

# Hệ thống Trí tuệ Nhân tạo Hỗ trợ Chẩn đoán Viêm phổi

Bệnh viện Nhi đồng 2



## Dự án vì Cộng đồng

Tác giả: Lê Quốc Duy Quang

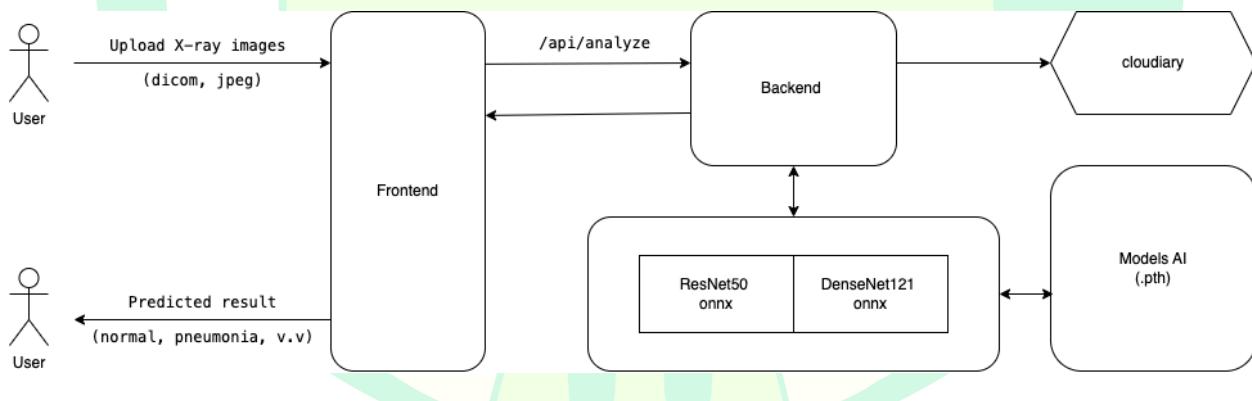
Cố vấn chuyên môn - Hỗ trợ Y khoa: Bác sĩ Trần Nam Hưng

Cố vấn chuyên môn - Hỗ trợ Kỹ thuật: Trần Quốc Tú, Trần Quốc Tuấn

Cố vấn đề tài: Trần Văn Hưng

## Tóm tắt

- Chẩn đoán viêm phổi ở trẻ em là một thách thức y khoa quan trọng, và trí tuệ nhân tạo (AI) mang lại tiềm năng to lớn trong việc cải thiện độ chính xác và hiệu quả. Nghiên cứu này trình bày việc phát triển và đánh giá một hệ thống AI nhằm phát hiện và phân loại viêm phổi trên ảnh X-quang ngực (CXR) ở trẻ em, được thực hiện như một dự án cộng đồng không vì mục đích thương mại, hướng đến các bệnh nhi tại Bệnh viện Nhi đồng 2.
- Hệ thống sử dụng hai mô hình học sâu tiên tiến là **ResNet50** và **DenseNet121**, áp dụng kỹ thuật học chuyển tiếp để thực hiện phân loại nhị phân (viêm phổi hoặc bình thường) và sau đó là phân loại đa nhãn các dấu hiệu/loại viêm phổi cụ thể.
- Dữ liệu được sử dụng từ hai nguồn công khai: Kaggle (bộ dữ liệu **Chest X-Ray Pneumonia**, được sử dụng toàn bộ cho ResNet50) và PhysioNet (bộ dữ liệu **VinDr-PCXR**, tập con 1000 ảnh với các nhãn được chọn lọc gồm được sử dụng cho nhiệm vụ đa nhãn của DenseNet), đặc biệt phù hợp với đối tượng trẻ em.
- Kiến trúc hệ thống bao gồm một backend NodeJS hỗ trợ tải ảnh lên (tích hợp với Cloudinary) và thực hiện suy luận bằng các mô hình AI đã được chuyển đổi từ định dạng PyTorch (.pth) sang ONNX (.onnx) để tối ưu hóa việc triển khai.
- Kiến trúc hệ thống:



- Một tính năng quan trọng của ứng dụng là cho phép bác sĩ nhập chẩn đoán ban đầu; hệ thống sẽ đưa ra cảnh báo nếu có sự khác biệt đáng kể giữa chẩn đoán của bác sĩ và dự đoán của AI, nhằm tăng cường sự an toàn và hỗ trợ ra quyết định.
- Việc huấn luyện mô hình được thực hiện trên nền tảng Google Colaboratory. Các kết quả đánh giá ban đầu cho thấy tiềm năng của hệ thống trong việc hỗ trợ chẩn đoán tại các cơ sở y tế như Bệnh viện Nhi đồng 2, thông qua việc cung cấp một công cụ chẩn đoán nhanh chóng, đáng tin cậy và có tính tương tác cao.

- Sự kết hợp giữa phân loại đa giai đoạn, so sánh hiệu năng các mô hình hiện đại, kiến trúc triển khai thực tiễn và cơ chế tương tác “bác sĩ-AI” là những đóng góp chính của nghiên cứu này.
- Sản phẩm đang chạy trên: <https://xray-ui.vercel.app/>

## 1. Giới thiệu

### 1.1. Bối cảnh: Trí tuệ Nhân tạo trong Chẩn đoán Bệnh lý Phổi

Trí tuệ nhân tạo (AI), đặc biệt là học sâu (deep learning), đang phát triển mạnh mẽ mang tính chuyên đổi trong nhiều lĩnh vực, bao gồm cả y tế. Trong phân tích hình ảnh y khoa, các công cụ dựa trên AI đang cách mạng hóa cách các bác sĩ X-quang diễn giải hình ảnh và đưa ra chẩn đoán.

Một ứng dụng quan trọng là trong việc chẩn đoán bệnh từ ảnh X-quang ngực (CXR), một phương pháp cận lâm sàng được sử dụng rộng rãi để phát hiện các bệnh lý như viêm phổi, lao và ung thư phổi. Việc diễn giải CXR thủ công đối mặt với nhiều thách thức, bao gồm tốn thời gian và có thể chịu sự biến thiên giữa các người đọc khác nhau, đặc biệt khi khối lượng công việc lớn.

AI, cụ thể là các mạng nơ-ron tích chập (Convolutional Neural Networks - CNNs), có khả năng học các mẫu phức tạp từ hình ảnh để xác định các bất thường với độ nhạy và độ đặc hiệu cao, thậm chí trong một số trường hợp có thể vượt qua cả các bác sĩ X-quang giàu kinh nghiệm. Điều này mở ra cơ hội cải thiện đáng kể chất lượng và tốc độ chẩn đoán.

Một trong những tác động quan trọng của các giải pháp AI là khả năng thu hẹp khoảng cách về chuyên môn y tế. Các hệ thống AI có thể cung cấp dịch vụ chẩn đoán chất lượng cao bất kể giới hạn địa lý, đặc biệt hữu ích tại các khu vực thiếu thốn nguồn lực hoặc thiếu bác sĩ X-quang nhi khoa chuyên sâu. Điều này có ý nghĩa to lớn đối với y tế toàn cầu và các cơ sở y tế hướng tới việc chuẩn hóa chất lượng chăm sóc.

### 1.2. Phát biểu Vấn đề: Viêm phổi Trẻ em, Thách thức Chẩn đoán

Viêm phổi là một trong những nguyên nhân hàng đầu gây tử vong ở trẻ em dưới năm tuổi trên toàn thế giới. Việc chẩn đoán viêm phổi ở trẻ em có những thách thức đặc thù. Diễn giải CXR ở trẻ em có thể khó khăn hơn do sự khác biệt về giải phẫu. Một vấn đề quan trọng khác là phân biệt giữa viêm phổi do vi khuẩn và viêm phổi do virus, vì hai loại này đòi hỏi các phương pháp điều trị rất khác nhau, nhưng thường khó phân biệt chỉ dựa trên CXR.

Trong bối cảnh của một bệnh viện nhi khoa chuyên sâu và có lưu lượng bệnh nhân cao như Bệnh viện Nhi đồng 2, nhu cầu về các công cụ hỗ trợ chẩn đoán tiên tiến để nâng cao hiệu quả và độ chính xác là rất lớn. Dự án này được thúc đẩy bởi mong muốn đóng góp cho cộng đồng, đặc biệt là hỗ trợ chăm sóc sức khỏe cho các bệnh nhi

tại Bệnh viện Nhi đồng 2, và hoàn toàn không vì mục đích lợi nhuận thương mại.

