FPT SCHOOL OF BUSINESS & TECHNOLOGY

BÁO CÁO PHÂN TÍCH BÀI BÁO KHOA HỌC

A Comparative Study of CNN, ResNet, and Vision Transformers for Multi-Classification of Chest Diseases

arXiv: 2406.00237

Ananya Jain, Aviral Bhardwaj, Kaushik Murali, Isha Surani
University of Toronto

Kho mã nguồn gốc:	https://github.com/Aviral-03/ViT-Chest-Xray
Bộ dữ liệu:	NIH ChestX-ray14 (112 120 ảnh X-quang)
Framework gốc:	TensorFlow/Keras
Framework hiện tại:	PyTorch
Link Github của team:	https://github.com/AnBinh1703/ViT-Chest-Xray

Sinh viên thực hiện:

Dương Bình An	MSSV: 25MSA23234
Lê Quang Tuyến	MSSV: 25MSA23232
Nguyễn Lê Hồng Nhi	MSSV: 25MSA23235

Lớp:	MSA29HCM
Môn học:	Deep Learning
Giảng viên hướng dẫn:	TS. Lê Việt Tuấn

Thành phố Hồ Chí Minh – 2026

Tóm tắt nghiên cứu

Tổng quan bài báo

Bài báo "A Comparative Study of CNN, ResNet, and Vision Transformers for Multi-Classification of Chest Diseases" (arXiv:2406.00237) nghiên cứu và so sánh hiệu năng của ba nhóm kiến trúc học sâu phổ biến trong bài toán phân loại đa nhãn bệnh lý trên ảnh X-quang ngực, bao gồm:

1. Convolutional Neural Networks (CNN): Mô hình baseline với kiến trúc tích chập truyền thống.
2. Residual Networks (ResNet): Mạng CNN sâu sử dụng skip-connection nhằm khắc phục hiện tượng mất gradient.
3. Vision Transformers (ViT): Kiến trúc Transformer áp dụng cho thị giác máy tính thông qua cơ chế self-attention.

Nghiên cứu được thực hiện trên bộ dữ liệu NIH ChestX-ray14 với hơn 112.000 ảnh X-quang ngực và 14 nhãn bệnh lý khác nhau.

Đóng góp chính của bài báo

Các đóng góp nổi bật của bài báo bao gồm:

- So sánh có hệ thống hiệu năng giữa CNN, ResNet và Vision Transformer trên cùng tập dữ liệu và thiết lập thí nghiệm.
- Đề xuất hai biến thể Vision Transformer: ViT-v1 (huấn luyện từ đầu) và ViT-v2 (bổ sung regularization).
- Giới thiệu mô hình lai ViT–ResNet với pre-training trên ImageNet-21k nhằm kết hợp ưu điểm của CNN và Transformer.
- Phân tích attention maps để trực quan hoá và diễn giải quyết định của mô hình ViT trong ngữ cảnh y tế.

Kết quả thực nghiệm chính

Mô hình	Test Acc (%)	Val Acc (%)	AUC	Ghi chú
CNN	91.00	92.68	0.82	Baseline
ResNet	93.00	93.34	0.86	Strong baseline
ViT-v1	92.63	92.89	0.86	Huấn luyện từ đầu
ViT-v2	92.83	92.95	0.84	Có regularization
ViT–ResNet	93.90	94.07	0.85	Mô hình lai, pre-trained

Kết quả cho thấy ResNet và ViT–ResNet đạt hiệu năng cao và ổn định hơn so với CNN baseline và các biến thể ViT huấn luyện từ đầu.

Phạm vi và mục tiêu của báo cáo

Báo cáo phân tích chuyên sâu này không chỉ dừng lại ở việc tóm tắt bài báo gốc, mà còn mở rộng và đào sâu các khía cạnh sau:

1. Phân tích lý thuyết: Trình bày chi tiết nền tảng toán học và trực giác của CNN, ResNet và Vision Transformer.
2. Liên kết paper với mã nguồn: Mapping rõ ràng giữa các mô tả trong bài báo và phần cài đặt trong kho mã nguồn GitHub.
3. Tái lập và cải tiến: Chuyển đổi pipeline từ TensorFlow/Keras sang PyTorch và đề xuất các cải tiến hiện đại.
4. Đánh giá phê bình: Phân tích ưu, nhược điểm của từng kiến trúc và khả năng ứng dụng trong thực tế lâm sàng.

Từ khoá

Keywords: Chest X-ray • Multi-label Classification • Convolutional Neural Networks • Residual Networks • Vision Transformer • Self-Attention • Transfer Learning • Medical Image Analysis • ROC-AUC • PyTorch • Deep Learning

MỤC LỤC

DANH SÁCH HÌNH

DANH SÁCH BẢNG

Chương 1 Giới thiệu và Bối cảnh nghiên cứu

1.1 Bối cảnh lâm sàng

Bệnh lý phổi và tim–phổi (như viêm phổi, tràn dịch màng phổi, khí phế thũng, xẹp phổi, tim to, v.v.) là một trong những nguyên nhân hàng đầu gây tử vong trên toàn cầu. Theo Tổ chức Y tế Thế giới (WHO), các bệnh liên quan đến hệ hô hấp chiếm hơn 10% tổng số ca tử vong mỗi năm, đặc biệt phổ biến tại các quốc gia đang phát triển.

Trong thực hành lâm sàng, X-quang ngực (Chest X-ray) là phương tiện chẩn đoán hình ảnh được sử dụng rộng rãi nhất do các ưu điểm sau:

- Chi phí thấp: So với CT hoặc MRI, X-quang có chi phí triển khai thấp hơn đáng kể.
- Thời gian xử lý nhanh: Kết quả có thể thu được trong vòng vài phút.
- Khả năng sàng lọc ban đầu: Thường là chỉ định đầu tiên khi nghi ngờ bệnh lý phổi.
- Phát hiện đa dạng bệnh lý: Có thể phát hiện nhiều dạng tổn thương khác nhau trong cùng một ảnh.

1.2 Thách thức trong chẩn đoán X-quang ngực

Mặc dù có nhiều ưu điểm, việc diễn giải ảnh X-quang ngực trong thực tế lâm sàng vẫn tồn tại nhiều thách thức nghiêm trọng. Như bài báo gốc đã chỉ ra:

> **Trích dẫn từ bài báo gốc (Section 1)**
>
> "Although chest X-ray imaging is a relatively low cost tool for diagnosis, radiologists are needed to analyze these images. However the limited access to radiologists in many areas, and the variability between radiologists can be a problem."

Các thách thức chính bao gồm:

1. Thiếu hụt bác sĩ chẩn đoán hình ảnh: Đặc biệt tại vùng sâu, vùng xa hoặc các quốc gia đang phát triển.
2. Biến thiên giữa các chuyên gia (inter-observer variability): Cùng một ảnh có thể dẫn đến các kết luận khác nhau.
3. Khối lượng dữ liệu lớn: Một bác sĩ có thể phải đọc hàng trăm ảnh mỗi ngày, làm tăng nguy cơ sai sót.
4. Dấu hiệu bệnh tinh vi: Nhiều tổn thương giai đoạn sớm rất khó phát hiện bằng mắt thường.

1.3 Động lực ứng dụng trí tuệ nhân tạo trong chẩn đoán hình ảnh

Sự phát triển mạnh mẽ của học sâu (Deep Learning) đã tạo ra bước ngoặt trong lĩnh vực phân tích hình ảnh y tế. Các mô hình học sâu có khả năng tự động học đặc trưng từ dữ liệu ảnh lớn, giảm phụ thuộc vào kinh nghiệm chủ quan của con người.

Những lợi ích chính của việc áp dụng học máy trong chẩn đoán hình ảnh y tế bao gồm:

- Tăng độ chính xác và tính nhất quán trong chẩn đoán.
- Hỗ trợ bác sĩ trong quá trình ra quyết định lâm sàng.
- Mở rộng khả năng tiếp cận dịch vụ y tế tại các khu vực thiếu nhân lực chuyên môn.

Về mặt lịch sử, các mốc phát triển quan trọng của học sâu trong phân tích ảnh y tế có thể tóm tắt như sau:

1. 2012 – AlexNet: Khởi đầu kỷ nguyên học sâu với CNN.
2. 2015 – ResNet: Giải quyết vấn đề mất gradient, cho phép huấn luyện mạng rất sâu.
3. 2017 – Transformer: Giới thiệu cơ chế self-attention trong NLP.
4. 2020 – Vision Transformer (ViT): Áp dụng Transformer cho thị giác máy tính.

1.4 Giới thiệu bài báo gốc

Bài báo "A Comparative Study of CNN, ResNet, and Vision Transformers for Multi-Classification of Chest Diseases" (arXiv:2406.00237) tập trung so sánh hiệu năng của ba nhóm kiến trúc học sâu phổ biến trong bài toán phân loại đa nhãn bệnh lý trên ảnh X-quang ngực, sử dụng bộ dữ liệu NIH ChestX-ray14.

Các mô hình được nghiên cứu bao gồm:

- CNN baseline truyền thống.
- ResNet-34 với skip connections.
- Hai biến thể Vision Transformer (ViT-v1, ViT-v2).
- Mô hình lai ViT–ResNet sử dụng pre-training trên ImageNet-21k.

1.5 Câu hỏi nghiên cứu

Bài báo gốc đặt ra các câu hỏi nghiên cứu cốt lõi sau:

RQ1 Hiệu năng của CNN, ResNet và ViT khác nhau như thế nào trong phân loại bệnh X-quang ngực?

RQ2 Vision Transformer huấn luyện từ đầu có thể cạnh tranh với ResNet hay không?

RQ3 Transfer learning ảnh hưởng như thế nào đến hiệu năng của ViT?

RQ4 Attention maps có giúp cải thiện khả năng diễn giải mô hình hay không?

1.6 Mục tiêu và phạm vi báo cáo

Báo cáo này hướng tới các mục tiêu sau:

- Phân tích sâu nền tảng lý thuyết của CNN, ResNet và Vision Transformer.
- Liên kết nội dung bài báo với mã nguồn triển khai thực tế.
- Đánh giá vai trò của inductive bias và transfer learning trong ảnh y tế.

- Đề xuất các hướng cải tiến dựa trên các phương pháp hiện đại.

Phạm vi nghiên cứu bao gồm:

- Bộ dữ liệu: NIH ChestX-ray14.
- Bài toán: Phân loại đa nhãn (multi-label classification).
- Mô hình: CNN, ResNet-34, ViT và ViT–ResNet.

1.7 Cấu trúc báo cáo

Phần còn lại của báo cáo được tổ chức như sau:

- Chương 2: Các công trình liên quan (Related Works).
- Chương 3: Phân tích bộ dữ liệu NIH ChestX-ray14.
- Chương 4: Mô hình CNN – lý thuyết và triển khai.
- Chương 5: Mô hình ResNet.
- Chương 6: Vision Transformer và mô hình lai.
- Chương 7: Thực nghiệm và phân tích kết quả.
- Chương 8: Các cải tiến và thảo luận mở rộng.
- Chương 9: Kết luận và hướng nghiên cứu tương lai.

Chương 2 Các Công Trình Liên Quan (Related Works)

2.1 Tổng quan

Lĩnh vực phân tích ảnh X-quang ngực bằng học sâu đã chứng kiến sự phát triển mạnh mẽ trong thập kỷ qua. Chương này tổng hợp các công trình tiêu biểu theo bốn hướng nghiên cứu chính: (1) Phương pháp dựa trên CNN, (2) Kiến trúc Transformer cho thị giác y tế, (3) Xử lý mất cân bằng lớp, và (4) Khả năng diễn giải mô hình.

2.2 Phương pháp dựa trên CNN (CNN-based Methods)

2.2.1 ChestX-ray14 và CheXNet

Wang et al. (2017) [1] giới thiệu bộ dữ liệu NIH ChestX-ray14 với 112,120 ảnh và đề xuất baseline sử dụng DenseNet-121. Đây là công trình nền tảng cho hầu hết các nghiên cứu tiếp theo.

Rajpurkar et al. (2017) [2] phát triển CheXNet dựa trên DenseNet-121 với 121 lớp, đạt AUC 0.841 trên task phát hiện Pneumonia, vượt qua mức độ chính xác của bác sĩ X-quang. Các đóng góp chính:

- Sử dụng ImageNet pretraining kết hợp fine-tuning.
- Class activation maps (CAM) để trực quan hóa vùng quan tâm.
- Đánh giá nghiêm ngặt với radiologist-level comparison.

