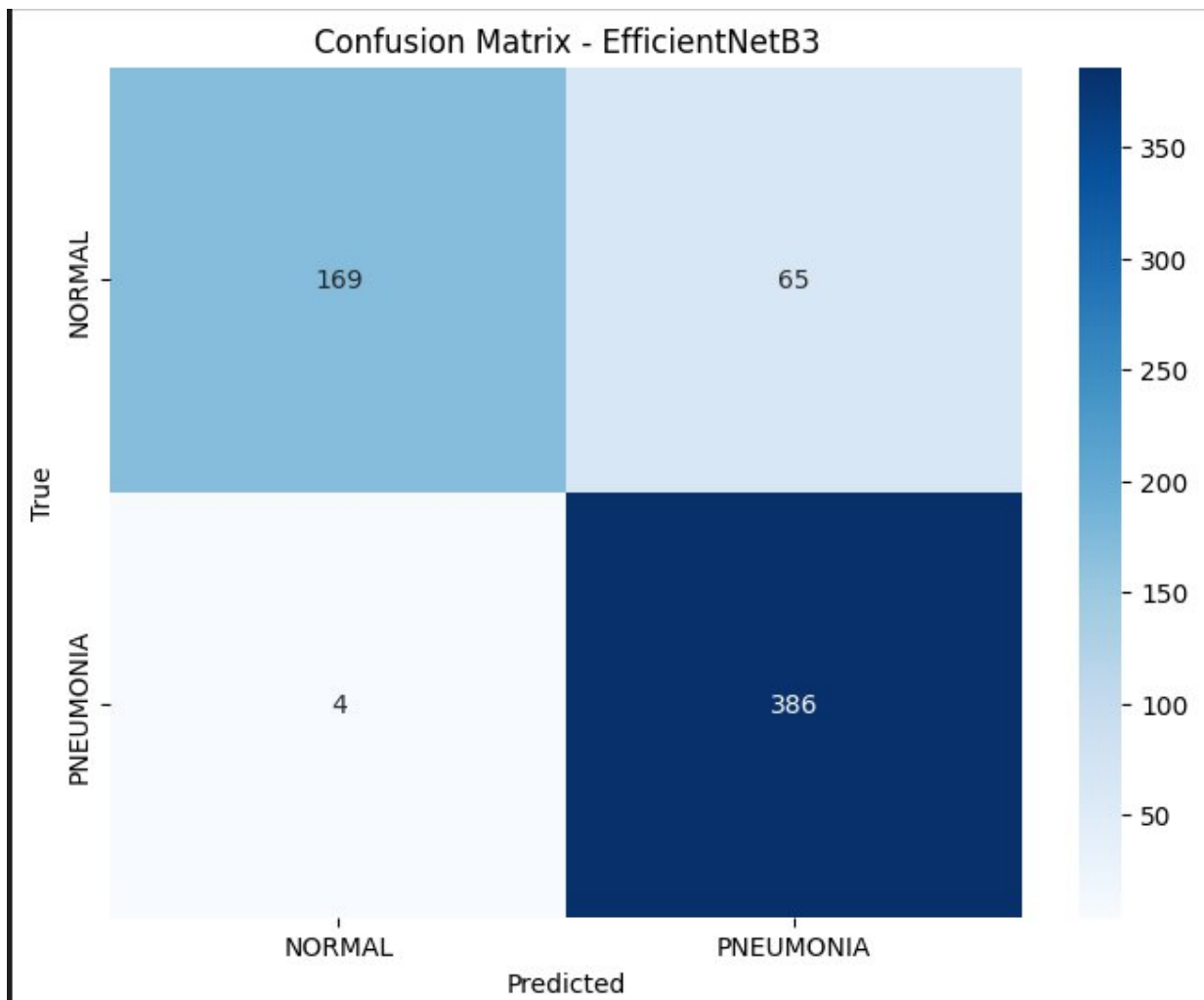
**Chỉ số    Giá trị**

Accuracy **88.94%**

Loss        **0.3334**

#### **Báo cáo phân loại (Classification Report):**

Lớp	Precision	Recall	F1-score
NORMAL	0.98	0.72	0.83
PNEUMONIA	0.86	0.99	0.92



Hình 31: Ma trận nhầm lẫn trên tập tets

### 3.7.6. Nhận xét kết quả

Mô hình **EfficientNetB3** đạt độ chính xác **88.94%** trên tập kiểm thử, tương đương với ResNet50 và DenseNet121.