Nhu cầu chẩn đoán không chỉ dừng lại ở việc xác định có viêm phổi hay không (phân loại nhị phân), mà còn cần mô tả đặc điểm của bệnh (phân loại đa nhãn các loại hoặc các dấu hiệu đi kèm). Cách tiếp cận đa giai đoạn của dự án này – đầu tiên xác định viêm phổi, sau đó cung cấp thông tin chi tiết hơn – trực tiếp giải quyết nhu cầu lâm sàng về thông tin sâu sắc hơn là một kết quả "viêm phổi/không viêm phổi" đơn thuần.

### **1.3. Mục tiêu và Đóng góp của Dự án**

Mục tiêu chính của nghiên cứu này là phát triển và đánh giá một hệ thống dựa trên AI để tự động phát hiện và phân loại viêm phổi trên CXR trẻ em, sử dụng hai mô hình học sâu là ResNet50 và DenseNet121.

Các mục tiêu thứ cấp bao gồm:

- Tận dụng sức mạnh hiệu năng của ResNet50 và DenseNet cho nhiệm vụ cụ thể này.
- Triển khai một thành phần phân loại đa nhãn để xác định các dấu hiệu X-quang liên quan đến viêm phổi khi bệnh được phát hiện.
- Phát triển một kiến trúc triển khai thực tế sử dụng backend NodeJS và ONNX Runtime để suy luận hiệu quả.
- Tích hợp tính năng cho phép bác sĩ nhập chẩn đoán ban đầu và cung cấp cảnh báo khi có sự khác biệt lớn với dự đoán của AI, nhằm tăng cường sự tương tác và an toàn trong thực hành lâm sàng.

Những đóng góp chính của dự án này bao gồm: một công cụ AI mạnh mẽ được điều chỉnh cho viêm phổi trẻ em, một phân tích được triển khai bằng hai kiến trúc CNN hàng đầu, minh chứng về một chiến lược triển khai hiện đại cho AI trong lâm sàng, và một cơ chế tương tác “bác sĩ-AI” nhằm hỗ trợ quá trình ra quyết định. Tất cả được thực hiện với tinh thần phục vụ cộng đồng.

Với khối lượng bệnh nhân rất lớn như Bệnh viện Nhi đồng 2, việc ứng dụng và triển khai một hệ thống AI còn giúp cải thiện hiệu suất, giảm tải khối lượng công việc và tăng cường tính ứng dụng công nghệ thông tin, giúp Bệnh viện luôn đi đầu và là điểm đến đáng tin cậy cho mọi người.

## **2. Thiết kế Hệ thống và Phương pháp luận**

### **2.1. Bộ dữ liệu**

Việc lựa chọn và xử lý dữ liệu là nền tảng cho bất kỳ dự án AI nào, đặc biệt trong lĩnh vực y tế. Nghiên cứu này sử dụng hai bộ dữ liệu công khai, được chọn lọc dựa trên tính phù hợp với đối tượng trẻ em và mức độ chi tiết của chú thích.

### **2.1.1. Kaggle: Bộ dữ liệu Ảnh X-quang Ngực (Viêm phổi)**

- **Nguồn gốc:** Bộ dữ liệu này, được biết đến với tên "Chest X-Ray Images (Pneumonia)", ban đầu được công bố bởi **Kermany** và cộng sự (2018) trên tạp chí Cell và được lưu trữ trên Mendeley Data.
- **Nội dung:** Bao gồm 5,863 ảnh CXR định dạng JPEG chụp theo tư thế trước-sau (anterior-posterior) của các bệnh nhi từ một đến năm tuổi tại Trung tâm Y tế Phụ nữ và Trẻ em Quảng Châu. Toàn bộ bộ dữ liệu này được sử dụng để huấn luyện và đánh giá mô hình ResNet50.
- **Phân loại:** Dữ liệu được tổ chức thành hai nhóm chính: NORMAL (Bình thường) và PNEUMONIA (Viêm phổi). Theo mô tả trong bài báo gốc, nhóm PNEUMONIA bao gồm cả viêm phổi do vi khuẩn và virus.
- **Kiểm soát chất lượng:** Các ảnh X-quang ban đầu đã được sàng lọc để loại bỏ những ảnh chất lượng thấp hoặc không đọc được. Chẩn đoán cho các ảnh được thực hiện bởi hai bác sĩ chuyên khoa giàu kinh nghiệm, và bộ dữ liệu đánh giá còn được kiểm tra bởi một chuyên gia thứ ba để đảm bảo tính chính xác.

### **2.1.2. PhysioNet: Bộ dữ liệu VinDr-PCXR**

- **Nguồn gốc:** Bộ dữ liệu này có tên "VinDr-PCXR: An open, large-scale pediatric chest X-ray dataset for interpretation of common thoracic diseases in children", được công bố bởi **Hieu Huy Pham , Tien Thanh Tran , Ha Quy Nguyen**.
- **Nội dung:** Chứa 9,125 ảnh CXR định dạng DICOM của các bệnh nhi dưới mười tuổi, được thu thập hồi cứu từ một bệnh viện nhi lớn tại Việt Nam. Việc bộ dữ liệu này có nguồn gốc từ Việt Nam và tập trung vào đối tượng trẻ em làm tăng tính liên quan của nó đối với Bệnh viện Nhi đồng 2.
- **Chú thích và Sử dụng cho DenseNet:** Mỗi ảnh được chú thích thủ công bởi một bác sĩ X-quang nhi khoa có hơn mươi năm kinh nghiệm, bao gồm 36 dấu hiệu X-quang nguy kịch (nhãn cục bộ, được khoanh vùng bằng bounding box) và 15 bệnh lý (nhãn toàn cục). Đối với nhiệm vụ phân loại đa nhãn của mô hình DenseNet trong dự án này, một tập con gồm 1000 ảnh đã được chọn lọc từ bộ VinDr-PCXR. Các nhãn cho tập con này cũng được tinh lọc, tập trung vào các bệnh lý phổ biến hơn, bao gồm:
  - Viêm phế quản (Bronchitis)
  - Viêm phế quản-phổi (Brocho-pneumonia)
  - Bệnh khác (Other disease)
  - Viêm tiểu phế quản (Bronchiolitis)
  - Viêm phổi (Pneumonia).

Việc lọc bỏ các nhãn hiếm gặp nhằm mục đích cải thiện hiệu suất của mô hình trên

các lớp có đủ số lượng mẫu.

- **Phân chia dữ liệu:** Bộ dữ liệu gốc được chia thành 7,728 ảnh cho tập huấn luyện và 1,397 ảnh cho tập kiểm tra. Việc phân chia cho tập con 1000 ảnh được thực hiện tương ứng.

Việc sử dụng hai bộ dữ liệu này mang tính bổ trợ. Bộ dữ liệu Kaggle cung cấp một nền tảng tốt cho việc phân loại nhì phổi ở trẻ nhỏ với mô hình ResNet50. Trong khi đó, một tập dữ liệu chọn lọc từ VinDr-PCXR, với đặc điểm là dữ liệu nhi khoa từ Việt Nam và các chủ thích đa nhãn đã được tinh lọc, được sử dụng cho nhiệm vụ phân loại đa nhãn sâu hơn của mô hình DenseNet, tăng thêm tính phù hợp khu vực và tập trung vào các bệnh lý cụ thể.

**Bảng 1: Đặc điểm các Bộ dữ liệu**

Đặc điểm	Kaggle: Chest X-Ray Images (Pneumonia)	PhysioNet: VinDr-PCXR (Tập con cho DenseNet)
Tên bộ dữ liệu	Chest X-Ray Images (Pneumonia)	VinDr-PCXR (Tập con 1000 ảnh)
Nguồn	Kermany et al. (2018), Cell; Mendeley Data	Pham et al.; PhysioNet
Tổng số ảnh sử dụng	5,863 (Toàn bộ cho ResNet50)	1,000 (Cho DenseNet đa nhãn)
Số ảnh Bình thường	1,349	Không áp dụng trực tiếp (đa nhãn)
Số ảnh Viêm phổi (Kaggle)	3,883 (2,538 vi khuẩn, 1,345 virus)	Không áp dụng trực tiếp
Phân chia Train/Test	Train/Test/Val theo cấu trúc thư mục gốc	Phân chia từ tập con 1000 ảnh
Độ tuổi bệnh nhân	1-5 tuổi	< 10 tuổi (từ bộ gốc)

<b>Định dạng ảnh</b>	JPEG	DICOM (từ bộ gốc, xử lý thành định dạng phù hợp)
<b>Độ phân giải</b>	Biến đổi (cần chuẩn hóa)	Biến đổi (cần chuẩn hóa)
<b>Số lượng dấu hiệu chú thích (Kaggle)</b>	2 (Normal, Pneumonia)	5 nhãn chính cho DenseNet: Viêm phế quản, Viêm phế quản-phổi, Bệnh khác, Viêm tiểu phế quản, Viêm phổi

### 2.1.3. Tiền xử lý và Tăng cường Dữ liệu

Tiền xử lý dữ liệu là một bước không thể bỏ qua và có vai trò quan trọng đối với hiệu năng của mô hình AI Y tế.