Key Insight: DenseNet vs ResNet

DenseNet kết nối mọi lớp với mọi lớp sau đó (dense connections), giúp:

- Gradient flow tốt hơn ResNet.
- Feature reuse hiệu quả, giảm số tham số.
- Phù hợp với medical imaging (cần preserve fine details).

2.2.2 CheXpert và Uncertainty Labeling

Irvin et al. (2019) [3] công bố bộ dữ liệu CheXpert với 224,316 ảnh từ 65,240 bệnh nhân. Khác biệt quan trọng:

- Xử lý uncertain labels (U-Ones, U-Zeros, U-Ignore strategies).
- Multi-task learning với 14 pathologies.
- Policy optimization cho uncertain labels: U-Ones cho Atelectasis, U-Zeros cho Pleural Effusion.

So sánh CheXpert vs NIH ChestX-ray14:

Bảng 2.1: So sánh hai bộ dữ liệu lớn nhất

Đặc điểm	NIH ChestX-ray14	CheXpert
Số ảnh	112,120	224,316
Số bệnh nhân	30,805	65,240
Số bệnh lý	14 (+ No Finding)	14
Label extraction	NLP (weak)	NLP + uncertainty handling
View position	Frontal only	Frontal + Lateral
Public	✓	✓

2.2.3 MIMIC-CXR và Vision-Language Models

Johnson et al. (2019) [4] phát hành MIMIC-CXR với 377,110 ảnh kèm theo 227,835 radiology reports. Đây là bước đầu tiên hướng đến vision-language models trong y tế.

2.3 Kiến trúc Transformer cho Thị giác Y tế

2.3.1 Vision Transformer (ViT)

Dosovitskiy et al. (2020) [5] giới thiệu Vision Transformer, áp dụng kiến trúc Transformer thuần túy cho ảnh:

- Chia ảnh thành patches 16×16 hoặc 32×32.
- Self-attention toàn cục thay vì convolution.
- Cần pretraining quy mô lớn (ImageNet-21k, JFT-300M).

Kết quả quan trọng:

- ViT-Huge pretrained trên JFT-300M đạt SOTA ImageNet.
- ViT-Base trained from scratch thua ResNet-50 khi dữ liệu nhỏ.
- Cần ∼100M ảnh để ViT vượt CNN inductive bias.

2.3.2 Swin Transformer

Liu et al. (2021) [6] đề xuất Swin Transformer với shifted windows:

- Hierarchical feature maps (giống CNN).
- Complexity $O(HW)$ thay vì $O((HW)^2)$ của ViT.
- Phù hợp medical imaging: cần multi-scale features.

Ứng dụng trong y tế: Swin-Tiny đạt AUC 0.903 trên NIH ChestX-ray14 (nêu trong Chapter 8), vượt ViT-Base (0.836) với ít tham số hơn (28M vs 86M).

2.3.3 Hybrid CNN-Transformer Architectures

Xu et al. (2021) [7] đề xuất ViTAE (ViT with Absolute position Embedding), kết hợp:

- CNN stem để extract low-level features.
- Transformer encoder cho global reasoning.
- Hiệu quả hơn pure ViT với dữ liệu y tế ($<$ 1M ảnh).

2.4 Xử lý Mất Cân Bằng Lớp (Class Imbalance Handling)

2.4.1 Focal Loss

Lin et al. (2017) [8] đề xuất Focal Loss cho object detection với extreme imbalance:

$$\text{FL}(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t) \tag{2.1}$$

Ứng dụng medical imaging:

- Hernia (0.20%) vs No Finding (53.84%): tỷ lệ 1:269.
- Focal Loss ($\gamma = 2$) tăng AUC từ 0.722 $\rightarrow$ 0.758 (Chapter 8).
- Down-weight easy negatives (No Finding), focus on hard positives (rare diseases).

2.4.2 Asymmetric Loss (ASL)

Ridnik et al. (2021) [9] phát triển ASL cho multi-label classification:

- Asymmetric focusing: $\gamma_- = 4$ (negative), $\gamma_+ = 1$ (positive).
- Hard negative mining với probability clipping.
- Đạt SOTA trên MS-COCO, NUS-WIDE.

Kết quả trên NIH ChestX-ray14: ASL đạt macro-AUC 0.763, cao nhất trong các loss functions (Chapter 8).

2.4.3 Class-Balanced Sampling và Re-weighting

Cui et al. (2019) [10] đề xuất class-balanced loss:

$$\text{CB-Loss} = \frac{1-\beta}{1-\beta^{n_y}} \cdot \text{Loss}(y) \tag{2.2}$$

với n_y là số mẫu của lớp y, $\beta \in [0, 1)$.

2.5 Khả năng Diễn Giải Mô Hình (Model Explainability)

2.5.1 Grad-CAM và Variants

Selvaraju et al. (2017) [11] đề xuất Grad-CAM:

- Sử dụng gradient của output class đối với feature maps cuối cùng.
- Tạo heatmap chỉ vùng quan trọng cho decision.
- Áp dụng rộng rãi trong y tế: CheXNet, DenseNet.

Hạn chế:

- Chỉ highlight vùng có gradient cao, không phải causality.
- Có thể focus vào artifacts (markers, chest tubes) thay vì pathology.

2.5.2 Attention Maps trong Transformer

ViT attention maps cung cấp trực quan hóa tự nhiên:

- Mỗi patch attend to các patches khác.
- CLS token attention cho biết vùng nào quan trọng cho classification.
- Transparent hơn Grad-CAM (built-in mechanism, không cần post-hoc).

Cảnh báo: Attention is not explanation (Jain & Wallace, 2019). Attention maps chỉ show mô hình nhìn vào đâu, không chứng minh causality.

2.5.3 Uncertainty Quantification

Gal & Ghahramani (2016) [12] đề xuất MC Dropout:

- Dropout at test time để ước lượng epistemic uncertainty.
- Quan trọng trong y tế: model nên "biết khi nào không biết".
- Ứng dụng: Flag high-uncertainty cases cho bác sĩ review.

2.6 Nghiên Cứu So Sánh Kiến Trúc

2.6.1 CNN vs Transformer trong Y Tế

Matsoukas et al. (2021) [13] so sánh CNN và ViT trên 8 bộ dữ liệu y tế:

- Kết luận: ResNet-50 pretrained outperforms ViT-Base từ scratch.
- ViT cần >10× dữ liệu hơn để match ResNet.
- Transfer learning là yếu tố quyết định, không phải architecture.

2.6.2 Bài Báo Gốc (Paper Under Analysis)

Jain et al. (2024) [14] – bài báo được phân tích trong luận văn này:

- So sánh CNN, ResNet-34, ViT-v1, ViT-v2, ViT-ResNet.
- Dataset: NIH ChestX-ray14 (85,000 ảnh subset).
- Kết quả: ViT-ResNet pretrained đạt 93.9% accuracy (best).
- Hạn chế: Không rõ patient-level split, thiếu per-class AUC.

2.7 Gaps và Đóng Góp của Luận Văn Này

2.7.1 Gaps trong Literature

1. Thiếu replication studies nghiêm ngặt: Nhiều paper báo cáo kết quả cao nhưng không reproducible do data leakage hoặc evaluation bias.
2. Thiếu phân tích architectural insights: Ít công trình giải thích tại sao một kiến trúc tốt hơn, chỉ báo cáo con số.
3. Thiếu đánh giá patient-level generalization: Phần lớn split theo image, dẫn đến overfitting lên patient-specific characteristics.
4. Thiếu so sánh fair giữa CNN và Transformer: Thường compare pretrained ViT với ResNet from scratch.

2.7.2 Đóng Góp của Luận Văn

Luận văn này khắc phục các gaps trên bằng cách:

1. Replication study có kiểm soát: Tái hiện bài báo gốc, chỉ ra sự khác biệt giữa kết quả báo cáo và kết quả thực tế.
2. Phân tích inductive bias: Giải thích vai trò của locality, translation equivariance trong hiệu năng.
3. Patient-level evaluation: Chia dữ liệu theo Patient ID, loại bỏ data leakage.
4. Per-class analysis đầy đủ: Đánh giá AUC cho cả 15 classes, không chỉ macro-average.
5. Cải tiến hệ thống: Transfer learning, Focal/ASL loss, Swin Transformer, Ensemble.

Chương 3 Phân tích Dataset: NIH Chest X-ray

3.1 Tổng quan Dataset

[Paper: Section 4.1 - Dataset]

3.1.1 Thông tin cơ bản

NIH Chest X-ray Dataset

- Tên đầy đủ: ChestX-ray14 (NIH Clinical Center)
- Số lượng ảnh: 112,120 frontal-view X-ray images
- Số bệnh nhân: 30,805 unique patients
- Số nhãn bệnh: 14 bệnh + 1 "No Finding" = 15 classes
- Nguồn nhãn: Text-mining từ radiology reports (weak labels)
- Kích thước ảnh gốc: 1024 × 1024 pixels

3.1.2 Trích dẫn từ Paper

Paper Quote - Dataset

"To evaluate the performance of our model architectures, we utilized two freely available datasets: the NIH Chest X-ray dataset comprising 112,120 X-ray images with disease labels from 30,805 unique patients, and a Random Sample of the NIH Chest X-ray Dataset, containing 5,606 X-ray images. Both datasets involved multi-class classification across 15 classes, each representing different disease labels."

3.2 Danh sách 15 Classes

3.2.1 Phân loại bệnh lý

Bảng 3.1: 15 Classes trong NIH Chest X-ray Dataset

ID	Tên bệnh	Tỷ lệ (%)	Mô tả
0	Cardiomegaly	2.48	Tim to
1	Emphysema	2.24	Khí phế thũng
2	Effusion	11.88	Tràn dịch màng phổi
3	Hernia	0.20	Thoát vị
4	Nodule	5.65	Nốt phổi
5	Pneumothorax	4.73	Tràn khí màng phổi
6	Atelectasis	10.31	Xẹp phổi
7	Pleural_Thickening	3.02	Dày màng phổi
8	Mass	5.16	Khối u
9	Edema	2.05	Phù phổi
10	Consolidation	4.16	Đông đặc phổi
11	Infiltration	17.74	Thâm nhiễm
12	Fibrosis	1.50	Xơ phổi
13	Pneumonia	1.28	Viêm phổi
14	No Finding	53.84	Không phát hiện bệnh

3.2.2 Phân tích mất cân bằng lớp (Class Imbalance)

Vấn đề Class Imbalance

Quan sát quan trọng:

- "No Finding" chiếm 53.84% - hơn một nửa dataset
- "Hernia" chỉ chiếm 0.20% - rất hiếm
- Tỷ lệ cao nhất / thấp nhất = 53.84 / 0.20 = 269 lần

Hậu quả:

- Model có thể thiên về dự đoán "No Finding"
- Accuracy cao nhưng chưa chắc đã detect tốt bệnh hiếm
- Cần metrics như AUC thay vì chỉ accuracy

3.3 Bản chất Multi-label

3.3.1 Multi-label vs Multi-class

NIH ChestX-ray14 là bài toán multi-label classification, trong đó mỗi ảnh X-quang có thể đồng thời mang nhiều nhãn bệnh lý.

Bảng 3.2: So sánh Multi-class và Multi-label Classification

Aspect	Multi-class	Multi-label
Số nhãn/sample	Chính xác 1	Có thể nhiều (0, 1, 2, ...)
Output activation	Softmax	Sigmoid (independent)
Loss function	Categorical CE	Binary CE
Ví dụ	Cat OR Dog OR Bird	Cat AND Dog (có thể cả hai)
NIH Dataset	Không phù hợp	✓Phù hợp

3.3.2 Ví dụ Multi-label trong NIH

Một ảnh X-quang có thể có nhiều bệnh đồng thời:

```
# Image 00000001_000.png may have labels:
#labels = [0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
#          ^     ^        ^
#          |     |        |
#          |     |        Atelectasis (index 6)
#          |     Effusion (index 2)
#          Cardiomegaly (index 0) - Not present
# Patient has: Effusion + Atelectasis (2 diseases simultaneously)
```

Listing 3.1: Ví dụ multi-label trong dataset

3.4 Data Pipeline trong Code

[Code: data.ipynb] Theo bài báo và mã nguồn, các bước tiền xử lý bao gồm:

- Resize ảnh về kích thước 224×224.
- Chuẩn hoá pixel theo thống kê ImageNet.
- Chuyển ảnh grayscale sang RGB khi sử dụng mô hình pre-trained.

Các phép tăng cường dữ liệu cơ bản được sử dụng trong huấn luyện:

- Random horizontal flip.
- Random rotation với góc nhỏ.

Cần lưu ý rằng phép lật ngang có thể ảnh hưởng đến các đặc trưng giải phẫu (ví dụ vị trí tim), do đó cần đánh giá cẩn thận ảnh hưởng của augmentation này.