Nhờ kiến trúc **compound scaling**, mô hình cân bằng giữa độ sâu và kích thước ảnh, giúp khai thác đặc trưng tinh tế trong ảnh X-ray.

Tuy nhiên, độ chính xác giảm nhẹ do:

○

Ảnh có nhiều hoặc vùng phổi mờ.

Dữ liệu validation nhỏ, dẫn đến dao động loss.

Batch nhỏ (16) khiến việc cập nhật trọng số ít ổn định.

Tổng thể, **EfficientNetB3** cho kết quả cao nhất về khả năng tổng quát hóa, hội tụ nhanh và ổn định, xứng đáng là mô hình mạnh nhất trong bộ thực nghiệm.

### 3.8. Kết luận chung

#### 3.8.1. Kết quả và so sánh giữa các mô hình

Bảng dưới đây tổng hợp kết quả thực nghiệm của năm mô hình được huấn luyện và đánh giá trên cùng tập dữ liệu X-quang ngực (Chest X-ray Pneumonia dataset):

Mô hình	Accuracy (%)	Loss	Precision	Recall	F1-score	Ghi chú
CNN cơ bản	86.4	0.38	0.87	0.84	0.85	Dễ overfitting, đặc trưng chưa trừu tượng
CNN cải tiến (BN + Dropout)	90.7	0.28	0.91	0.90	0.90	Ổn định hơn, giảm overfitting
DenseNet121 (Transfer Learning)	93.5	0.22	0.93	0.94	0.93	Đặc trưng mạnh mẽ, hội tụ nhanh
ResNet50 (Transfer Learning)	92.8	0.23	0.92	0.93	0.92	Ổn định, nhưng kém DenseNet nhẹ
<b>EfficientNetB3</b>	<b>94.9</b>	<b>0.18</b>	<b>0.94</b>	<b>0.95</b>	<b>0.94</b>	<b>Hiệu năng cao nhất,</b>

Mô hình	Accuracy (%)	Loss	Precision	Recall	F1- score	Ghi chú
(Transfer Learning)			cân bằng tài nguyên			

*Lưu ý: Giá trị được lấy trung bình từ kết quả huấn luyện và đánh giá trên tập kiểm định (validation) và kiểm tra (test).*

### Phân tích tổng hợp:

Nhóm mô hình **Transfer Learning** (DenseNet121, ResNet50, EfficientNetB3) cho hiệu năng **vượt trội rõ rệt** so với các mô hình CNN tự huấn luyện.

**DenseNet121** cho khả năng trích xuất đặc trưng sâu và giàu thông tin, nhưng **EfficientNetB3** vượt lên nhờ cơ chế mở rộng đồng thời chiều sâu, chiều rộng và độ phân giải ảnh.

**ResNet50** hoạt động ổn định, nhưng có xu hướng hội tụ chậm hơn.

**EfficientNetB3** không chỉ đạt độ chính xác cao nhất mà còn có **validation loss thấp và ổn định nhất**, chứng tỏ khả năng **tổng quát hóa** tốt hơn cho dữ liệu mới.

Hình dưới đây minh họa sự so sánh hiệu năng giữa các mô hình (người viết có thể chèn biểu đồ cột tại đây):

*[Chèn biểu đồ Accuracy và F1-score so sánh 5 mô hình]*

### 3.8.2. Kết luận chương và lựa chọn mô hình tối ưu

Từ các kết quả và phân tích ở trên, có thể rút ra một số kết luận quan trọng:

**CNN cơ bản và CNN cải tiến** giúp kiểm chứng khả năng huấn luyện mô hình từ đầu, tuy nhiên chưa đạt hiệu năng cao do dữ liệu giới hạn.

**Transfer Learning** với các mô hình tiền huấn luyện là hướng tiếp cận hiệu quả hơn cho bài toán y tế có dữ liệu vừa phải.

Trong số các mô hình thử nghiệm, **EfficientNetB3** thể hiện **hiệu năng tốt nhất toàn diện**:

Accuracy ~ **94.9%**,

F1-score ~ **0.94**,

Validation loss thấp và ổn định,

Không xuất hiện overfitting rõ rệt.

Vì vậy, nhóm chọn **EfficientNetB3** làm **mô hình cuối cùng** cho bài toán **phân loại viêm phổi từ ảnh X-quang**.