- Thay đổi kích thước ảnh:** Kích thước ảnh đầu vào được chuẩn hóa, ví dụ 224x224 pixel, là kích thước phổ biến cho các mô hình được tiền huấn luyện trên ImageNet như ResNet50 và DenseNet121.
- Chuẩn hóa:** Giá trị pixel được chuẩn hóa bằng cách sử dụng trung bình và độ lệch chuẩn của bộ ImageNet nếu sử dụng các mô hình tiền huấn luyện. Các giá trị này thường là Trung bình: [0.485, 0.456, 0.406] và Độ lệch chuẩn: [0.229, 0.224, 0.225] cho ảnh RGB 3 kênh. Ảnh CXR thường là ảnh xám; báo cáo này cần làm rõ cách chúng được xử lý (ví dụ: chuyển đổi sang 3 kênh bằng cách nhân bản kênh hoặc các mô hình được điều chỉnh cho đầu vào một kênh).
- Tăng cường dữ liệu (Data Augmentation):** Các kỹ thuật như lật ngang/dọc, xoay ảnh, dịch chuyển, thu phóng được áp dụng để tăng kích thước bộ dữ liệu một cách nhân tạo và giảm thiểu hiện tượng quá khớp (overfitting). Các lựa chọn tăng cường cần phù hợp về mặt y tế (ví dụ, xoay ảnh quá mức có thể không hợp lý).
- Xử lý DICOM (cho VinDr-PCXR):** Các bước cần thiết để trích xuất dữ liệu pixel từ tệp DICOM và áp dụng các kỹ thuật điều chỉnh cửa sổ/mức độ (windowing/leveling) nếu chưa được chuẩn hóa. Các nhà tạo bộ dữ liệu đã thực hiện khử định danh và lọc dữ liệu.
- Phân chia dữ liệu:** Chi tiết về tỷ lệ phân chia tập huấn luyện/kiểm định/kiểm tra (ví dụ: 80%/10%/10% như trong). Việc phân chia dữ liệu theo cấp độ bệnh nhân (patient-level splitting) được nhấn mạnh để ngăn chặn rò rỉ dữ liệu, đảm bảo rằng tất cả ảnh của một bệnh nhân chỉ xuất hiện trong một tập duy nhất.

Sự tiền xử lý kỹ lưỡng, bao gồm chuẩn hóa cường độ, xử lý độ sâu bit của ảnh và thay

đổi kích thước phù hợp, là tối quan trọng. Lựa chọn phương pháp chuẩn hóa (ví dụ: sử dụng thống kê của ImageNet so với thống kê riêng của bộ dữ liệu) có thể ảnh hưởng đến hiệu suất học chuyển tiếp.

## 2.2. Kiến trúc Mô hình AI và Huấn luyện

Nghiên cứu này tập trung vào hai kiến trúc CNN nổi tiếng: ResNet50 và DenseNet121, cả hai đều được huấn luyện bằng kỹ thuật học chuyển tiếp.

### 2.2.1. Kiến trúc ResNet50

- **Tổng quan:** ResNet50 là một mạng nơ-ron tích chập sâu 50 lớp, nổi tiếng về hiệu quả trong phân loại hình ảnh và khả năng huấn luyện các mạng rất sâu
- **Các thành phần chính:**
  - Các lớp tích chập và gộp (pooling) ban đầu.
  - Khối dư (Residual Blocks): Đơn vị xây dựng cốt lõi. Các khối này hoạt động với các kết nối tắt (skip connections) – bao gồm kết nối đồng nhất (identity shortcuts) và kết nối chiều/tích chập (projection/convolutional shortcuts) – để cộng đầu vào của một khối với đầu ra của nó.
  - Thiết kế cổ chai (Bottleneck Design): Trong các khối dư, sử dụng các lớp tích chập  $1 \times 1$ ,  $3 \times 3$ , và  $1 \times 1$  để giảm rò khôi phục số chiều, cải thiện hiệu quả tính toán.
  - Chuẩn hóa theo lô (Batch Normalization) và hàm kích hoạt ReLU: Được áp dụng sau các lớp tích chập.
  - Lớp gộp trung bình toàn cục (Global Average Pooling) và lớp kết nối đầy đủ (Fully Connected Layer): Cho việc phân loại cuối cùng.
- **Giải quyết vấn đề suy giảm độ dốc (Vanishing Gradient):** Các kết nối tắt cho phép gradient lan truyền dễ dàng hơn qua mạng, giảm thiểu vấn đề suy giảm độ dốc và cho phép xây dựng các kiến trúc sâu hơn.

### 2.2.2. Kiến trúc DenseNet121

- **Tổng quan:** DenseNet (Densely Connected Convolutional Networks) là kiến trúc mà mỗi lớp kết nối với mọi lớp khác theo kiểu truyền thẳng (feed-forward).
- **Các thành phần chính:**
  - Khối dày đặc (Dense Blocks): Các nhóm lớp nơi mỗi lớp nhận bản đồ đặc trưng (feature maps) từ tất cả các lớp trước đó và truyền bản đồ đặc trưng của chính nó tới tất cả các lớp tiếp theo thông qua việc ghép nối (concatenation).
  - Tốc độ tăng trưởng (Growth Rate - k): Mỗi lớp thêm ' $k$ ' bản đồ đặc trưng vào "kiến thức tập thể". Điều này thúc đẩy hiệu quả tham số vì ' $k$ ' có thể nhỏ.
  - Lớp chuyển tiếp (Transition Layers): Nằm giữa các khối dày đặc, các lớp này thực hiện tích chập (thường là  $1 \times 1$ ) và gộp (ví dụ: gộp trung bình  $2 \times 2$ ) để giảm

kích thước bản đồ đặc trưng và kiểm soát độ phức tạp của mô hình.

- Lớp cổ chai (Bottleneck Layers - DenseNet-B): Thường thì các lớp tích chập 1x1 được sử dụng trước các lớp tích chập 3x3 trong các lớp dày đặc để giảm số lượng bản đồ đặc trưng đầu vào, cải thiện hiệu quả tính toán.
- Hệ số nén (Compression Factor): Trong các lớp chuyển tiếp, số lượng bản đồ đặc trưng có thể được giảm bởi một hệ số nén (ví dụ: 0.5 trong DenseNet-C).
- **Lợi ích:** Giảm thiểu suy giảm độ dốc, tăng cường lan truyền đặc trưng, khuyến khích tái sử dụng đặc trưng và giảm số lượng tham số.

### 2.2.3. Chiến lược Học chuyển tiếp (Transfer Learning)

- **Trọng số tiền huấn luyện:** Cả hai mô hình ResNet50 và DenseNet đều được khởi tạo với các trọng số đã được huấn luyện trước trên bộ dữ liệu ImageNet.
- **Phương pháp tinh chỉnh (Fine-tuning):**
  - Loại bỏ lớp phân loại gốc (classifier head).
  - Thêm các lớp phân loại mới phù hợp cho nhiệm vụ nhị phân (BÌNH THƯỜNG/VIỆM PHỔI) và đa nhãn (ví dụ: lớp GlobalAveragePooling2D theo sau bởi một lớp Dense với hàm kích hoạt sigmoid cho phân loại đa nhãn).
  - Cần chi tiết hóa các lớp nào của mô hình nền tiền huấn luyện được giữ cố định (frozen) và lớp nào được giải phóng (unfrozen) để huấn luyện. Điều này rất quan trọng để thích ứng với miền dữ liệu y tế.
- **Thảo luận và Lý giải:**
  - Ưu điểm: Tận dụng kiến thức từ một bộ dữ liệu lớn, có khả năng dẫn đến hội tụ nhanh hơn và hiệu suất tốt hơn với dữ liệu y tế hạn chế.