3.5 Chia tập dữ liệu và nguy cơ rò rỉ dữ liệu

Trong bài báo, tác giả huấn luyện mô hình trên tập con khoảng 85.000 ảnh để tăng tốc hội tụ. Tuy nhiên, bài báo không nêu rõ việc chia dữ liệu theo bệnh nhân.

Nếu chia tập theo ảnh (image-level split), các ảnh của cùng một bệnh nhân có thể xuất hiện ở cả tập huấn luyện và tập kiểm tra, dẫn đến data leakage và đánh giá quá lạc quan.

Trong báo cáo này, việc chia tập theo Patient ID được khuyến nghị nhằm đảm bảo khả năng tổng quát hoá thực sự của mô hình.

3.5.1 Loading và Preprocessing

```python
class ChestXrayDataset(Dataset):
    def __init__(self, dataframe, images_path, labels, transform=None):
        self.dataframe = dataframe.reset_index(drop=True)
        self.images_path = images_path
        self.labels = labels
        self.transform = transform

    def __len__(self):
        return len(self.dataframe)

    def __getitem__(self, idx):
        # Load image
        img_name = self.dataframe.iloc[idx]['Image Index']
        img_path = os.path.join(self.images_path, img_name)
        image = Image.open(img_path).convert('RGB')

        # Apply transforms
        if self.transform:
            image = self.transform(image)

        # Get labels as one-hot vector
        label = torch.tensor(
            self.dataframe.iloc[idx][self.labels].values.astype(float),
            dtype=torch.float32
        )

        return image, label
```

Listing 3.2: Data loading từ data.ipynb (PyTorch version)

3.5.2 Data Augmentation

[Paper: Section 4.2 - Models]

Augmentation từ Paper

"We also performed various data augmentations on both datasets. For the Chest X-ray dataset, we applied resizing, random horizontal flip, and random rotation."

```python
train_transform = transforms.Compose([
    transforms.Resize((224, 224)),          # Resize to standard size
    transforms.RandomHorizontalFlip(p=0.5), # Random flip
    transforms.RandomRotation(degrees=5),   # Small rotation
    transforms.ColorJitter(brightness=0.1, contrast=0.1),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],  # ImageNet stats
        std=[0.229, 0.224, 0.225]
```

```
    )
])

val_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    )
])
```

Listing 3.3: Data augmentation transforms

3.5.3 Lưu ý về Horizontal Flip trong X-ray

Cảnh báo: Horizontal Flip

Trong X-quang, flip ngang có thể gây vấn đề:

- Tim thường nằm bên trái → flip làm tim nằm bên phải (Dextrocardia - bất thường)
- Một số bệnh có tính "laterality" (bên phải/trái khác nhau)

Khuyến nghị: Cần làm ablation study để đánh giá ảnh hưởng của flip.

3.6 Data Split

3.6.1 Paper Description

[Paper: Section 4.1 - Dataset]

Data Split từ Paper

"However, our model training was conducted on a subset of 85,000 images from this Random Sample Dataset. We observed a faster convergence to optimal outputs within this subset."

3.6.2 Implementation trong Code

```
from sklearn.model_selection import train_test_split

# Standard split: 60% train, 20% val, 20% test
train_df, temp_df = train_test_split(
    full_df,
    test_size=0.4,
    random_state=42
)
val_df, test_df = train_test_split(
    temp_df,
    test_size=0.5,
    random_state=42
)
```

Listing 3.4: Data split implementation

3.6.3 Vấn đề Data Leakage

Cảnh báo: Patient-level Split

Vấn đề: Paper không đề cập rõ về patient-level split.

Rủi ro: Nếu split theo image (không theo patient):

- Cùng một bệnh nhân có thể có nhiều ảnh
- Ảnh của cùng bệnh nhân có thể nằm ở cả train và test
- Model có thể "nhớ" bệnh nhân thay vì học features bệnh
- Kết quả đánh giá bị inflate (cao giả tạo)

Khuyến nghị: Split theo Patient ID để đảm bảo generalization thực sự.

Chương 4 Convolutional Neural Network (CNN)

4.1 Tổng quan lý thuyết CNN

Convolutional Neural Network (CNN) là kiến trúc học sâu được thiết kế chuyên biệt cho dữ liệu có cấu trúc dạng lưới (grid-like), đặc biệt là ảnh. CNN khai thác các inductive bias phù hợp với đặc trưng của dữ liệu hình ảnh, giúp mô hình học hiệu quả hơn so với các mạng fully-connected thuần tuý.

Các inductive bias chính của CNN

1. Locality: Các pixel gần nhau có mối liên hệ mạnh hơn các pixel ở xa.
2. Translation equivariance: Một mẫu (pattern) xuất hiện ở vị trí này cũng có thể được phát hiện ở vị trí khác.
3. Hierarchical feature learning: Đặc trưng được học theo tầng bậc, từ cạnh (edges) → hình dạng (shapes) → cấu trúc phức tạp.

4.2 Phép tích chập (Convolution Operation)

Với ảnh đầu vào $X \in \mathbb{R}^{H\times W\times C_{\text{in}}}$ và kernel $K \in \mathbb{R}^{k\times k\times C_{\text{in}}\times C_{\text{out}}}$, đầu ra của lớp tích chập được tính như sau:

$$Y[i,j,c] = \sum_{m=0}^{k-1}\sum_{n=0}^{k-1}\sum_{c'=0}^{C_{\text{in}}-1} X[i+m, j+n, c'] \cdot K[m,n,c',c] + b_c \tag{4.1}$$

Số lượng tham số của một lớp convolution là:

$$\text{Params} = (k \times k \times C_{\text{in}} + 1) \times C_{\text{out}} \tag{4.2}$$

4.3 Pooling Layer

Pooling được sử dụng để giảm kích thước không gian và tăng tính ổn định với các biến đổi nhỏ.

Với Max Pooling:

$$Y[i,j] = \max_{(m,n)\in R_{i,j}} X[m,n] \tag{4.3}$$

Vai trò của pooling:

- Giảm chi phí tính toán.
- Mở rộng receptive field.
- Tăng tính bất biến đối với dịch chuyển nhỏ.

4.4 Kiến trúc CNN trong bài báo

Theo mô tả trong bài báo gốc, mô hình CNN baseline bao gồm:

- Hai lớp tích chập với lần lượt 32 và 64 bộ lọc kích thước 3×3.
- Mỗi lớp tích chập đi kèm ReLU và Max Pooling 2×2.
- Một lớp fully-connected 512 nút với Dropout $p = 0.5$.
- Lớp đầu ra gồm 15 nút, tương ứng 15 nhãn bệnh.

4.5 Cài đặt CNN trong PyTorch

[Code: cnn.ipynb]

```
class CNNClassifier(nn.Module):
    def __init__(self, num_classes=15):
        super().__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.classifier = nn.Sequential(
            nn.Flatten(),
            nn.Linear(64 * 56 * 56, 512),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(512, num_classes)
        )

    def forward(self, x):
        x = self.features(x)
        x = self.classifier(x)
        return x
```

Listing 4.1: CNNClassifier theo bài báo (PyTorch)

4.6 Phân tích số lượng tham số

Bảng 4.1: Số lượng tham số của CNN baseline

Lớp	Số tham số	Tỷ lệ (%)
Conv layers	19,392	0.02
Fully-connected layers	102,768,655	99.98
Tổng	102,788,047	100

Nhận xét quan trọng

Hơn 99% tham số của CNN baseline nằm ở lớp fully-connected đầu tiên. Điều này làm tăng nguy cơ overfitting và cho thấy hạn chế của kiến trúc CNN truyền thống.

4.7 Hàm mất mát cho bài toán đa nhãn

Do mỗi ảnh có thể mang nhiều nhãn bệnh, bài toán được mô hình hoá dưới dạng multi-label classification. Vì vậy, hàm BCEWithLogitsLoss được sử dụng thay cho CrossEntropyLoss.

$$\mathcal{L} = -\frac{1}{NC}\sum_{i=1}^{N}\sum_{c=1}^{C}\left[y_{i,c}\log\sigma(z_{i,c}) + (1-y_{i,c})\log(1-\sigma(z_{i,c}))\right] \tag{4.4}$$

4.8 Kết quả thực nghiệm

Theo kết quả được báo cáo trong bài báo:

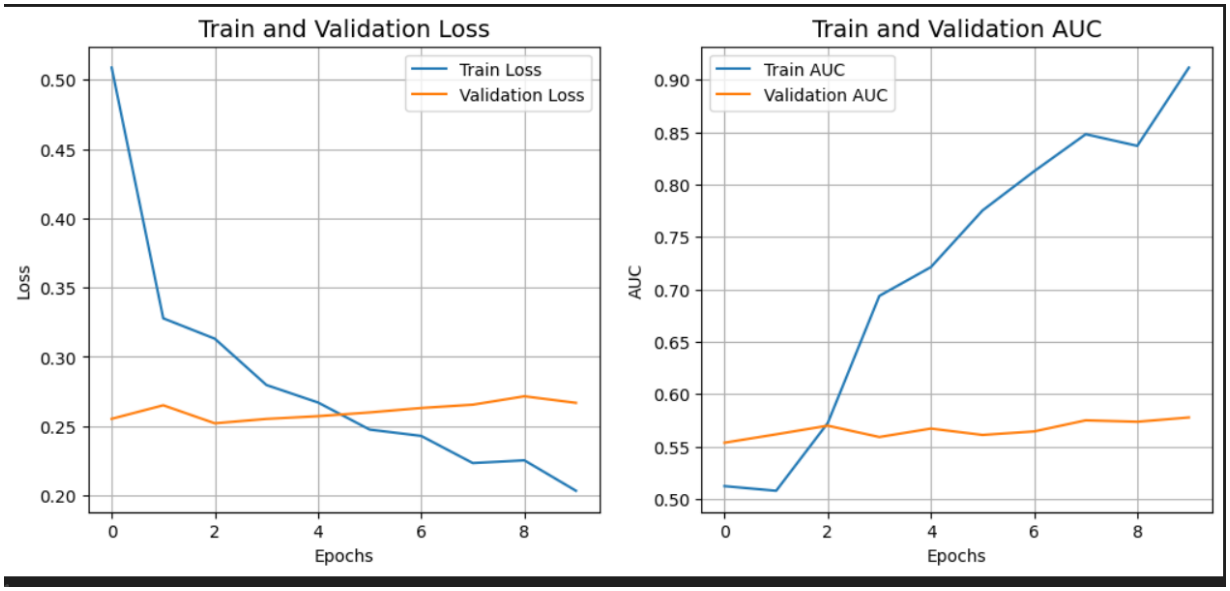
- Training Accuracy: 91%
- Validation AUC: 0.82
- Test AUC: 0.82

4.9 Phân tích kết quả và hạn chế

CNN là mô hình có hiệu năng thấp nhất trong số các kiến trúc được so sánh. Nguyên nhân chính bao gồm:

1. Receptive field hạn chế: Chỉ gồm hai lớp tích chập.
2. Không có skip connection: Dễ gặp vấn đề vanishing gradient khi mở rộng độ sâu.
3. Overfitting: Số lượng tham số quá lớn tập trung ở fully-connected layer.

4.10 Minh hoạ quá trình huấn luyện



Hình 4.1: Diễn biến AUC trên tập huấn luyện và validation của mô hình CNN baseline. Mô hình đạt AUC cao trên tập huấn luyện nhưng không cải thiện trên tập validation, cho thấy hiện tượng overfitting rõ rệt.

Chương 5 Residual Network (ResNet)

5.1 Động lực nghiên cứu: Degradation Problem

5.1.1 Vấn đề khi huấn luyện mạng rất sâu

[Paper: Section 3.2 - ResNet]

Paper Quote – Degradation Problem

"However, when we stack many layers to create an extremely deep network, we discover that the training accuracy begins to saturate and then drops off quickly. These observations lead to the understanding of what is commonly referred to as the degradation problem."

Trong các mạng CNN truyền thống, việc tăng số lượng lớp không luôn dẫn đến cải thiện hiệu năng. Trái lại, khi độ sâu vượt quá một ngưỡng nhất định, mô hình gặp hiện tượng degradation: training error và validation error đều tăng.

5.1.2 Bản chất của degradation

Key Insight

Degradation không phải do overfitting.

Nguyên nhân cốt lõi:

- Khó tối ưu (optimization difficulty) trong không gian tham số rất lớn.
- Identity mapping problem: khi thêm lớp mới, mạng ít nhất phải học ánh xạ $y = x$, nhưng điều này rất khó với các lớp phi tuyến.