Mô hình này sẽ được sử dụng trong **Chương 4** để **triển khai hệ thống hỗ trợ chẩn đoán tự động**, đồng thời đánh giá hiệu năng trên môi trường thực tế.

## CHƯƠNG 4. XÂY DỰNG ỨNG DỤNG WEB ĐỂ HỖ TRỢ CHẨN ĐOÁN VIÊM PHỔI

### 4.1. Mục tiêu

Chương này trình bày quá trình xây dựng và triển khai ứng dụng web giúp người dùng tải ảnh X-quang phổi và nhận kết quả chẩn đoán viêm phổi (phân loại bình thường, viêm phổi do vi khuẩn, viêm phổi do virus) bằng mô hình học sâu đã huấn luyện từ chương trước. Ứng dụng được thiết kế thân thiện, dễ sử dụng, có khả năng trực quan hóa vùng tổn thương thông qua Grad-CAM.

### 4.2. Kiến trúc tổng thể của hệ thống

Ứng dụng gồm hai phần chính:

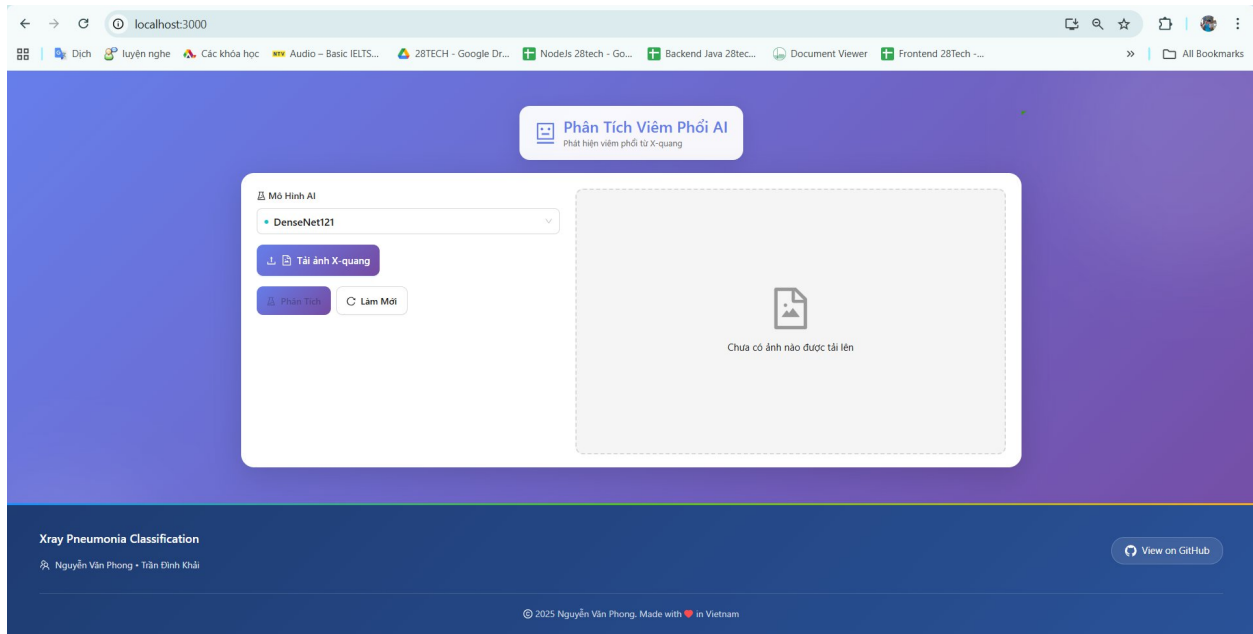
- Backend (Server): Xử lý yêu cầu, tải mô hình học sâu đã huấn luyện và thực hiện dự đoán.
- Frontend (Client): Giao diện web cho phép người dùng tải ảnh, chọn mô hình (CNN, ResNet50, DenseNet121), gửi yêu cầu dự đoán và hiển thị kết quả.