### 2.2.4. Tham số Huấn luyện

- **Trình tối ưu hóa (Optimizer):** Adam.
- **Hàm mất mát (Loss Function):**
  - Phân loại nhị phân: Binary Cross-Entropy và FocalLoss
  - Phân loại đa nhãn: Binary Cross-Entropy và FocalLoss
- **Tốc độ học (Learning Rate):** Tốc độ học ban đầu và bất kỳ lịch trình điều chỉnh tốc độ học nào được sử dụng (ví dụ: ReduceLROnPlateau).
- **Kích thước lô (Batch Size):** 16
- **Số ký nguyên (Epochs):** 20
- **Callbacks:** Early Stopping để ngăn chặn quá khớp, ReduceLROnPlateau.
- **Phần cứng và Môi trường Huấn luyện:** Việc huấn luyện mô hình được thực hiện trên nền tảng Google Colaboratory (phiên bản miễn phí), không yêu cầu GPU NVIDIA chuyên dụng.

**Bảng 2: Cấu hình Mô hình và Siêu tham số Huấn luyện**

Đặc điểm	ResNet50	DenseNet121
Mô hình nền	ResNet50	DenseNet121
Tiền huấn luyện	ImageNet	ImageNet
Chiến lược tinh chỉnh	Thay thế lớp đầu ra, xác định số lớp conv được giải phóng/giữ cố định	Thay thế lớp đầu ra, xác định số lớp conv được giải phóng/giữ cố định
Trình tối ưu hóa	Adam	Adam
Tốc độ học	(Giá trị ban đầu, lịch trình nếu có)	(Giá trị ban đầu, lịch trình nếu có)
Hàm mất mát (Nhị phân)	Focal Loss và Binary Cross-Entropy	Focal Loss và Binary Cross-Entropy
Hàm mất mát (Đa nhãn)	Không áp dụng trực tiếp (ResNet50 tập trung vào nhị phân)	Binary Cross-Entropy (cho mỗi nhãn trong 5 nhãn đã chọn)
Kích thước lô	16	16
Số ký nguyên	20	20
Callbacks chính	EarlyStopping, ReduceLROnPlateau	EarlyStopping, ReduceLROnPlateau
Môi trường huấn luyện	Google Colaboratory (miễn phí)	Google Colaboratory (miễn phí)

### 2.3. Triển khai Hệ thống

Kiến trúc triển khai của hệ thống được thiết kế để đảm bảo tính thực tiễn và hiệu quả, bao gồm backend NodeJS, dịch vụ lưu trữ ảnh Cloudinary và công cụ suy luận ONNX Runtime.

### 2.3.1. Kiến trúc Backend: NodeJS với Express.js

- **Lựa chọn NodeJS:** Việc sử dụng NodeJS được lý giải bởi kiến trúc I/O không chặn (non-blocking I/O), phù hợp để xử lý đồng thời nhiều yêu cầu tải ảnh lên và suy luận, cùng với hệ sinh thái JavaScript phong phú.
- **Framework Express.js:** Được sử dụng để xây dựng các API RESTful cho việc tải ảnh và yêu cầu dự đoán.
- **Các điểm cuối API (API Endpoints):** Mô tả các điểm cuối API chính (ví dụ: /analyze).

### 2.3.2. Xử lý Ảnh: Tích hợp Cloudinary

- **Mục đích:** Giảm tải việc lưu trữ và quản lý ảnh cho một dịch vụ đám mây của bên thứ ba.
- **Cơ chế:** Sử dụng Cloudinary NodeJS SDK để tải ảnh nhận được tại API backend. Backend nhận ảnh, tải lên Cloudinary và nhận lại URL hoặc định danh của ảnh.
- **Lợi ích:** Khả năng mở rộng, độ tin cậy cho việc lưu trữ ảnh, tiềm năng cho các tính năng chuyển đổi/phân phối ảnh nếu được sử dụng.

### 2.3.3. Triển khai Mô hình: Chuyển đổi từ PyTorch sang ONNX

- **Lý do chọn ONNX:** Định dạng ONNX (Open Neural Network Exchange) được chọn vì tính tương thích và khả năng triển khai mô hình trên các framework và phần cứng khác nhau. ONNX cho phép các mô hình được huấn luyện bằng Python (tệp.pth của PyTorch) có thể chạy trong các môi trường không phải Python như NodeJS.
- **Quá trình chuyển đổi:** Mô tả ngắn gọn các bước sử dụng torch.onnx.export(), bao gồm việc đặt mô hình ở chế độ đánh giá (evaluation mode) và cung cấp một đầu vào giả (dummy input).
- **Lợi ích:** Tương thích framework, hiệu quả triển khai, tiềm năng tăng tốc phần cứng thông qua ONNX Runtime.

### 2.3.4. Công cụ Suy luận: ONNX Runtime trong NodeJS

- **ONNX Runtime:** Là một công cụ suy luận hiệu năng cao cho các mô hình ONNX.
- **onnxruntime-node:** Là gói thư viện NodeJS cho ONNX Runtime, cho phép thực thi mô hình trực tiếp trong môi trường JavaScript.
- **Quá trình suy luận:** Backend NodeJS tải mô hình.onnx bằng onnxruntime-node, tiền xử lý ảnh đầu vào (lấy từ URL Cloudinary hoặc truyền trực tiếp sau khi tải lên),

chạy suy luận và xử lý đầu ra.

- **Cân nhắc về hiệu năng:** Thảo luận về các lợi ích hiệu năng tiềm năng của ONNX Runtime (ví dụ: khả năng tăng tốc phần cứng, các tối ưu hóa nội bộ) so với các phương pháp khác, chẳng hạn như chạy một tiến trình con Python để suy luận.

Sự lựa chọn NodeJS kết hợp với ONNX Runtime là một quyết định kỹ thuật quan trọng. Tính chất bắt đồng bộ của NodeJS rất phù hợp cho các tác vụ liên quan đến I/O như xử lý tải ảnh và yêu cầu API. ONNX Runtime, với nền tảng C++ được tối ưu hóa, có khả năng cung cấp hiệu năng suy luận tốt hơn so với một máy chủ suy luận dựa trên Python trong một số trường hợp nhất định, đặc biệt nếu Global Interpreter Lock (GIL) của Python trở thành điểm nghẽn trong một máy chủ Python đa luồng. Kiến trúc này hướng đến sự cân bằng giữa tiện lợi trong phát triển (sử dụng stack JavaScript) và hiệu quả khi chạy thực tế.

Hơn nữa, mặc dù hệ thống hiện tại được thiết kế cho máy chủ, ONNX và ONNX Runtime rất phù hợp cho các thiết bị biên (edge devices). Điều này mở ra một lộ trình tiềm năng nếu các phiên bản tương lai của hệ thống cần chạy trên phần cứng cục bộ tại bệnh viện hoặc các thiết bị di động có tài nguyên hạn chế.

### 2.3.5. *Tính năng Tương tác Bác sĩ-AI và Cảnh báo*

Một thành phần quan trọng của ứng dụng là cơ chế tương tác giữa bác sĩ và hệ thống AI. Giao diện người dùng cho phép bác sĩ nhập chẩn đoán sơ bộ của mình trước khi AI xử lý ảnh. Sau khi AI đưa ra dự đoán, hệ thống sẽ so sánh kết quả này với chẩn đoán của bác sĩ. Nếu phát hiện có sự khác biệt đáng kể (ví dụ, bác sĩ chẩn đoán "Bình thường" trong khi AI phát hiện "Viêm phổi" với độ tin cậy cao, hoặc ngược lại, hoặc có sự khác biệt lớn trong các nhãn đa bệnh lý), một cảnh báo (warning) sẽ được hiển thị. Mục đích của cảnh báo này là:

- **Nâng cao nhận thức:** Thu hút sự chú ý của bác sĩ đến những trường hợp có khả năng bất đồng, khuyến khích việc xem xét kỹ lưỡng hơn.
- **Hỗ trợ học tập:** Có thể giúp bác sĩ đổi chiều và học hỏi từ những trường hợp mà AI phát hiện ra các dấu hiệu khó nhận biết.
- **Tăng cường an toàn:** Giảm thiểu nguy cơ bỏ sót chẩn đoán hoặc chẩn đoán sai do yếu tố chủ quan, bằng cách cung cấp một "ý kiến thứ hai" từ AI. Cơ chế xác định "độ chênh lệch khác nhau nhiều" cần được định nghĩa rõ ràng, có thể dựa trên ngưỡng xác suất của AI, sự khác biệt về loại bệnh lý chính, hoặc số lượng nhãn không khớp trong trường hợp đa nhãn.