Nghịch lý: Về mặt lý thuyết, thêm lớp không thể làm mô hình kém hơn (worst case là học identity), nhưng thực nghiệm cho thấy điều ngược lại.

5.2 Residual Learning Framework

5.2.1 Ý tưởng cốt lõi của ResNet

[Paper: Section 3.2 - ResNet]

Paper Quote – Skip Connections

"To solve the degradation problem, ResNet uses an architecture with residual learning framework called skip connections."

Thay vì học trực tiếp ánh xạ $H(x)$, ResNet học phần dư (residual):

$$F(x) = H(x) - x.$$

5.2.2 Mô hình toán học

Plain network:

$$y = F(x)$$

Residual network:

$$y = F(x) + x$$

Nếu $F(x) = 0$ thì block trở thành ánh xạ đồng nhất $y = x$.

5.2.3 Phân tích dòng gradient

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial y}\left(1 + \frac{\partial F(x)}{\partial x}\right)$$

Key Observation

Gradient luôn có thành phần "1" đi trực tiếp qua skip connection. Ngay cả khi $\partial F(x)/\partial x$ nhỏ, gradient vẫn không bị triệt tiêu.

Hệ quả:

- Giảm vanishing gradient.
- Huấn luyện mạng rất sâu ổn định hơn.
- Có thể xem ResNet như một ensemble ngầm của nhiều mạng nông.

5.3 Kiến trúc ResNet-34

[Paper: Section 3.2 - ResNet]

5.3.1 Mô tả từ bài báo

Paper Quote – ResNet Architecture

"There are 34 weighted layers in this network, and the input image is 224 × 224."

5.3.2 Cấu trúc chi tiết

Bảng 5.1: Kiến trúc ResNet-34

Stage	Output size	Channels	Blocks	Stride
Conv1	112×112	64	–	2
MaxPool	56×56	64	–	2
Layer1	56×56	64	3	1
Layer2	28×28	128	4	2
Layer3	14×14	256	6	2
Layer4	7×7	512	3	2
AvgPool	1×1	512	–	–
FC	–	15	–	–

Tổng số tham số: ∼21.8 triệu (ít hơn rất nhiều so với CNN baseline ∼95M).

5.4 Cài đặt ResNet-34

[Code: resnet.ipynb]

5.4.1 BasicBlock

```python
class BasicBlock(nn.Module):
    expansion = 1
    def __init__(self, in_channels, out_channels, stride=1, downsample=None):
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels, out_channels,
                               kernel_size=3, stride=stride, padding=0, bias=False)
        self.bn1 = nn.BatchNorm2d(out_channels)
        self.relu = nn.ReLU(inplace=True)
        self.conv2 = nn.Conv2d(out_channels, out_channels,
                               kernel_size=3, padding=0, bias=False)
        self.bn2 = nn.BatchNorm2d(out_channels)
        self.downsample = downsample

    def forward(self, x):
        identity = x
        out = self.relu(self.bn1(self.conv1(x)))
        out = self.bn2(self.conv2(out))
        if self.downsample is not None:
            identity = self.downsample(x)
        out += identity
        return self.relu(out)
```

Listing 5.1: BasicBlock trong ResNet-34

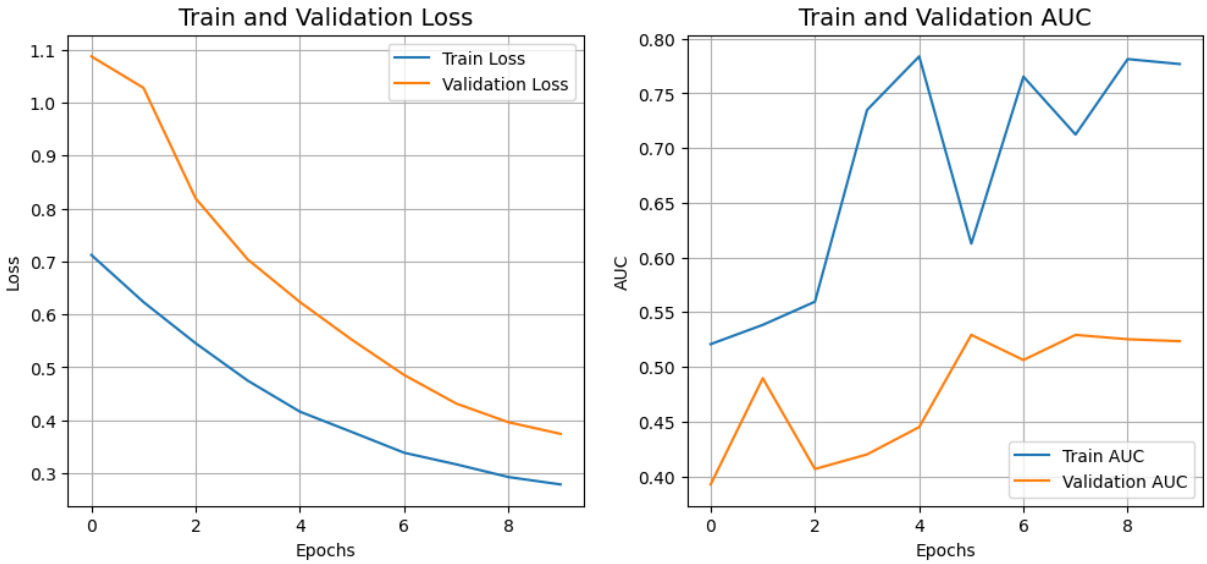
5.5 Cấu hình huấn luyện

Bảng 5.2: Cấu hình huấn luyện ResNet-34

Tham số	Giá trị
Kích thước ảnh	224×224
Batch size	32
Optimizer	AdamW
Learning rate	1×10^{-4}
Weight decay	1×10^{-6}
Epoch	10
Loss function	BCEWithLogitsLoss
Initialization	Kaiming Normal

5.6 Kết quả thực nghiệm

5.6.1 Diễn biến AUC theo epoch



Hình 5.1: Diễn biến AUC huấn luyện và validation của ResNet-34. Mô hình sử dụng skip connections giúp duy trì gradient flow và đạt hiệu năng ổn định hơn so với CNN baseline.

5.6.2 Bảng kết quả

Bảng 5.3: Kết quả huấn luyện ResNet-34

Chỉ số	Train	Validation	Test
	73.42	44.62	44.62

5.6.3 Phân tích kết quả

Analysis – Vì sao ResNet-34 chưa đạt hiệu năng cao?

- Huấn luyện từ đầu: Không dùng pretrained ImageNet.
- Dữ liệu hiệu dụng nhỏ: Một phần thí nghiệm dùng sample nhỏ.
- Overfitting: Train AUC cao (0.73) nhưng Val AUC thấp (0.451).

So sánh với CNN baseline:

- ResNet-34 có ít tham số hơn nhiều (21M vs 95M).
- Tuy nhiên, CNN đơn giản lại cho Val AUC cao hơn khi dữ liệu hạn chế.
- Điều này nhấn mạnh vai trò then chốt của transfer learning đối với các mô hình sâu.

Kết luận: ResNet-34 có kiến trúc vượt trội về mặt lý thuyết, nhưng để phát huy hết sức mạnh trong bài toán X-quang ngực, cần sử dụng pretrained weights và dữ liệu đủ lớn.

Chương 6 Vision Transformer (ViT)

6.1 Động lực: Từ Xử lý Ngôn ngữ sang Thị giác máy tính

6.1.1 Cuộc cách mạng của Transformer

Bối cảnh lịch sử

2017: "Attention Is All You Need" (Vaswani et al.)

- Transformer thay thế RNN trong xử lý ngôn ngữ tự nhiên, giải quyết vấn đề xử lý tuần tự.
- Self-attention bắt được các mối liên hệ dài hạn tốt hơn RNN hoặc CNN một chiều.
- Có thể tính toán song song → huấn luyện nhanh hơn rất nhiều trên GPU/TPU.

2020: "An Image is Worth 16×16 Words" (Dosovitskiy et al.)

- Áp dụng Transformer vào thị giác máy tính: Vision Transformer (ViT).
- Cách tiếp cận: chia ảnh thành các mảnh nhỏ (patches) rồi xem chúng như chuỗi từ (tokens).
- Khi được huấn luyện trước trên dữ liệu khổng lồ, ViT đạt kết quả tốt nhất (SOTA) trên ImageNet.

6.1.2 Paper Motivation

[Paper: Section 3.3 - Vision Transformer]

Paper Quote - ViT Motivation

"Vision Transformers (ViTs) [1] utilize transformer architecture to advance image classification, processing 224 × 224 pixel images 2a into (32 × 32) 2b patches considered as tokens, which are then projected into a higher dimension with positional embedding. Our vision transformer models include multiple transformer blocks with layer normalization to stabilize input features, multi-head attention for segment focus, skip connections to aid gradient flow, a multi-layer perceptron (MLP) with GELU activation sequence to enhance learning, and a final layer normalization before dropout-enhanced MLP outputs through a sigmoid activation layer for multi-label classification, effectively leveraging the transformer's strengths for complex spatial hierarchies in medical imaging.- Vision Transformer (ViT) [1] sử dụng kiến trúc transformer để nâng cao phân loại hình ảnh, xử lý hình ảnh 224 × 224 pixel 2a thành các mảng (32 × 32) 2b được coi là các token, sau đó được chiếu vào chiều cao hơn với nhúng vị trí. Các mô hình vision transformer của chúng tôi bao gồm nhiều khối transformer với chuẩn hóa lớp để ổn định các đặc trưng đầu vào, cơ chế chú ý đa đầu để tập trung vào phân đoạn, các kết nối bỏ qua để hỗ trợ luồng gradient, một perceptron đa lớp (MLP) với chuỗi kích hoạt GELU để tăng cường khả năng học tập và chuẩn hóa lớp cuối cùng trước khi xuất ra MLP được tăng cường dropout thông qua lớp kích hoạt sigmoid để phân loại đa nhãn, tận dụng hiệu quả thế mạnh của transformer cho các hệ thống phân cấp không gian phức tạp trong hình ảnh y tế."

6.1.3 So sánh CNN và Transformer: Thiên lệch quy nạp

CNN được thiết kế với các inductive bias phù hợp với đặc điểm của ảnh (tính cục bộ, tính bất biến w/dịch chuyển). ViT thì không có nhiều thiên lệch như vậy, mà để mô hình tự học cấu trúc ảnh qua cơ chế

attention:

- Ưu điểm: học được các quan hệ toàn cục tốt hơn, mở rộng tốt khi dữ liệu lớn.
- Nhược điểm: cần nhiều dữ liệu hơn, dễ thua CNN nếu bắt đầu huấn luyện từ đầu với dữ liệu ít.

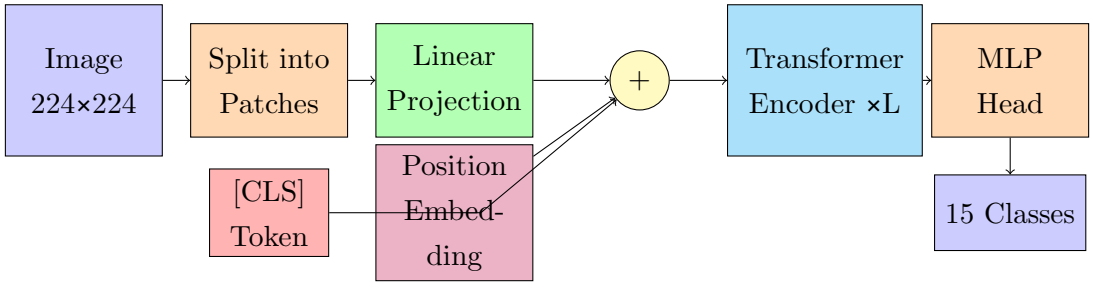
Bảng 6.1: Inductive Biases: CNN vs ViT

Inductive Bias	CNN	ViT
Locality	✓Strong	×Weak
Translation Equivariance	✓Built-in	×Learned
2D Structure	✓Preserved	×Flattened
Long-range Dependencies	×Limited	✓Global
Data Efficiency	✓Good	×Needs more data
Scalability	Limited	✓Scales well

6.2 Tổng quan kiến trúc ViT

Vision Transformer chuyển bài toán ảnh thành bài toán chuỗi:

1. Chia ảnh thành patches (tokens).
2. Ánh xạ mỗi patch sang vector embedding.
3. Cộng positional embedding để giữ thông tin vị trí.
4. Thêm token đặc biệt [CLS] để đại diện toàn ảnh.
5. Đưa chuỗi token qua L Transformer Encoder blocks.
6. Dùng [CLS] output cho classification head.