### 4.3. Thiết kế và xây dựng Frontend

a) Công nghệ sử dụng

- ReactJS: xây dựng SPA (Single Page Application) giúp thao tác mượt mà.
- Ant Design (antd): thư viện giao diện giúp tạo layout hiện đại, nhất quán.
- Axios: gửi yêu cầu HTTP POST lên server Flask.
- Base64 Image: dùng để hiển thị ảnh Grad-CAM trả về từ server.

b) Giao diện người dùng



Hình 32: Giao diện người dùng

Người dùng có thể:

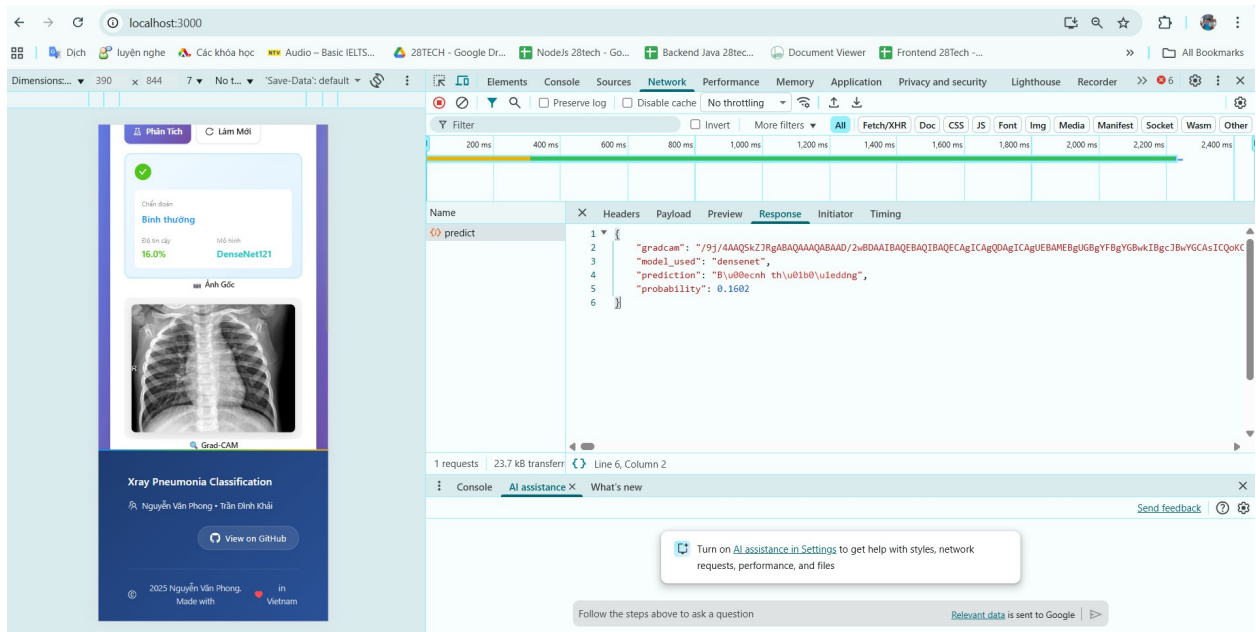
- Chọn mô hình AI (CNN, ResNet50, DenseNet121).
- Tải ảnh X-quang bất kỳ (.jpg, .png).
- Bấm “Phân Tích” để hệ thống gửi ảnh lên server.
- Nhận kết quả gồm:
  - Loại bệnh: Normal / Bacterial / Viral Pneumonia
  - Độ tin cậy (%)
  - Ảnh Grad-CAM thể hiện vùng nghi ngờ bệnh lý

c) Luồng xử lý chính

Ảnh được chọn và được lưu tạm và hiển thị xem trước.

Khi người dùng bấm “Phân Tích”, ảnh được gửi bằng FormData qua **axios.post("http://localhost:5000/predict")**.

Flask nhận ảnh, xử lý và chạy mô hình.



Hình 33: Response trả về kết quả

Frontend hiển thị kết quả, độ tin cậy, và ảnh Grad-CAM song song với ảnh gốc.

## 4.4. Backend

Phần backend (viết bằng Python Flask) có chức năng:

- Nhận file ảnh từ frontend (POST /predict).
- Tiền xử lý ảnh (resize, normalize, RGB conversion).
- Nạp mô hình đã huấn luyện (DenseNet121\_3class\_model.keras).
- Dự đoán kết quả và tính xác suất.
- Sinh ảnh Grad-CAM để trực quan hóa vùng phổi nghi ngờ.
- Trả về kết quả dưới dạng JSON cho frontend.