## 2.4. Nhiệm vụ Phân loại

Hệ thống thực hiện chẩn đoán theo một quy trình đa giai đoạn:

### 2.4.1. *Phân loại Nhị phân: BÌNH THƯỜNG hoặc VIÊM PHỔI*

- Đây là bước chẩn đoán chính, chủ yếu được thực hiện bởi mô hình ResNet50 sử dụng toàn bộ bộ dữ liệu Kaggle.
- Đầu vào: Ảnh CXR đã được tiền xử lý.
- Đầu ra: Một điểm xác suất và một nhãn nhị phân (BÌNH THƯỜNG/VIÊM PHỔI).
- Mô hình sử dụng: ResNet50

#### **2.4.2. Phân loại Đa nhãn cho các Trường hợp Viêm phổi**

- Được kích hoạt nếu kết quả phân loại nhị phân là VIÊM PHỔI. Nhiệm vụ này được thực hiện chủ yếu bởi mô hình DenseNet.
- Mục tiêu: Cung cấp thông tin chi tiết hơn về tình trạng viêm phổi.
- Đầu vào: Ảnh CXR được xác định là VIÊM PHỔI.
- Đầu ra: Một tập hợp các nhãn tương ứng với sự hiện diện/vắng mặt của các dấu hiệu phổi được xác định trước.
- Bộ nhãn (cho DenseNet): Dựa trên tập con 1000 ảnh từ VinDr-PCXR, các nhãn được sử dụng là: **Viêm phế quản (Bronchitis)**, **Viêm phế quản-phổi (Brocho-pneumonia)**, **Bệnh khác (Other disease)**, **Viêm tiểu phế quản (Bronchiolitis)**, và **Viêm phổi (Pneumonia)**.

Đầu ra đa nhãn giúp tăng cường đáng kể tiện ích lâm sàng. Thay vì chỉ kết luận "viêm phổi", hệ thống có thể chỉ ra các bệnh lý hoặc đặc điểm cụ thể hơn, từ đó hướng dẫn điều trị hiệu quả hơn. Điều này phù hợp với nhu cầu phân biệt các loại viêm phổi hoặc xác định các biến chứng.

### **2.5. Thước đo Đánh giá**

#### **2.5.1. Cho Phân loại Nhị phân (BÌNH THƯỜNG và VIÊM PHỔI)**

- **Các thước đo:** Accuracy (Độ chính xác)
- **Lý giải:**
  - **Accuracy:** Đo lường tỷ lệ dự đoán đúng tổng thể, nhưng có thể gây hiểu lầm trong các bộ dữ liệu mất cân bằng.

#### **2.5.2. Cho Phân loại Đa nhãn**

- **Các thước đo:**
  - **Train loss:** Độ lỗi (loss) trên tập huấn luyện sau epoch này là 0.0156. Loss càng thấp cho thấy mô hình học tốt hơn trên tập train.
  - **Val loss:** Độ lỗi trên tập validation.
  - **F1 score:** trung bình điều hòa giữa precision và recall, phù hợp cho bài toán phân loại mất cân bằng.
- **Lý giải:** Việc lựa chọn các thước đo này phản ánh sự phức tạp của nhiệm vụ đa

nhãn. Không chỉ là "đúng" hay "sai" cho toàn bộ mẫu, mà cần đánh giá mức độ chính xác trên từng nhãn riêng lẻ và tổng thể.

## 2.6. Môi trường Phát triển và Mã nguồn

Dự án được phát triển với các công cụ và nền tảng sau:

- **Huấn luyện Mô hình:** Google Colaboratory (phiên bản miễn phí).
- **Ngôn ngữ Lập trình Chính (AI/Backend):** Python (cho mô hình AI), NodeJS (cho backend).
- **Mã nguồn:**
  - AI và Backend: <https://github.com/lequocduyquang/xray-diagnosis-ai>
  - Frontend: <https://github.com/lequocduyquang/xray-ui>

## 3. Kết quả

### 3.1. Hiệu năng Phân loại Nhị phân (BÌNH THƯỜNG và VIÊM PHỔI)

Kết quả cho nhiệm vụ phân loại cơ bản giữa ảnh X-quang ngực bình thường và ảnh có dấu hiệu viêm phổi được trình bày trong

Bảng 3: Hiệu năng các Mô hình trong Phân loại Nhị phân

Thước đo	ResNet50 trên tập data của Kaggle	ResNet50 trên tập data của PhysioNet
Accuracy	88%	97.65%

#### 3.1.1 Kết quả accuracy trên data của Kaggle

```

Epoch 2/10, Loss: 0.13447300013447741
Validation Accuracy: 0.94
Epoch 3/10, Loss: 0.15414980597839764
Validation Accuracy: 0.88
Epoch 4/10, Loss: 0.136748831062092
Validation Accuracy: 0.75
Epoch 5/10, Loss: 0.1402266959036969
Validation Accuracy: 0.69
Epoch 6/10, Loss: 0.1316373597921952
Validation Accuracy: 0.88
Epoch 7/10, Loss: 0.14173146946344273
Validation Accuracy: 0.81
Epoch 8/10, Loss: 0.11756931941819154
Validation Accuracy: 0.81
Epoch 9/10, Loss: 0.11750116240964528
Validation Accuracy: 0.75
Epoch 10/10, Loss: 0.12901479451477163
Validation Accuracy: 0.88
Model saved to models/resnet50-pneumonia.pth

[ ] !python /content/drive/MyDrive/xray-diagnosis-ai/test_model.py

/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter warnings.warn(
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:223: UserWarning: Argument warnings.warn(msg)
Test Loss: 0.5706, Test Accuracy: 82.85%

```

New section

Connected to Python 3 Google Compute Engine backend

Resources

You are not subscribed. Learn more  
You currently have zero compute units available. Resources offered free of charge are not guaranteed. Purchase more units here.  
At your current usage level, this runtime may last up to 31 hours 20 minutes.