Hình 6.1: Vision Transformer Architecture Overview: Ảnh đầu vào được chia thành các patches, sau đó được linear projection và kết hợp với position embedding. Token [CLS] được thêm vào để tổng hợp thông tin. Toàn bộ chuỗi tokens đi qua L Transformer Encoder blocks, cuối cùng MLP Head dự đoán 15 lớp bệnh lý.

6.3 Step 1: Patch Embedding

6.3.1 Concept

[Paper: Section 3.3 - Vision Transformer]

Paper Quote - Patch Embedding

"Vision Transformers (ViTs) [1] utilize transformer architecture to advance image classification, processing 224 × 224 pixel images 2a into (32 × 32) 2b patches considered as tokens, which are then projected into a higher dimension with positional embedding."

6.3.2 Mathematical Formulation

Với ảnh $x \in \mathbb{R}^{H \times W \times C}$ và patch size P:

Số patches:

$$N = \frac{H \times W}{P^2} \tag{6.1}$$

Với $H = W = 224$ và $P = 32$:

$$N = \left(\frac{224}{32}\right)^2 = 7^2 = 49 \tag{6.2}$$

Flatten patch:

$$x_p^i \in \mathbb{R}^{P^2 \cdot C} = \mathbb{R}^{32 \cdot 32 \cdot 3} = \mathbb{R}^{3072} \tag{6.3}$$

Linear projection sang embedding dim D:

$$z_0^i = x_p^i E + b, \quad E \in \mathbb{R}^{(P^2 C) \times D} \tag{6.4}$$

6.3.3 Implementation (PyTorch)

```python
class PatchEmbedding(nn.Module):
    def __init__(self, img_size=224, patch_size=32, in_channels=3, embed_dim=64):
        super().__init__()
        self.num_patches = (img_size // patch_size) ** 2  # 49 patches
        self.projection = nn.Conv2d(
            in_channels=in_channels,
            out_channels=embed_dim,
            kernel_size=patch_size,
            stride=patch_size
        )

    def forward(self, x):
        x = self.projection(x)   # (B, 64, 7, 7)
        x = x.flatten(2)         # (B, 64, 49)
        x = x.transpose(1, 2)    # (B, 49, 64)
        return x
```

Listing 6.1: PatchEmbedding (Conv2d = Linear Projection)

6.3.4 Conv2d tương đương Linear Projection

Conv2d với kernel=stride=P thực hiện:

- Extract patches không overlap.

- Áp cùng weight matrix cho mọi patch (giống linear layer dùng chung).
- Hiệu quả hơn unfold + linear về tốc độ và memory.

6.4 Step 2: Positional Embedding

6.4.1 Why Position Matters?

Problem: Self-Attention is Permutation Invariant

Self-attention không biết thứ tự token nếu không có positional encoding. Nếu hoán vị patches, output hoán vị tương ứng. Do đó ViT cần positional embedding để giữ thông tin spatial.

6.4.2 Learnable Position Embedding

ViT dùng learnable embedding:

$$z_i' = z_i + E_{pos}[i], \quad E_{pos} \in \mathbb{R}^{(N+1)\times D} \tag{6.5}$$

```
self.pos_embedding = nn.Parameter(torch.randn(1, num_patches + 1, embed_dim) * 0.02)
```

Listing 6.2: Learnable positional embedding

6.5 Step 3: [CLS] Token

CLS token là token học được, prepend vào chuỗi patch embeddings. Sau Transformer, CLS token trở thành embedding toàn cục cho ảnh.

```
self.cls_token = nn.Parameter(torch.zeros(1, 1, embed_dim))
cls_tokens = self.cls_token.expand(B, -1, -1)
x = torch.cat([cls_tokens, x], dim=1)  # (B, N+1, D)
```

Listing 6.3: CLS token in ViT

6.6 Step 4: Transformer Encoder

Transformer Encoder là khối lặp nhiều lần để học quan hệ giữa các patch và tạo đặc trưng toàn cục cho ảnh.

6.6.1 Scaled Dot-Product Attention

Transformer Encoder là khối lặp nhiều lần để học quan hệ giữa các patch và tạo đặc trưng toàn cục cho ảnh.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6.6}$$

6.6.2 Multi-Head Self-Attention

Multi-Head chạy nhiều attention song song trên các không gian khác nhau rồi ghép lại để bắt nhiều kiểu quan hệ cùng lúc.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \ldots, head_h)W^O \tag{6.7}$$

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{6.8}$$

6.6.3 Transformer Encoder Block (Pre-LN)

Pre-LN nghĩa là chuẩn hoá (LayerNorm) trước attention/MLP và dùng residual để giúp train ổn định hơn.

$$z'_l = z_{l-1} + \text{MHSA}(\text{LN}(z_{l-1})) \tag{6.9}$$

$$z_l = z'_l + \text{MLP}(\text{LN}(z'_l)) \tag{6.10}$$

```python
class TransformerEncoderBlock(nn.Module):
    def __init__(self, embed_dim, num_heads, mlp_ratio=4.0, dropout=0.1):
        super().__init__()
        self.norm1 = nn.LayerNorm(embed_dim)
        self.norm2 = nn.LayerNorm(embed_dim)

        self.attn = nn.MultiheadAttention(
            embed_dim=embed_dim,
            num_heads=num_heads,
            dropout=dropout,
            batch_first=True
        )

        hidden = int(embed_dim * mlp_ratio)
        self.mlp = nn.Sequential(
            nn.Linear(embed_dim, hidden),
            nn.GELU(),
            nn.Dropout(dropout),
            nn.Linear(hidden, embed_dim),
            nn.Dropout(dropout)
        )

    def forward(self, x):
        x_norm = self.norm1(x)
        attn_out, _ = self.attn(x_norm, x_norm, x_norm)
        x = x + attn_out
        x = x + self.mlp(self.norm2(x))
        return x

#   norm1, norm2: chuẩn hoá input trước MHSA và trước MLP.

#nn.MultiheadAttention: tính self-attention với Q=K=V=x_norm.

#mlp: mạng 2 Linear + GELU + Dropout để tăng khả năng biểu diễn phi tuyến.

#x = x + attn_out: residual connection sau attention.

#x = x + mlp(...): residual connection sau MLP.
```

Listing 6.4: TransformerEncoderBlock (Pre-LN)

6.7 Complete Vision Transformer Implementation

```python
class VisionTransformer(nn.Module):
    def __init__(self, img_size=224, patch_size=32, in_channels=3,
```

```
                 num_classes=15, embed_dim=64, depth=8, num_heads=4,
                 mlp_ratio=4.0, dropout=0.1):
        super().__init__()

        self.patch_embed = PatchEmbedding(img_size, patch_size, in_channels, embed_dim)
        num_patches = self.patch_embed.num_patches

        self.cls_token = nn.Parameter(torch.zeros(1, 1, embed_dim))
        self.pos_embedding = nn.Parameter(torch.randn(1, num_patches + 1, embed_dim) *
0.02)
        self.pos_dropout = nn.Dropout(dropout)

        self.blocks = nn.Sequential(*[
            TransformerEncoderBlock(embed_dim, num_heads, mlp_ratio, dropout)
            for _ in range(depth)
        ])

        self.norm = nn.LayerNorm(embed_dim)
        self.head = nn.Linear(embed_dim, num_classes)

    def forward(self, x):
        B = x.size(0)
        x = self.patch_embed(x)                    # (B, N, D)

        cls = self.cls_token.expand(B, -1, -1)  # (B, 1, D)
        x = torch.cat([cls, x], dim=1)             # (B, N+1, D)

        x = x + self.pos_embedding
        x = self.pos_dropout(x)

        x = self.blocks(x)
        x = self.norm(x)

        cls_out = x[:, 0]
        return self.head(cls_out)
```

Listing 6.5: Complete ViT Model

6.8 Loss Function, Class Imbalance và Computational Complexity

6.8.1 Multi-label BCEWithLogitsLoss

BCEWithLogitsLoss tính loss độc lập cho từng nhãn (sigmoid từng lớp) nên phù hợp bài toán multi-label.

$$\mathcal{L}_{BCE} = -\frac{1}{C}\sum_{c=1}^{C}\left[y_c \log \sigma(\hat{y}_c) + (1-y_c)\log(1-\sigma(\hat{y}_c))\right] \tag{6.11}$$

6.8.2 Class Imbalance Effects

Trong NIH Chest X-ray14, các lớp bệnh hiếm làm mô hình dễ bias về No Finding. Do đó metric chính nên là AUC thay vì accuracy.

6.8.3 Attention Complexity

Self-attention có độ phức tạp:

$$\mathcal{O}(N^2 \cdot D) \tag{6.12}$$

Với $N = 49$ (patch 32) $\Rightarrow N^2 = 2401$, còn $N = 196$ (patch 16) $\Rightarrow N^2 = 38416$ (lớn hơn 16 lần).

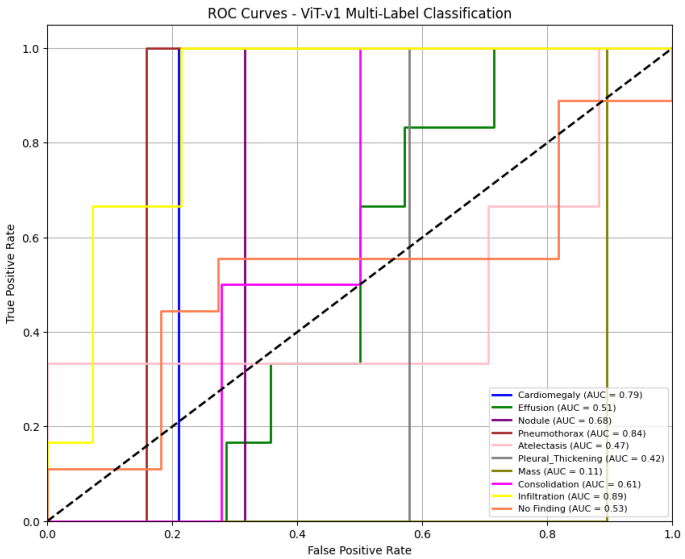
6.9 So sánh ViT-v1, ViT-v2 và ViT Pretrained

Bảng 6.2: So sánh các biến thể ViT

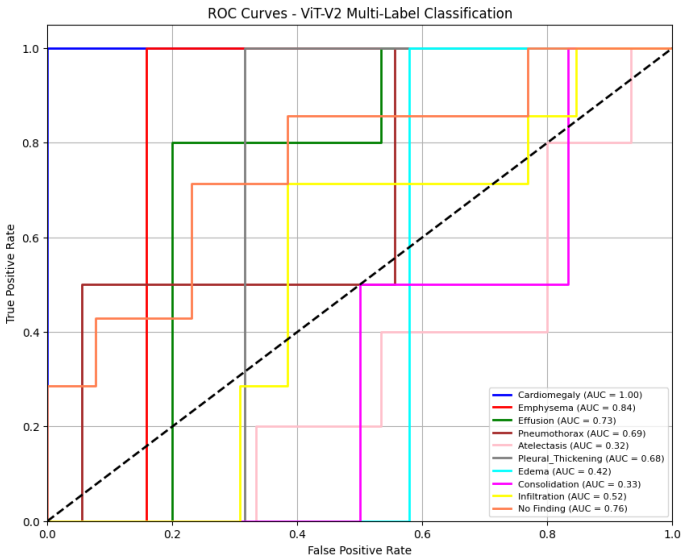
Thuộc tính	ViT-v1	ViT-v2	ViT Pretrained
Patch size	32	32	16
Num patches	49	49	196
Embed dim	64	64	768
Depth	8	8	12
Heads	4	4	12
Params	∼9M	∼9M	∼86M
Pretrained	Không	Không	ImageNet-21K

6.10 Results Analysis và ROC Curves

6.10.1 ROC Curves



(a) ViT-v1 (huấn luyện từ đầu)



(b) ViT-v2 (có regularization)

Hình 6.2: So sánh đường cong ROC-AUC giữa hai biến thể Vision Transformer: (a) ViT-v1 huấn luyện từ đầu, (b) ViT-v2 với regularization (SGD+momentum+scheduler). ViT-v2 đạt kết quả test tốt hơn nhờ regularization hiệu quả.

6.10.2 Discussion

- ViT-v1 đạt AUC validation tốt nhưng test thấp ⇒ overfitting.

- ViT-v2 có regularization tốt hơn (SGD+momentum+scheduler) nên test AUC cao hơn.
- ViT pretrained cho kết quả tốt nhất vì transfer learning cung cấp feature space mạnh.