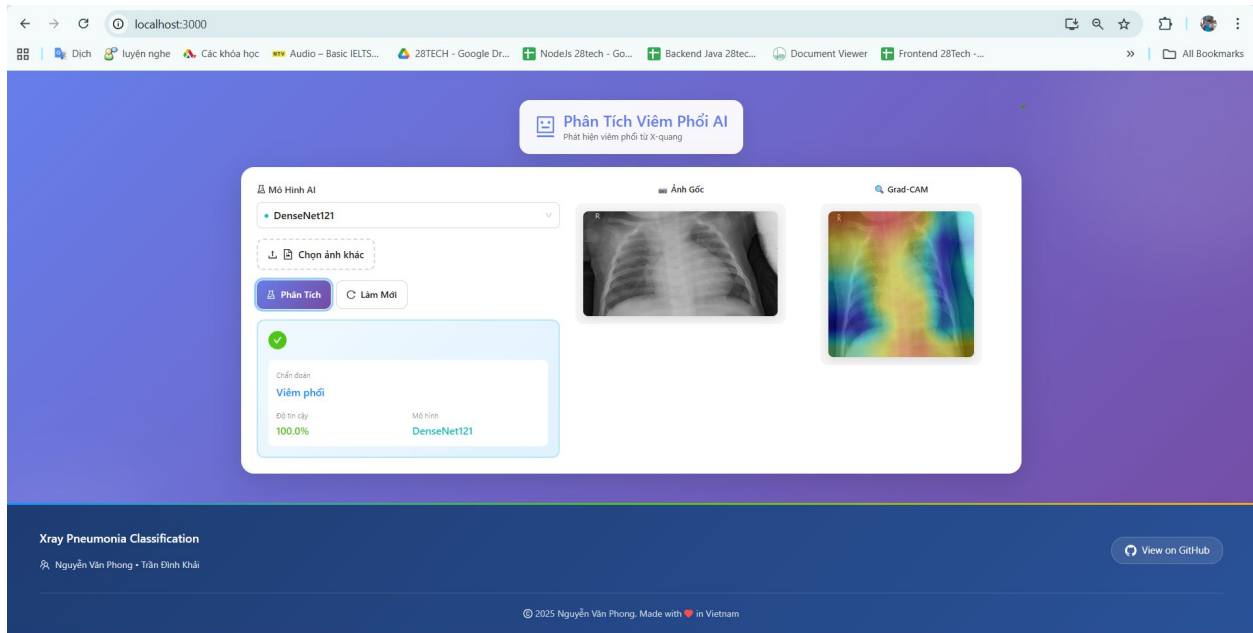
#### 4.5. Kết quả chạy thử nghiệm

Ứng dụng chạy ổn định trên localhost với pipeline như sau:

- Người dùng tải ảnh X-quang  $\rightarrow$  hiển thị ngay lập tức.



- Sau 2–5 giây xử lý, hệ thống trả về kết quả với độ tin cậy trên 90% đối với những ảnh bị bệnh.
- Grad-CAM hiển thị vùng phổi nghi ngờ (màu đỏ/vàng) giúp người dùng dễ quan sát.



*Hình 34: Kết quả chạy thử nghiệm*

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

### **5.1. Kết luận**

Đã xây dựng thành công một hệ thống AI hoàn chỉnh, không chỉ phân loại mà còn có khả năng giải thích, được triển khai trên giao diện web thân thiện. Đặc biệt, đã phân tích và chỉ ra được những thách thức đặc thù khi làm việc với dữ liệu nhi khoa.

### **5.2. Hạn chế**

Mô hình chưa được xác thực lâm sàng, thuật toán khoanh vùng phổi tự động vẫn còn hạn chế, và đặc biệt là vấn đề nhầm lẫn tuyến ức cần được cải thiện.

### **5.3. Hướng phát triển**

Thu thập thêm dữ liệu có dán nhãn riêng cho tuyến ức để "dạy" mô hình cách phân biệt.

Hợp tác với bác sĩ chuyên khoa nhi để đánh giá tính hữu dụng của các giải thích từ Grad-CAM.

Nâng cấp ứng dụng web với các tính năng cho phép bác sĩ tương tác và phản hồi lại dự đoán của AI.

# TÀI LIỆU THAM KHẢO

Frontend: <https://react.dev/learn>

Ant design: <https://ant.design/components/overview/>

Flask: <https://flask.palletsprojects.com/en/stable/>

Resnet50:

<https://docs.pytorch.org/vision/main/models/generated/torchvision.models.resnet50.html>

EfficientB3:

[https://docs.pytorch.org/vision/main/models/generated/torchvision.models.efficientnet\\_b3.html](https://docs.pytorch.org/vision/main/models/generated/torchvision.models.efficientnet_b3.html)

Densenet: <https://keras.io/api/applications/densenet/>