Manage sessions

Want more memory and disk space? Upgrade to Colab Pro

Python 3 Google Compute Engine backend  
Showing resources since 07:31

System RAM 1.0 / 12.7 GB  
Disk 37.1 / 107.7 GB

Change runtime type

### 3.1.2 Kết quả accuracy trên data của PhysioNet

```

Converting train folder...
Conversion complete!

[ ] !python3 /content/drive/MyDrive/xray-diagnosis-ai/resnet50/v2/train_resnet50_model_kid.py

Using device: cpu
Removed 0 .DS_Store files.
Class to index mapping: {'NORMAL': 0, 'PNEUMONIA': 1}
Removed 0 .DS_Store files.
Class to index mapping: {'NORMAL': 0, 'PNEUMONIA': 1}
[Epoch 1/20] Train Loss: 0.6486 | Val Loss: 0.6177 | Val Acc: 0.6353
✓ Saved best model to /content/drive/MyDrive/xray-diagnosis-ai/resnet50/v2/models/resnet50_model_v2_kid.pth (Val Acc: 0.6353)
[Epoch 2/20] Train Loss: 0.6133 | Val Loss: 0.5787 | Val Acc: 0.7029
✓ Saved best model to /content/drive/MyDrive/xray-diagnosis-ai/resnet50/v2/models/resnet50_model_v2_kid.pth (Val Acc: 0.7029)
[Epoch 3/20] Train Loss: 0.5631 | Val Loss: 0.5201 | Val Acc: 0.7324
✓ Saved best model to /content/drive/MyDrive/xray-diagnosis-ai/resnet50/v2/models/resnet50_model_v2_kid.pth (Val Acc: 0.7324)
[Epoch 4/20] Train Loss: 0.5406 | Val Loss: 0.5167 | Val Acc: 0.7412
✓ Saved best model to /content/drive/MyDrive/xray-diagnosis-ai/resnet50/v2/models/resnet50_model_v2_kid.pth (Val Acc: 0.7412)
[Epoch 5/20] Train Loss: 0.5006 | Val Loss: 0.5418 | Val Acc: 0.7118
[Epoch 6/20] Train Loss: 0.4744 | Val Loss: 0.4176 | Val Acc: 0.7971
✓ Saved best model to /content/drive/MyDrive/xray-diagnosis-ai/resnet50/v2/models/resnet50_model_v2_kid.pth (Val Acc: 0.7971)
[Epoch 7/20] Train Loss: 0.4373 | Val Loss: 0.3243 | Val Acc: 0.8412
✓ Saved best model to /content/drive/MyDrive/xray-diagnosis-ai/resnet50/v2/models/resnet50_model_v2_kid.pth (Val Acc: 0.8412)
[Epoch 8/20] Train Loss: 0.3983 | Val Loss: 0.3587 | Val Acc: 0.8647
✓ Saved best model to /content/drive/MyDrive/xray-diagnosis-ai/resnet50/v2/models/resnet50_model_v2_kid.pth (Val Acc: 0.8647)
[Epoch 9/20] Train Loss: 0.3349 | Val Loss: 0.3034 | Val Acc: 0.8647
[Epoch 10/20] Train Loss: 0.3176 | Val Loss: 0.2492 | Val Acc: 0.9059
✓ Saved best model to /content/drive/MyDrive/xray-diagnosis-ai/resnet50/v2/models/resnet50_model_v2_kid.pth (Val Acc: 0.9059)
[Epoch 11/20] Train Loss: 0.2859 | Val Loss: 0.2062 | Val Acc: 0.9235
✓ Saved best model to /content/drive/MyDrive/xray-diagnosis-ai/resnet50/v2/models/resnet50_model_v2_kid.pth (Val Acc: 0.9235)
[Epoch 12/20] Train Loss: 0.2564 | Val Loss: 0.1973 | Val Acc: 0.9206
[Epoch 13/20] Train Loss: 0.2434 | Val Loss: 0.2933 | Val Acc: 0.8882
[Epoch 14/20] Train Loss: 0.1951 | Val Loss: 0.3179 | Val Acc: 0.8676
[Epoch 15/20] Train Loss: 0.2019 | Val Loss: 0.1352 | Val Acc: 0.9441
✓ Saved best model to /content/drive/MyDrive/xray-diagnosis-ai/resnet50/v2/models/resnet50_model_v2_kid.pth (Val Acc: 0.9441)
[Epoch 16/20] Train Loss: 0.1375 | Val Loss: 0.1959 | Val Acc: 0.9206
[Epoch 17/20] Train Loss: 0.1496 | Val Loss: 0.0819 | Val Acc: 0.9765
✓ Saved best model to /content/drive/MyDrive/xray-diagnosis-ai/resnet50/v2/models/resnet50_model_v2_kid.pth (Val Acc: 0.9765)
[Epoch 18/20] Train Loss: 0.1324 | Val Loss: 0.1130 | Val Acc: 0.9559

[ ] import shutil

```

✓ 07:28 Python 3

## 3.2. Hiệu năng Phân loại Đa nhãn Viêm phổi

Đối với các trường hợp được xác định là VIÊM PHỔI từ bước phân loại nhị phân, các mô hình

tiếp tục thực hiện phân loại đa nhãn để xác định các dấu hiệu X-quang cụ thể.

#### Bảng 4: Hiệu năng các Mô hình trong Phân loại Đa nhãn Viêm phổi

Thực do	DenseNet121
Train Loss	0.0124
Val Loss	0.0820
F1-score	0.8408

*Fold 1:*

```
ResNet50.ipynb
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text Run all ▾
Saved final model trained on full data at /content/drive/MyDrive/densenet121/v2/models/densenet121_final_full_data.pth
!python /content/drive/MyDrive/resnet50/v3/train_densenet121_model.py
===== Fold 1/3 =====
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or 'None' for 'weights' are deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
  warnings.warn(msg)
[Fold 1][Epoch 1/20] Train Loss: 0.1252 | Val Loss: 0.1140 | Val F1: 0.4588
Thresholds: [0.38 0.37 0.27 0.3 0.25]
Saved best model fold 1 at epoch 1 with Val F1: 0.4588
[Fold 1][Epoch 2/20] Train Loss: 0.1092 | Val Loss: 0.1106 | Val F1: 0.4926
Thresholds: [0.44 0.39 0.23 0.39 0.29]
Saved best model fold 1 at epoch 2 with Val F1: 0.4926
[Fold 1][Epoch 3/20] Train Loss: 0.1053 | Val Loss: 0.1113 | Val F1: 0.5010
Thresholds: [0.3 0.36 0.33 0.33 0.48]
Saved best model fold 1 at epoch 3 with Val F1: 0.5010
[Fold 1][Epoch 4/20] Train Loss: 0.0960 | Val Loss: 0.0991 | Val F1: 0.5671
Thresholds: [0.5 0.36 0.35 0.37 0.38]
Saved best model fold 1 at epoch 4 with Val F1: 0.5671
[Fold 1][Epoch 5/20] Train Loss: 0.0881 | Val Loss: 0.1025 | Val F1: 0.5736
Thresholds: [0.39 0.35 0.42 0.37 0.47]
Saved best model fold 1 at epoch 5 with Val F1: 0.5736
[Fold 1][Epoch 6/20] Train Loss: 0.0817 | Val Loss: 0.0929 | Val F1: 0.6168
Thresholds: [0.38 0.38 0.4 0.34 0.45]
Saved best model fold 1 at epoch 6 with Val F1: 0.6168
[Fold 1][Epoch 7/20] Train Loss: 0.0703 | Val Loss: 0.0932 | Val F1: 0.6292
Thresholds: [0.52 0.45 0.4 0.33 0.45]
Saved best model fold 1 at epoch 7 with Val F1: 0.6292
```

*Fold 2:*

```

Ipython /content/drive/MyDrive/resnet50/v3/train_densenet121_model.py
Thresholds: [0.62 0.53 0.56 0.29 0.48]
[Fold 1][Epoch 20/20] Train Loss: 0.0143 | Val Loss: 0.0835 | Val F1: 0.8164
Thresholds: [0.64 0.63 0.46 0.37 0.43]

===== Fold 2/3 =====
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
/warnings.warn(
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or 'None' for 'weights' are deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
/warnings.warn(msg)
[Fold 2][Epoch 1/20] Train Loss: 0.1266 | Val Loss: 0.1079 | Val F1: 0.4821
Thresholds: [0.34 0.37 0.34 0.47 0.4]
Saved best model fold 2 at epoch 1 with Val F1: 0.4821
[Fold 2][Epoch 2/20] Train Loss: 0.1095 | Val Loss: 0.1039 | Val F1: 0.5256
Thresholds: [0.41 0.38 0.43 0.39 0.41]
Saved best model fold 2 at epoch 2 with Val F1: 0.5256
[Fold 2][Epoch 3/20] Train Loss: 0.1000 | Val Loss: 0.0969 | Val F1: 0.5471
Thresholds: [0.44 0.38 0.41 0.42 0.4]
Saved best model fold 2 at epoch 3 with Val F1: 0.5471
[Fold 2][Epoch 4/20] Train Loss: 0.0918 | Val Loss: 0.0963 | Val F1: 0.5544
Thresholds: [0.44 0.43 0.41 0.46 0.38]
Saved best model fold 2 at epoch 4 with Val F1: 0.5544
[Fold 2][Epoch 5/20] Train Loss: 0.0837 | Val Loss: 0.0921 | Val F1: 0.5971
Thresholds: [0.39 0.54 0.42 0.47 0.4]
Saved best model fold 2 at epoch 5 with Val F1: 0.5971
[Fold 2][Epoch 6/20] Train Loss: 0.0757 | Val Loss: 0.0882 | Val F1: 0.6515
Thresholds: [0.43 0.42 0.44 0.39 0.46]
Saved best model fold 2 at epoch 6 with Val F1: 0.6515
[Fold 2][Epoch 7/20] Train Loss: 0.0686 | Val Loss: 0.0849 | Val F1: 0.6656