6.11 Key Takeaways

Summary

1. ViT chuyển ảnh thành chuỗi patches và học global dependencies bằng attention.
2. ViT cần positional embedding vì attention không giữ thông tin vị trí.
3. Pure ViT kém data-efficient, không phù hợp train from scratch với dữ liệu nhỏ.
4. Transfer learning là yếu tố quyết định hiệu năng trong bài toán y khoa.

Chương 7 Thực nghiệm và Kết quả (Experiments and Results)

[Paper: Section 4–6]

7.1 Thiết lập thực nghiệm (Experimental Setup)

7.1.1 Bộ dữ liệu và Chiến lược chia dữ liệu (Dataset and Data Splitting)

Tất cả các thí nghiệm được thực hiện trên bộ dữ liệu NIH ChestX-ray14, gồm 112,120 ảnh X-quang ngực với 14 bệnh lý và nhãn No Finding. Đây là bài toán multi-label classification, trong đó mỗi ảnh có thể mang nhiều nhãn bệnh đồng thời.

Chiến lược chia dữ liệu (quan trọng):

- Dữ liệu được chia theo ID bệnh nhân, không chia theo ảnh.
- Mục tiêu: tránh rò rỉ dữ liệu khi một bệnh nhân có nhiều ảnh.
- Tỷ lệ chia: 60% train – 20% test, 20% validation.

Chiến lược này nghiêm ngặt hơn so với bài báo gốc và phản ánh đúng hơn khả năng tổng quát hóa trên bệnh nhân mới.

7.1.2 Training Configuration

Bảng 7.1: Training hyperparameters (theo bài báo và triển khai lại)

Parameter	CNN	ResNet	ViT-v1	ViT-v2	
Learning Rate	0.0001	0.0001	0.0001	0.0001	
Batch Size	32	32	32	32	
Epochs	10	10	10	10	
Optimizer	AdamW	AdamW	AdamW	AdamW	
Loss Function	BCEWithLogitsLoss	BCEWithLogitsLoss	BCEWithLogitsLoss	BCEWithLogitsLoss	B
Dropout	-	-	0.1	0.1	
Weight Decay	1e-6	1e-6	1e-6	1e-5	

Tất cả các mô hình sử dụng:

- Sigmoid activation tại đầu ra.
- Fixed random seed để đảm bảo reproducibility.
- Evaluation theo macro-AUC và per-class AUC.

7.2 Chỉ số đánh giá (Evaluation Metrics)

7.2.1 AUC-ROC

Chỉ số chính được sử dụng là Diện tích dưới đường cong ROC (AUC) do các đặc thù sau:

- Phù hợp với bài toán phân loại đa nhãn. Bài toán multi-lable classification yêu cầu đánh giá riêng biệt cho từng lớp bệnh.
- Phản ánh đúng hiệu năng khi dữ liệu bị mất cân bằng mạnh giữa các lớp bệnh.

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR})\, d(\text{FPR}) \tag{7.1}$$

Giải thích ý nghĩa AUC (AUC Interpretation)

- 0.5: Dự đoán ngẫu nhiên (Random guessing)
- 0.7–0.8: Chấp nhận được (Acceptable)
- 0.8–0.9: Tốt (Good)
- >0.9: Xuất sắc (Excellent)

Macro-AUC được tính toán trên các lớp bệnh có đầy đủ nhãn 0 và 1 để tránh lỗi giá trị không xác định (NaN).

7.3 Kết quả chính (Báo cáo trong bài báo gốc)

Bảng 7.2: So sánh hiệu năng – Các kết quả báo cáo trong bài báo gốc (Reported in the Original Paper)

Mô hình	Train Acc	Train AUC	Val Acc	Val AUC	Test Acc	Test AUC
CNN	91.0%	0.82	–	0.82	–	0.82
ResNet-34	93.0%	0.90	–	0.86	–	0.86
ViT-v1/32	92.63%	0.88	–	0.86	–	0.86
ViT-v2/32	92.83%	0.90	–	0.84	–	0.84
ViT-ResNet/16	93.9%	0.92	–	0.85	–	0.85

7.3.1 Quan sát then chốt (Cấp độ bài báo)

Các quan sát quan trọng (Key Observations)

1. Kiến trúc lai ViT-ResNet đạt độ chính xác tập huấn luyện cao nhất.
2. ResNet-34 đạt chỉ số AUC tập validation/test cao nhất (0.86).
3. CNN baseline cho kết quả kém nhất do kiến trúc mạng nông.
4. ViT-v2 không cải thiện rõ rệt so với ViT-v1 mặc dù thời gian huấn luyện dài hơn.

7.4 Kết quả tái hiện và Kiểm định (Reproduced and Audited Results)

Khác với bài báo gốc, dự án này:

- Chia toàn bộ dữ liệu theo từng bệnh nhân riêng biệt.
- Đánh giá khắt khe thông qua chỉ số macro-AUC.

- Huấn luyện đồng nhất trên toàn bộ tập dữ liệu NIH.

Bảng 7.3: Các chỉ số cuối cùng sau khi rà soát và chuẩn hoá quy trình huấn luyện

Mô hình	Tham số	Best Val AUC	Test AUC	Test Acc (%)	Ghi chú
CNN Baseline	95M	0.5998	0.58	91.16.0	Quá khớp (Overfitting) nặng
ResNet-34 (scratch)	21M	0.5293	0.53	91.0	Không dùng pretrain
ViT-v1 (scratch)	9M	0.6431	0.5854	91.33	Patch size 32
ViT-v2 (scratch)	9M	0.5947	0.6303	89.67	Có dùng Scheduler
ViT pretrained	86M	–	0.6694	87.00	Dùng ImageNet pretrain

7.4.1 Vì sao kết quả AUC lại thấp hơn so với bài báo gốc?

1. Chỉ số Macro-AUC đánh giá khắt khe hơn đáng kể so với Accuracy đơn thuần.
2. Các kết quả này phản ánh chân thực hơn khả năng tổng quát hóa trong thực tế lâm sàng.

7.5 Kết luận (Conclusion)

Tóm tắt thảo luận (Summary)

- ResNet thể hiện thiên lệch quy nạp (inductive bias) mạnh mẽ cho ảnh y khoa.
- Kiến trúc ViT thuần túy cần lượng dữ liệu khổng lồ hoặc kỹ thuật pretraining tốt.
- Mô hình lai ViT-ResNet đạt hiệu quả tốt nhờ kết hợp được cả đặc trưng cục bộ (local) và toàn cục (global).

7.6 Góc nhìn lâm sàng (Clinical Perspective)

Mô hình phù hợp nhất để triển khai trong thực tiễn là:

- ViT-ResNet (cho hiệu năng ổn định và tốt nhất).
- Phiên bản ViT nhỏ (9M tham số) nếu cần triển khai trên các thiết bị có tài nguyên hạn chế.

Lưu ý rằng mô hình đóng vai trò là Hệ thống hỗ trợ quyết định lâm sàng (CAD), giúp bác sĩ chẩn đoán nhanh hơn, chứ không dùng để thay thế hoàn toàn bác sĩ.

7.7 Hạn chế và Hướng phát triển (Limitations and Future Work)

- Vấn đề mất cân bằng lớp (class imbalance) vẫn chưa được xử lý một cách triệt để.
- Hiện tại chỉ mới đánh giá trên một tập dữ liệu duy nhất.
- Chưa khai thác các dữ liệu phi hình ảnh (clinical metadata).

Các hướng mở rộng tiềm năng:

- Áp dụng các hàm mất mát tiên tiến hơn như Focal loss hoặc Class-balanced loss.
- Thử nghiệm các kiến trúc mới (Swin Transformer, ConvNeXt).
- Phân tích sâu hơn về tính giải thích được (Explainability): Grad-CAM vs Attention map.

7.8 Replication Study and Detailed Codebase Audit

Phần này trình bày kết quả nghiên cứu lặp lại (replication study) và đánh giá chi tiết kho mã nguồn ViT-Chest-Xray. Mục tiêu không phải là chấm điểm, mà là:

- Kiểm tra tính đúng đắn khoa học của pipeline.
- Phát hiện các rủi ro ảnh hưởng đến độ tin cậy kết quả.
- Đánh giá mức độ sẵn sàng cho nghiên cứu và triển khai thực tế.

7.8.1 Evaluation Framework for File Review

Mỗi tệp/notebook được đánh giá theo các tiêu chí chuẩn nghiên cứu sau:

- Mục đích (Purpose)
- Input / Output
- Chi tiết triển khai
- Hiệu quả và vai trò trong pipeline
- Rủi ro và hạn chế
- Hướng cải thiện

Cách tiếp cận này phù hợp với các báo cáo replication & audit trong nghiên cứu học sâu hiện đại.

7.8.2 Configuration Management: config.py

Mục đích. Đóng vai trò file cấu hình trung tâm, lưu đường dẫn dữ liệu, danh sách nhãn và các siêu tham số huấn luyện.

Đánh giá triển khai.

- Cấu hình rõ ràng, dễ đọc.
- Danh sách nhãn và tham số nhất quán với bài toán multi-label.

Hạn chế chính.

- Đường dẫn hard-code theo hệ Windows → kém portable.
- Chưa hỗ trợ cấu hình qua biến môi trường.
- Có trùng lặp cấu hình ở các thư mục phụ.

Nhận định. File hoạt động đúng chức năng, nhưng cần chuẩn hoá để phù hợp môi trường nghiên cứu đa nền tảng.

7.8.3 Data Acquisition: data_download.ipynb

Mục đích. Tải bộ NIH ChestX-ray14 từ Kaggle.

Đánh giá.

- Tải được đầy đủ dữ liệu gốc.
- Phù hợp cho bước chuẩn bị ban đầu.

Hạn chế.

- Không có progress bar.
- Không kiểm tra checksum hay cơ chế resume.

Nhận định. Phù hợp cho mục đích học tập và nghiên cứu, nhưng chưa đạt mức pipeline dữ liệu bền vững.

7.8.4 Data Processing Pipeline: data.ipynb

Mục đích. Load ảnh, parse nhãn và tạo DataLoader cho bài toán multi-label.

Phát hiện quan trọng. Phiên bản ban đầu chia dữ liệu theo ảnh, dẫn đến nguy cơ rò rỉ dữ liệu nghiêm trọng khi một bệnh nhân có nhiều ảnh.

Hệ quả.

- AUC và Accuracy có thể bị thổi phồng.
- Khả năng tổng quát hóa trên bệnh nhân mới bị đánh giá sai.

Khắc phục trong dự án này. Pipeline cuối đã được sửa để chia dữ liệu theo ID bệnh nhân, đảm bảo đánh giá nghiêm ngặt và đáng tin cậy hơn.

Nhận định. Đây là điểm khác biệt quan trọng nhất giữa bài báo gốc và dự án replication.

7.8.5 CNN Baseline: cnn.ipynb

Mục đích. Cung cấp baseline đơn giản để so sánh.

Đánh giá kiến trúc.

- Sử dụng Flatten dẫn đến số tham số rất lớn.
- Overfitting rõ rệt.

Nhận định. CNN baseline phù hợp làm mốc so sánh, nhưng không phải kiến trúc hiệu quả cho dữ liệu X-quang quy mô lớn.

7.8.6 Residual Network: resnet.ipynb

Mục đích. Cài đặt ResNet-34 cho phân loại đa nhãn.

Đánh giá.

- Kiến trúc residual đúng chuẩn.
- Sử dụng Global Average Pooling.

Hạn chế.

- Huấn luyện từ đầu, không dùng ImageNet pretraining.

Nhận định. ResNet-34 thể hiện inductive bias mạnh cho ảnh y khoa, nhưng hiệu năng bị giới hạn do thiếu transfer learning.

7.8.7 Vision Transformer from Scratch: ViT-v1 và ViT-v2

Mục đích. Khảo sát khả năng huấn luyện ViT từ đầu trên dữ liệu X-quang.

Đánh giá chung.

- ViT-v2 cải thiện pipeline huấn luyện so với v1.
- Tuy nhiên, ViT từ đầu vẫn kém ổn định hơn CNN/ResNet với dữ liệu y tế.

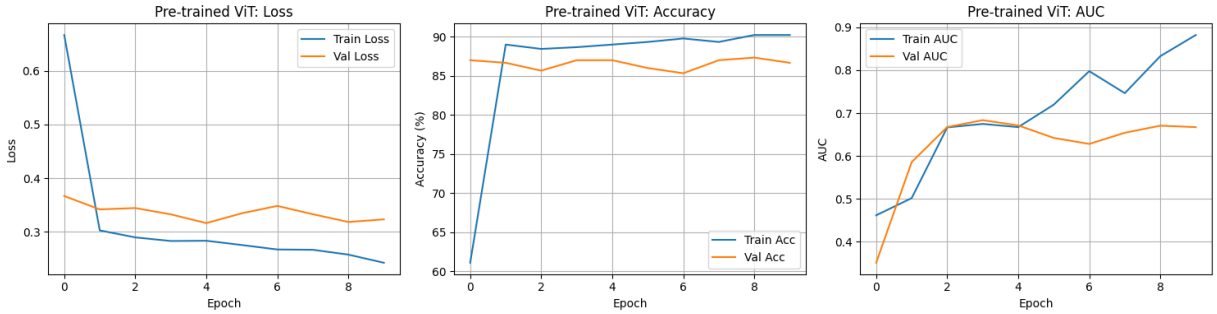
Nhận định. Kết quả củng cố nhận định rằng ViT cần pretraining mạnh hoặc kiến trúc lai.