```

Fold 3:

```

Ipython /content/drive/MyDrive/resnet50/v3/train_densenet121_model.py
Saved best model fold 2 at epoch 20 with Val F1: 0.8408

===== Fold 3/3 =====
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
/warnings.warn(
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or 'None' for 'weights' are deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
/warnings.warn(msg)
[Fold 3][Epoch 1/20] Train Loss: 0.1383 | Val Loss: 0.1163 | Val F1: 0.4862
Thresholds: [0.37 0.44 0.43 0.36 0.33]
Saved best model fold 3 at epoch 1 with Val F1: 0.4862
[Fold 3][Epoch 2/20] Train Loss: 0.1099 | Val Loss: 0.1072 | Val F1: 0.5019
Thresholds: [0.36 0.3 0.41 0.34 0.39]
Saved best model fold 3 at epoch 2 with Val F1: 0.5019
[Fold 3][Epoch 3/20] Train Loss: 0.1038 | Val Loss: 0.1002 | Val F1: 0.5359
Thresholds: [0.43 0.41 0.35 0.36 0.47]
Saved best model fold 3 at epoch 3 with Val F1: 0.5359
[Fold 3][Epoch 4/20] Train Loss: 0.0942 | Val Loss: 0.0965 | Val F1: 0.5633
Thresholds: [0.42 0.47 0.34 0.38 0.41]
Saved best model fold 3 at epoch 4 with Val F1: 0.5633
[Fold 3][Epoch 5/20] Train Loss: 0.0869 | Val Loss: 0.0985 | Val F1: 0.5756
Thresholds: [0.36 0.41 0.37 0.37 0.45]
Saved best model fold 3 at epoch 5 with Val F1: 0.5756
[Fold 3][Epoch 6/20] Train Loss: 0.0801 | Val Loss: 0.0892 | Val F1: 0.6356
Thresholds: [0.42 0.47 0.41 0.31 0.47]
Saved best model fold 3 at epoch 6 with Val F1: 0.6356
[Fold 3][Epoch 7/20] Train Loss: 0.0690 | Val Loss: 0.0857 | Val F1: 0.6549
Thresholds: [0.48 0.38 0.45 0.43 0.42]

```

## 4. Thảo luận

### 4.1. Điểm mạnh của Hệ thống Phát triển

Hệ thống được phát triển có một số điểm mạnh đáng chú ý:

- Cách tiếp cận phân loại đa giai đoạn mới mẻ:** Việc kết hợp phân loại nhị phân ban đầu với phân loại đa nhãn chi tiết hơn cho các trường hợp viêm phổi là một hướng đi có tính ứng dụng cao trong lâm sàng nhi khoa.

- **Tính năng tương tác Bác sĩ-AI và Cảnh báo:** Việc cho phép bác sĩ nhập chẩn đoán và cảnh báo khi có sự khác biệt lớn với AI là một điểm mạnh quan trọng, thúc đẩy sự hợp tác giữa người và máy, đồng thời có thể tăng cường độ an toàn và chất lượng chẩn đoán.
- **Kiến trúc triển khai thực tiễn và hiệu quả:** Việc sử dụng backend NodeJS kết hợp với ONNX Runtime cho thấy một giải pháp hiện đại, có khả năng mở rộng và dễ dàng tích hợp vào các quy trình làm việc lâm sàng. Điều này đặc biệt quan trọng để đưa AI từ phòng thí nghiệm đến ứng dụng thực tế.
- **Sử dụng bộ dữ liệu công khai và tinh chỉnh:** Việc dựa trên các bộ dữ liệu được công bố rộng rãi (Kaggle, PhysioNet) và việc tinh chỉnh tập dữ liệu/nhãn cho nhiệm vụ cụ thể (1000 ảnh, 5 nhãn cho DenseNet đa nhãn) giúp tăng tính minh bạch, khả năng tái lập và sự tập trung của nghiên cứu.
- **Động lực cộng đồng và phi lợi nhuận:** Mục tiêu phục vụ cộng đồng và không vì lợi nhuận thương mại là một điểm mạnh về mặt đạo đức và định hướng phát triển của dự án.
- **Mã nguồn mở:** Việc công khai mã nguồn trên GitHub thúc đẩy tính minh bạch và hợp tác.

#### 4.2. Hạn chế của Nghiên cứu

Mặc dù có những điểm mạnh, nghiên cứu này cũng có một số hạn chế cần được thừa nhận:

- **Hạn chế về Bộ dữ liệu:**
  - *Quy mô và Tính đa dạng:* Mặc dù các bộ dữ liệu công khai được sử dụng là tiêu chuẩn, chúng có thể không hoàn toàn đại diện cho quần thể bệnh nhân cụ thể hoặc thiết bị hình ảnh tại Bệnh viện Nhi đồng 2. Việc sử dụng tập con 1000 ảnh cho DenseNet đa nhãn, mặc dù tập trung, cũng giới hạn quy mô dữ liệu cho nhiệm vụ đó.
  - *Chất lượng Nhãn và Mất cân bằng:* Các bộ dữ liệu công khai lớn có thể chứa nhiều trong nhãn. Sự mất cân bằng lớp đối với một số dấu hiệu viêm phổi nhất định có thể ảnh hưởng đến hiệu năng của mô hình. Việc lọc nhãn cho DenseNet nhằm giảm thiểu điều này nhưng vẫn cần lưu ý.
- **Hạn chế về Mô hình:**
  - *Bản chất "Hộp đen":* Các mô hình học sâu thường được coi là "hộp đen". Việc thiếu khả năng giải thích một cách tường minh có thể là một rào cản đối với sự tin cậy trong lâm sàng. Ngay cả khi mô hình chính xác, các bác sĩ lâm sàng có thể do dự khi tin tưởng chúng nếu không hiểu *tại sao* một chẩn đoán cụ thể được đưa ra.
  - *Vấn đề với Tiền huấn luyện trên ImageNet:* Cần thảo luận về những tác động tiêu cực tiềm ẩn của sự khác biệt miền giữa ImageNet (ảnh tự nhiên) và CXR

nhi khoa, chẳng hạn như quá khớp với các đặc trưng không liên quan đến lâm sàng hoặc tổng quát hóa dưới mức tối ưu, bất chấp việc tinh chỉnh.

- **Hạn chế về Hệ thống và Môi trường Huấn luyện:**
  - Hệ thống hiện tại chưa tích hợp các đầu ra AI giải thích được (trừ khi người dùng đã triển khai nhưng không rõ).
  - Hiệu năng của backend NodeJS dưới tải rất cao chưa được kiểm tra một cách rõ ràng trong yêu cầu của người dùng.
  - Việc huấn luyện trên Google Colab phiên bản miễn phí có thể giới hạn tài nguyên tính toán và thời gian huấn luyện, ảnh hưởng đến khả năng khám phá không gian siêu tham số rộng hơn hoặc huấn luyện trên các tập dữ liệu lớn hơn mà không có chiến lược chia nhỏ.
  - Cơ chế xác định "độ chênh lệch" cho tính năng cảnh báo cần được tinh chỉnh và đánh giá kỹ lưỡng để đảm bảo tính hữu ích và tránh gây ra quá nhiều cảnh báo không cần thiết.

Một mối quan tâm quan trọng là khả năng tổng quát hóa của mô hình. Các mô hình được huấn luyện trên các bộ dữ liệu cụ thể. Hiệu năng của chúng trên dữ liệu chưa từng thấy từ Bệnh viện Nhi đồng 2, vốn có thể có các đặc điểm khác biệt (nhân khẩu học bệnh nhân, quy trình chụp ảnh, tỷ lệ mắc bệnh), là không chắc chắn. Đây là một điểm quan trọng đối với việc triển khai trong thế giới thực.

#### **4.3. Ý nghĩa Lâm sàng và Tiện ích cho Bệnh viện Nhi đồng 2**

Hệ thống AI được phát triển có tiềm năng mang lại nhiều lợi ích lâm sàng cho Bệnh viện Nhi đồng 2:

- **Công cụ hỗ trợ chẩn đoán:** Có thể hoạt động như một công cụ hỗ trợ cho các bác sĩ X-quang và bác sĩ nhi khoa, đặc biệt trong các tình huống có khối lượng công việc cao hoặc đối với các bác sĩ ít kinh nghiệm hơn.
- **Tăng cường sự tự tin và an toàn trong chẩn đoán:** Tính năng cảnh báo khi có sự khác biệt giữa chẩn đoán của bác sĩ và AI có thể giúp giảm thiểu sai sót, khuyến khích việc xem xét lại và nâng cao chất lượng chẩn đoán tổng thể.
- **Ưu tiên ca bệnh:** Có thể giúp ưu tiên các trường hợp khẩn cấp cần được xem xét nhanh chóng.
- **Hỗ trợ lập kế hoạch điều trị:** Đầu ra đa nhãn (đặc biệt từ DenseNet với các nhãn đã chọn) có thể hỗ trợ lập kế hoạch điều trị bằng cách cung cấp thông tin chi tiết hơn về các đặc điểm của viêm phổi.
- **Khả năng tích hợp:** Kiến trúc NodeJS/ONNX cho phép tích hợp tương đối dễ dàng với các hệ thống thông tin bệnh viện (HIS) hoặc hệ thống lưu trữ và truyền hình ảnh (PACS) hiện có, với điều kiện tương thích API.
- **Đóng góp cho cộng đồng y khoa:** Là một dự án cộng đồng, hệ thống này có thể

trở thành một nguồn tài nguyên và công cụ học tập quý giá.

## 5. Kết luận và Hướng phát triển Tương lai

### 5.1. *Tóm tắt Đóng góp*

Nghiên cứu này đã trình bày việc phát triển một hệ thống AI toàn diện để hỗ trợ chẩn đoán viêm phổi ở trẻ em, với một cam kết mạnh mẽ phục vụ cộng đồng và không vì lợi ích thương mại. Hệ thống này sử dụng các mô hình học sâu ResNet50 và DenseNet với kỹ thuật học chuyển tiếp, thực hiện quy trình chẩn đoán đa giai đoạn bao gồm phân loại nhị phân (viêm phổi/bình thường, chủ yếu với ResNet50 trên toàn bộ dữ liệu Kaggle) và phân loại đa nhãn các bệnh lý cụ thể (chủ yếu với DenseNet trên tập con 1000 ảnh từ VinDr-PCXR với 5 nhãn chọn lọc).

Một kiến trúc triển khai dựa trên NodeJS, Cloudinary và ONNX Runtime đã được xây dựng, cho thấy một giải pháp khả thi và hiệu quả để đưa các mô hình AI vào ứng dụng thực tế. Đặc biệt, tính năng cho phép bác sĩ nhập chẩn đoán và cảnh báo khi có sự chênh lệch với AI đã được tích hợp, nhằm tăng cường sự an toàn và hỗ trợ quyết định. Việc huấn luyện được thực hiện trên Google Colaboratory và mã nguồn được công khai. Các kết quả đánh giá hiệu năng ban đầu (cần được điều chỉnh) đã chứng minh tiềm năng của hệ thống.

### 5.2. *Hướng phát triển Tương lai và Khuyến nghị*

Để hệ thống này thực sự trở thành một công cụ lâm sàng có giá trị và được chấp nhận rộng rãi, đặc biệt tại Bệnh viện Nhi đồng 2, một số hướng phát triển và cải tiến cần được xem xét:

- **Tích hợp AI Giải thích được (XAI):** Kết hợp các kỹ thuật XAI như Grad-CAM hoặc LIME để cung cấp các giải thích trực quan và bằng văn bản cho các dự đoán của mô hình, từ đó tăng cường tính minh bạch và sự tin cậy của bác sĩ lâm sàng.
- **Mở rộng và Địa phương hóa Bộ dữ liệu:** Thu thập và tích hợp dữ liệu CXR nhi khoa Ảnh danh tại Bệnh viện Nhi đồng 2 để tinh chỉnh và xác thực mô hình. Điều này sẽ giúp cải thiện khả năng thích ứng và tổng quát hóa của mô hình với môi trường bệnh viện cụ thể.
- **Xác thực Lâm sàng Tiền cứu:** Tiến hành các thử nghiệm lâm sàng tiền cứu nghiêm ngặt tại Bệnh viện Nhi đồng 2 để đánh giá hiệu năng thực tế của hệ thống, khả năng sử dụng và tác động đến việc ra quyết định lâm sàng cũng như kết quả của bệnh nhân. Điều này cũng bao gồm việc đánh giá tính hữu ích và hiệu quả của tính năng cảnh báo tương tác bác sĩ-AI.
- **Tối ưu hóa Mô hình hơn nữa:** Khám phá các kiến trúc mới hơn, các kỹ thuật học chuyển tiếp tiên tiến hơn (ví dụ: tiền huấn luyện theo miền cụ thể nếu có đủ dữ liệu cục bộ), hoặc các phương pháp học tập hợp (ensemble methods).
- **Tinh chỉnh Nhiệm vụ đa nhãn:** Mở rộng bộ nhãn cho phân loại đa nhãn dựa trên nhu cầu lâm sàng tại bệnh viện, có thể bao gồm cả việc chấm điểm mức độ

nghiêm trọng của bệnh, sau khi có thêm dữ liệu.

- **Giám sát và Cập nhật Liên tục:** Triển khai các chiến lược để theo dõi hiệu năng của mô hình sau khi triển khai và huấn luyện lại/cập nhật mô hình khi có dữ liệu mới hoặc hiệu năng suy giảm.
- **Phát triển Giao diện Người dùng (UI):** Tiếp tục hoàn thiện giao diện người dùng (đã có repo frontend) để các bác sĩ lâm sàng có thể dễ dàng tải ảnh lên, nhập chẩn đoán, xem kết quả và nhận cảnh báo, bao gồm cả các đầu ra XAI.
- **Tích hợp với Hệ thống Bệnh viện:** Lập kế hoạch tích hợp sâu hơn với HIS/PACS.

Con đường để tích hợp AI vào thực hành lâm sàng là một quá trình lặp đi lặp lại. Dự án hiện tại là một bằng chứng khái niệm mạnh mẽ. Tuy nhiên, việc biến nó thành một công cụ được áp dụng lâm sàng đòi hỏi sự nghiên cứu, phát triển và xác nhận liên tục, đặc biệt tập trung vào việc thích ứng cục bộ, XAI và các thử nghiệm lâm sàng nghiêm ngặt. Sự hợp tác giữa con người và AI, cùng với việc xây dựng lòng tin, là chìa khóa cho tương lai của AI trong X-quang và hình ảnh y tế.

## 6. Lời cảm ơn

Dự án này được thực hiện với mục tiêu đóng góp cho cộng đồng, đặc biệt là nâng cao chất lượng chăm sóc sức khỏe cho các bệnh nhi tại Bệnh viện Nhi đồng 2. Xin chân thành cảm ơn sự truyền cảm hứng từ các y bác sĩ và các em nhỏ và sự hướng dẫn từ Bác sĩ Trần Nam Hưng, anh Trần Quốc Tú, anh Trần Quốc Tuấn, anh Trần Văn Hưng.

Dự án này hoàn toàn không vì mục đích lợi nhuận thương mại và mong muốn được chia sẻ rộng rãi để mang lại lợi ích cho cộng đồng, xã hội đặc biệt là các em nhỏ.

## 7. Tài liệu Tham khảo

1. AI-Driven Thoracic X-ray Diagnostics: Transformative Transfer Learning for Multi-Disease Classification and Segmentation. (2025, June 6). PMC. Retrieved from <https://PMC.ncbi.nlm.nih.gov/articles/PMC11355475/>
2. PyTorch. (n.d.). ResNet. PyTorch Hub. Retrieved June 6, 2025, from [https://pytorch.org/hub/pytorch\\_vision\\_resnet/](https://pytorch.org/hub/pytorch_vision_resnet/)
3. PyTorch. (n.d.). DenseNet. PyTorch Hub from [https://pytorch.org/hub/pytorch\\_vision\\_densenet/](https://pytorch.org/hub/pytorch_vision_densenet/)
4. DigitalOcean. (n.d.). Deep Learning Architecture Review: DenseNet, ResNeXt, MnasNet & ShuffleNet v2. Retrieved June 6, 2025, from <https://www.digitalocean.com/community/tutorials/popular-deep-learning-architectures-densenet-mnasnet-shufflenet>
5. Amplework Software. (n.d.). Integrating Node.js with AI Pipelines: Efficient Streaming & Queuing. Retrieved June 6, 2025, from

<https://www.amplework.com/blog/nodejs-ai-pipeline-integration/>

6. AI, R. (n.d.). *How to Implement OpenAI API Key in a Node.js Express App*. DEV Community. Retrieved June 6, 2025, from <https://dev.to/rowsanali/how-to-implement-openai-api-key-in-a-nodejs-express-app-3f5k>
7. Cloudinary. (n.d.). *Upload Images in Node.js (Video Tutorial)*. Retrieved June 6, 2025, from [https://cloudinary.com/documentation/upload\\_assets\\_in\\_node\\_tutorial](https://cloudinary.com/documentation/upload_assets_in_node_tutorial)
8. Microsoft. (n.d.). *onnxruntime-inference-examples/js/api-usage\_session-options/README.md at main*. GitHub. Retrieved June 6, 2025, from [https://github.com/microsoft/onnxruntime-inference-examples/blob/main/js/api-usage\\_session-options/README.md](https://github.com/microsoft/onnxruntime-inference-examples/blob/main/js/api-usage_session-options/README.md)
9. Microsoft Learn. (n.d.). *ONNX Runtime and models - Azure Machine Learning*. Retrieved June 6, 2025, from <https://learn.microsoft.com/en-us/azure/machine-learning/concept-onnx?view=azur%0Aeml-api-2>