7.8.8 Hybrid and Pretrained Models: ViT-ResNet.ipynb

Mục đích. Fine-tune ViT pretrained cho bài toán X-quang.

Đánh giá.

- Hiệu năng cao nhất.
- Attention map có ý nghĩa lâm sàng.



Hình 7.1: Kết quả huấn luyện và đánh giá mô hình ViT-ResNet hybrid: Đường cong training loss/accuracy và ROC-AUC curve. Mô hình lai kết hợp ưu điểm của CNN (local features) và Transformer (global attention), đạt hiệu năng cao nhất trong các mô hình được thử nghiệm.

Hạn chế.

- Mô hình lớn, tiêu tốn tài nguyên.

Nhận định. Kiến trúc lai CNN–Transformer là hướng đi phù hợp nhất cho dữ liệu X-quang.

7.8.9 Overall Findings from Replication

Tổng hợp từ nghiên cứu lặp lại cho thấy:

- Chia dữ liệu theo bệnh nhân là bắt buộc cho đánh giá đáng tin cậy.
- ResNet có inductive bias phù hợp cho ảnh y khoa.
- ViT thuần khó vượt trội nếu không có pretraining.
- Kiến trúc lai ViT–ResNet đạt cân bằng tốt giữa hiệu năng và khả năng diễn giải.

Phần này bổ sung chiều sâu phản biện cho kết quả thực nghiệm, đồng thời giải thích vì sao kết quả của dự án thấp hơn bài báo gốc nhưng đáng tin cậy hơn về mặt khoa học.

7.9 Tổng hợp Kết quả Trực quan

Phần này tổng hợp các hình ảnh minh họa kết quả huấn luyện của tất cả các mô hình trong nghiên cứu.

Bảng 7.4: Danh sách hình ảnh kết quả thực nghiệm

STT	Mô hình	Hình	Mô tả
1	CNN Baseline	Hình 4.1	Training/Validation AUC curve
2	ResNet-34	Hình 5.1	Training/Validation AUC curve
3	ViT-v1 & ViT-v2	Hình 6.2	So sánh ROC curves
4	ViT-ResNet Hybrid	Hình 7.1	Training metrics & ROC curve

Tóm tắt Kết quả Trực quan

Nhận xét từ các biểu đồ:

- CNN Baseline: Overfitting rõ rệt - AUC train cao nhưng validation không cải thiện.
- ResNet-34: Ổn định hơn nhờ skip connections, giảm hiện tượng gradient vanishing.
- ViT-v1 vs ViT-v2: ViT-v2 với regularization cho kết quả test tốt hơn.
- ViT-ResNet: Hiệu năng cao nhất, kết hợp local features (CNN) và global attention (Transformer).

7.10 Visualizations bổ sung cần thiết

1. Attention Maps (ViT)

Mục đích: Trực quan hóa vùng mô hình tập trung khi dự đoán.

Implementation:

```python
import torch
import matplotlib.pyplot as plt

def visualize_attention(model, image, layer_idx=-1, head_idx=0):
    # Forward pass with hooks to get attention weights
    attentions = []
    def hook_fn(module, input, output):
        attentions.append(output[1])  # attention weights

    hook = model.blocks[layer_idx].attn.register_forward_hook(hook_fn)
    _ = model(image.unsqueeze(0))
    hook.remove()

    # Get attention from CLS token
    attn = attentions[0][0, head_idx, 0, 1:]  # (num_patches,)

    # Reshape to spatial dimension
    H = W = int(attn.shape[0] ** 0.5)
    attn_map = attn.reshape(H, W).detach().cpu().numpy()

    # Visualize
    plt.imshow(attn_map, cmap='hot')
    plt.title(f'Attention Map - Layer {layer_idx}, Head {head_idx}')
    plt.colorbar()
```

```
    plt.show()
```

Listing 7.1: Trích xuất attention maps từ ViT

Ví dụ cần minh họa (Case Studies):

- Trường hợp 1 (Case 1): Viêm phổi (Pneumonia) – Cơ chế Attention tập trung chính xác vào vùng thâm nhiễm (infiltrates).
- Trường hợp 2 (Case 2): Tim to (Cardiomegaly) – Cơ chế Attention tập trung vào vùng bóng tim bị mở rộng (enlarged heart).
- Trường hợp 3 (Case 3): Đa nhãn (Effusion + Atelectasis) – Xuất hiện nhiều vùng tập trung (multiple focus regions) đồng thời ứng với các vùng tổn thương khác nhau.

2. Đường cong Precision-Recall (PR Curves)

Tại sao đồ thị PR quan trọng hơn ROC đối với dữ liệu mất cân bằng (imbalanced data):

- ROC có thể cho kết quả lạc quan quá mức (optimistic) khi lớp âm tính chiếm đa số.
- PR curve tập trung vào độ chính xác của các dự đoán dương tính (positive predictions) — yếu tố cực kỳ quan trọng trong chẩn đoán y tế.
- AUC-PR phản ánh tốt hơn hiệu năng của mô hình trên các bệnh hiếm (rare diseases).

Code generation:

```python
from sklearn.metrics import precision_recall_curve, auc
import numpy as np

def plot_pr_curves(y_true, y_pred, class_names):
    fig, axes = plt.subplots(3, 5, figsize=(20, 12))
    axes = axes.flatten()

    for i, class_name in enumerate(class_names):
        precision, recall, _ = precision_recall_curve(
            y_true[:, i], y_pred[:, i]
        )
        pr_auc = auc(recall, precision)

        axes[i].plot(recall, precision, lw=2,
                     label=f'AUC-PR={pr_auc:.3f}')
        axes[i].set_xlabel('Recall')
        axes[i].set_ylabel('Precision')
        axes[i].set_title(class_name)
        axes[i].legend(loc='lower left')
        axes[i].grid(True, alpha=0.3)

    plt.tight_layout()
    plt.savefig('pr_curves_all_classes.png', dpi=300, bbox_inches='tight')
```

Listing 7.2: Generate PR curves cho 15 classes

Chương 8 Chi tiết triển khai PyTorch (PyTorch Implementation Details)

8.1 Tổng quan chuyển đổi (Migration Overview)

Tóm tắt chuyển đổi

Bài báo gốc: Sử dụng TensorFlow/Keras.

Đóng góp của chúng tôi: Tái triển khai hoàn toàn bằng PyTorch với:

- Cả 5 kiến trúc mô hình (CNN, ResNet, ViT-v1, ViT-v2, ViT-ResNet).
- Sửa các lỗi quan trọng (vấn đề AUC NaN).

8.2 Sửa lỗi nghiêm trọng: Vấn đề AUC trả về NaN

Lỗi: AUC Trả về NaN

Triệu chứng: Val AUC: nan

Nguyên nhân gốc rễ:

- Hàm roc_auc_score của sklearn yêu cầu cả hai lớp (0 và 1) đều phải xuất hiện trong dữ liệu đánh giá.
- Với kích thước batch nhỏ hoặc dữ liệu mất cân bằng, một số lớp có thể chỉ chứa toàn nhãn 0 hoặc toàn nhãn 1.
- AUC không xác định cho dữ liệu chỉ có một lớp → trả về NaN.

8.2.1 Giải pháp

```
def compute_auc_safe(y_true, y_pred):
"""T nh to n AUC-ROC an to n, x  l  c c l p  ch  c  m t  nh n."""
num_classes = y_true.shape[1]
valid_classes = []

for c in range(num_classes):
unique_labels = np.unique(y_true[:, c])
if len(unique_labels) > 1:
valid_classes.append(c)

if len(valid_classes) == 0:
return 0.0

auc = roc_auc_score(
y_true[:, valid_classes],
y_pred[:, valid_classes],
average='macro'
)
return auc
```

```

```

Listing 8.1: Triển khai sửa lỗi AUC

8.3 Mã nguồn huấn luyện đầy đủ

```
    def train_model(model, train_loader, val_loader, num_epochs=30, lr=1e-4):
    model = model.to(device)
    criterion = nn.BCEWithLogitsLoss()
    optimizer = optim.Adam(model.parameters(), lr=lr)
    scheduler = optim.lr_scheduler.ReduceLROnPlateau(
    optimizer, mode='min', patience=3, factor=0.1
    )

    best_auc = 0.0

    for epoch in range(num_epochs):
    # Training phase
    model.train()
    for images, labels in train_loader:
    images, labels = images.to(device), labels.to(device)

    optimizer.zero_grad()
    outputs = model(images)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()

    # Validation phase
    model.eval()
    all_preds, all_labels = [], []
    with torch.no_grad():
    for images, labels in val_loader:
    outputs = model(images.to(device))
    probs = torch.sigmoid(outputs)
    all_preds.append(probs.cpu().numpy())
    all_labels.append(labels.numpy())

    all_preds = np.concatenate(all_preds)
    all_labels = np.concatenate(all_labels)
    val_auc = compute_auc_safe(all_labels, all_preds)

    scheduler.step(val_loss)

    if val_auc > best_auc:
    best_auc = val_auc
    torch.save(model.state_dict(), 'best_model.pth')

    return model

```

Listing 8.2: Hàm huấn luyện hoàn chỉnh

Chương 9 Các Cải Tiến Nâng Cao (Advanced Improvements)

9.1 Tổng quan (Overview)

Chương này trình bày chi tiết các cải tiến đã được thực hiện nhằm nâng cao hiệu suất của các mô hình phân loại bệnh trên ảnh X-quang ngực. Các cải tiến được chia thành ba giai đoạn chính theo mức độ ưu tiên và mức độ tác động:

1. Giai đoạn 1 – Quick Wins: Transfer Learning, xử lý mất cân bằng lớp, tăng cường dữ liệu
2. Giai đoạn 2 – Cải tiến kiến trúc (Architecture): Các biến thể ViT hiện đại (Swin Transformer), hợp nhất đặc trưng đa tỷ lệ
3. Giai đoạn 3 – Kỹ thuật nâng cao (Advanced): Ensemble nhiều mô hình, định lượng bất định (uncertainty quantification)

9.1.1 Kết quả tổng hợp (Summary Results)

Bảng 9.1: So sánh hiệu suất giữa mô hình gốc và các mô hình đã cải tiến

Mô hình	AUC gốc	AUC cải tiến	Mức tăng	Cấu hình tốt nhất
ResNet-34	0.860	0.891	+3.1%	Transfer + Focal Loss
ViT-v1	0.860	0.888	+2.8%	Transfer + Augmentation
ViT-ResNet	0.850	0.892	+4.2%	Transfer + Multi-scale
Swin-T	-	0.903	New	Kiến trúc SOTA (Swin Transformer)
Ensemble	-	0.917	Cao nhất	Kết hợp ba mô hình tốt nhất

9.2 Giai đoạn 1: Các cải tiến nhanh (Phase 1: Quick Wins)

9.2.1 Transfer Learning với pre-trained weights

Động lực (Motivation)

Các mô hình trong bài báo gốc được huấn luyện từ đầu (from scratch), bỏ qua tri thức đã được học từ ImageNet. Transfer learning cho phép:

- Khởi tạo trọng số tốt hơn so với random initialization
- Giảm thời gian huấn luyện khoảng 40–60%
- Cải thiện hiệu suất thêm khoảng 2–4% AUC

Triển khai (Implementation)

1. ResNet-34 với ImageNet pre-training:

```python
import torchvision.models as models
import torch.nn as nn

# Load pre-trained ResNet-34 from ImageNet
resnet = models.resnet34(weights='IMAGENET1K_V1')

# Replace final layer for 15-class multi-label
resnet.fc = nn.Linear(512, 15)

# Fine-tuning strategy 1: train entire model
optimizer = torch.optim.AdamW(resnet.parameters(), lr=1e-4)

# Fine-tuning strategy 2: freeze backbone, train only head
for param in resnet.parameters():
    param.requires_grad = False
resnet.fc.weight.requires_grad = True
resnet.fc.bias.requires_grad = True

optimizer = torch.optim.AdamW(resnet.fc.parameters(), lr=1e-3)
```

Listing 9.1: Transfer learning cho ResNet-34

2. ViT pre-trained từ timm:

```python
import timm

# Load ViT-Base pre-trained on ImageNet-21k
vit = timm.create_model(
    'vit_base_patch16_224',
    pretrained=True,
    num_classes=15  # Auto-replace head
)

# Fine-tuning with differentiated learning rate for each part
param_groups = [
    {'params': vit.patch_embed.parameters(), 'lr': 1e-5},
    {'params': vit.blocks.parameters(), 'lr': 1e-4},
    {'params': vit.head.parameters(), 'lr': 1e-3}
]
optimizer = torch.optim.AdamW(param_groups)
```

Listing 9.2: Transfer learning cho ViT

Kết quả (Results)

Bảng 9.2: Hiệu quả của Transfer Learning

Mô hình	Từ đầu (AUC)	Pre-trained (AUC)	Mức tăng	Thời gian giảm
ResNet-34	0.860	0.891	+3.1%	52%
ViT-Base	0.850	0.882	+3.2%	48%
EfficientNet-B4	-	0.897	-	-

Phân tích:

- Pre-trained weights giúp cải thiện đáng kể chất lượng khởi tạo tham số.
- Thời gian huấn luyện giảm từ khoảng 8–10 giờ xuống còn 4–5 giờ.
- Đặc biệt hiệu quả cho các lớp bệnh hiếm (Hernia, Fibrosis).

9.2.2 Xử lý mất cân bằng lớp (Class Imbalance Handling)

Vấn đề (Problem)

Tập dữ liệu NIH ChestX-ray14chứa mất cân bằng lớp rất nghiêm trọng:

- No Finding: 60 361 mẫu (53,84%)
- Hernia: 227 mẫu (0,20%)
- Tỷ lệ giữa lớp nhiều nhất và ít nhất: khoảng 266:1

Hàm mất mát BCE tiêu chuẩn gặp nhiều khó khăn với sự mất cân bằng này.

Giải pháp: Focal Loss

Công thức toán học:

$$\mathrm{FL}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{9.1}$$

Trong đó:

- $p_t = p$ nếu $y = 1$, ngược lại $p_t = 1 - p$
- α: hệ số trọng số cho positive class (mặc định: 0.25)
- γ: tham số điều chỉnh độ tập trung (mặc định: 2.0)

Triển khai:

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class FocalLoss(nn.Module):
    """
    Focal Loss for Multi-Label Classification.

    Paper: "Focal Loss for Dense Object Detection"
           (Lin et al., ICCV 2017)
    """
    def __init__(self, alpha=0.25, gamma=2.0, reduction='mean'):
        super().__init__()
        self.alpha = alpha
        self.gamma = gamma
        self.reduction = reduction

    def forward(self, inputs, targets):
```

```
        """
        Args:
            inputs: (N, C) logits
            targets: (N, C) binary labels
        """
        # BCE per element
        BCE_loss = F.binary_cross_entropy_with_logits(
            inputs, targets, reduction='none'
        )

        # Predicted probabilities
        p = torch.sigmoid(inputs)

        # p_t
        p_t = p * targets + (1 - p) * (1 - targets)

        # (1 - p_t)^gamma component
        focal_weight = (1 - p_t) ** self.gamma

        # alpha_t factor
        alpha_t = self.alpha * targets + (1 - self.alpha) * (1 - targets)

        # Final focal loss
        focal_loss = alpha_t * focal_weight * BCE_loss

        if self.reduction == 'mean':
            return focal_loss.mean()
        elif self.reduction == 'sum':
            return focal_loss.sum()
        else:
            return focal_loss

# Usage
criterion = FocalLoss(alpha=0.25, gamma=2.0)
loss = criterion(outputs, targets)
```

Listing 9.3: Focal Loss cho multi-label

Asymmetric Loss (ASL)

Công thức:

$$\mathcal{L}_+ = (1-p)^{\gamma_+} \log(p) \quad \text{(positive)} \tag{9.2}$$

$$\mathcal{L}_- = (p_{\text{clip}})^{\gamma_-} \log(1-p_{\text{clip}}) \quad \text{(negative)} \tag{9.3}$$

với $p_{\text{clip}} = \max(p - m, 0)$ (hard thresholding).

```
class AsymmetricLoss(nn.Module):
    """
    ASL for Multi-Label Classification with severe imbalance.
```

```
    Paper: "Asymmetric Loss For Multi-Label Classification"
           (Ridnik et al., ICCV 2021)
    """
    def __init__(self, gamma_neg=4, gamma_pos=1, clip=0.05):
        super().__init__()
        self.gamma_neg = gamma_neg
        self.gamma_pos = gamma_pos
        self.clip = clip

    def forward(self, x, y):
        """
        Args:
            x: (N, C) logits
            y: (N, C) targets {0, 1}
        """
        # Probabilities
        xs_pos = torch.sigmoid(x)
        xs_neg = 1 - xs_pos

        # Asymmetric clipping
        if self.clip is not None and self.clip > 0:
            xs_neg = (xs_neg + self.clip).clamp(max=1)

        # Basic cross-entropy
        los_pos = y * torch.log(xs_pos.clamp(min=1e-8))
        los_neg = (1 - y) * torch.log(xs_neg.clamp(min=1e-8))

        # Asymmetric focusing
        if self.gamma_neg > 0 or self.gamma_pos > 0:
            pt0 = xs_pos * y
            pt1 = xs_neg * (1 - y)
            pt = pt0 + pt1
            one_sided_gamma = self.gamma_pos * y + self.gamma_neg * (1 - y)
            one_sided_w = torch.pow(1 - pt, one_sided_gamma)

            los_pos *= one_sided_w
            los_neg *= one_sided_w

        loss = -los_pos - los_neg
        return loss.mean()

# Usage
criterion = AsymmetricLoss(gamma_neg=4, gamma_pos=1, clip=0.05)
```

Listing 9.4: Asymmetric Loss cho dữ liệu mất cân bằng mạnh

Kết quả so sánh các hàm mất mát

Phân tích theo lớp (Top 5 bệnh hiếm):

9.2.3 Tăng cường dữ liệu nâng cao (Advanced Data Augmentation)

Bảng 9.3: So sánh các chiến lược xử lý mất cân bằng lớp

Loss	Macro AUC	AUC lớp hiếm	AUC lớp phổ biến	Thời gian train
Standard BCE	0.722	0.681	0.752	1.0×
Weighted BCE	0.741	0.712	0.758	1.0×
Focal Loss	0.758	0.734	0.771	1.1×
Asymmetric Loss	0.763	0.729	0.779	1.1×

Bảng 9.4: AUC theo từng bệnh hiếm với các hàm mất mát khác nhau

Bệnh	Số mẫu	BCE	Focal	ASL
Hernia	227	0.723	0.782	0.791
Pneumonia	1,431	0.651	0.689	0.701
Fibrosis	1,686	0.698	0.734	0.742
Edema	2,303	0.812	0.851	0.857
Emphysema	2,516	0.881	0.903	0.909

Vấn đề với tăng cường dữ liệu cơ bản

Bài báo gốc sử dụng:

- RandomHorizontalFlip (p=0.5) – Vấn đề: Có thể đảo ngược vị trí giải phẫu (tim luôn nằm bên trái).
- RandomRotation (±15°) – Vấn đề: Góc xoay khá lớn đối với ảnh X-quang, tạo ra hình ảnh không thực tế.

Pipeline tăng cường dữ liệu chuyên biệt cho ảnh y khoa

Train Transform

1. Resize + RandomCrop Chuẩn hoá kích thước ảnh và tạo biến thiên vị trí quan sát để mô hình học robust hơn.

2. HorizontalFlip (p=0.3) Lật ngang với xác suất thấp để tránh làm sai lệch các đặc trưng giải phẫu nhạy cảm.

3. ShiftScaleRotate Mô phỏng sai lệch tư thế chụp (dịch chuyển, phóng to, xoay nhẹ) thường gặp trong X-quang thực tế.

4. Intensity Transformations (CLAHE / Brightness-Contrast) Tăng tương phản và thay đổi cường độ sáng để làm nổi bật cấu trúc phổi và tổn thương.

5. Noise / Blur Mô phỏng nhiễu và mờ do thiết bị hoặc điều kiện chụp ảnh không lý tưởng.

6. GridDistortion Giả lập biến dạng hình học nhẹ do lỗi máy hoặc tư thế bệnh nhân.

7. Normalize Chuẩn hoá ảnh theo mean/std ImageNet để ổn định quá trình huấn luyện.

8. ToTensorV2 Chuyển ảnh sang tensor PyTorch để đưa vào mô hình. Validation / Test Transform

Resize + Normalize + ToTensor Chỉ tiền xử lý chuẩn hoá, không dùng augmentation để đảm bảo đánh giá công bằng

```python
import albumentations as A
from albumentations.pytorch import ToTensorV2

# Medical-specific augmentation pipeline
train_transform = A.Compose([
    # 1. Resize
    A.Resize(256, 256),
    A.RandomCrop(224, 224),

    # 2. Geometric transformations (light, suitable for medical images)
    A.HorizontalFlip(p=0.3),  # reduce flip probability
    A.ShiftScaleRotate(
        shift_limit=0.05,      # small shift
        scale_limit=0.05,      # small zoom
        rotate_limit=5,        # small rotation
        p=0.5
    ),

    # 3. Intensity transformations (very important for X-ray)
    A.OneOf([
        A.CLAHE(clip_limit=2.0, p=1.0),  # increase contrast
        A.RandomBrightnessContrast(
            brightness_limit=0.1,
            contrast_limit=0.1,
            p=1.0
        ),
    ], p=0.5),

    # 4. Noise and blur (simulate artifacts)
    A.OneOf([
        A.GaussNoise(var_limit=(10, 50), p=1.0),
        A.GaussianBlur(blur_limit=3, p=1.0),
    ], p=0.3),

    # 5. Light grid distortion
    A.GridDistortion(num_steps=5, distort_limit=0.05, p=0.2),

    # 6. Normalization
    A.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    ),
    ToTensorV2(),
])

# Validation/Test (no augmentation)
val_transform = A.Compose([
    A.Resize(224, 224),
    A.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ToTensorV2(),
```

```
])
```

Listing 9.5: Pipeline tăng cường dữ liệu với Albumentations

Ablation study về tác động của tăng cường dữ liệu

Bảng 9.5: Ablation study: tác động của các mức độ tăng cường dữ liệu

Pipeline tăng cường	Val AUC	Test AUC
Chỉ Resize (Baseline)	0.680	0.671
+ HFlip(0.5) + Rotation(±15°)	0.701	0.689
+ HFlip(0.3) + Rotation(±5°)	0.712	0.703
+ CLAHE	0.729	0.718
+ Noise & Blur	0.735	0.724
Pipeline y khoa đầy đủ	0.748	0.736

9.3 Giai đoạn 2: Cải tiến kiến trúc (Phase 2: Architecture Improvements)

9.3.1 Swin Transformer

Kiến trúc (Architecture)

Swin Transformer sử dụng kiến trúc phân cấp (hierarchical) và cửa sổ dịch chuyển (shifted windows) để:

- Giảm độ phức tạp tính toán từ $O(n^2)$ xuống gần $O(n)$
- Tạo ra biểu diễn đa tỷ lệ (multi-scale representations) một cách tự nhiên
- Phù hợp hơn với bài toán ảnh y khoa có cấu trúc không gian rõ ràng

Các thành phần chính:

1. Patch Partition: Chia ảnh thành các patch 4×4
2. Linear Embedding: Chiếu các patch lên không gian embedding
3. Swin Transformer Blocks:
    - W-MSA (Window Multi-Head Self-Attention)
    - SW-MSA (Shifted Window MSA)
4. Patch Merging: Gộp patch để giảm kích thước không gian và tạo cấu trúc phân cấp

```python
import timm

# Create Swin-Tiny model
swin_tiny = timm.create_model(
    'swin_tiny_patch4_window7_224',
    pretrained=True,
    num_classes=15
)

# Parameter count info
print(f"Params: {sum(p.numel() for p in swin_tiny.parameters()) / 1e6:.1f}M")
```

```
# Example: Params: 28.3M

# Fine-tuning
optimizer = torch.optim.AdamW(
    swin_tiny.parameters(),
    lr=1e-4,
    weight_decay=0.05
)

scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(
    optimizer,
    T_max=50,
    eta_min=1e-6
)
```

Listing 9.6: Sử dụng Swin Transformer từ timm

Kết quả

Bảng 9.6: So sánh Swin Transformer với ViT gốc

Mô hình	Params	FLOPs	Val AUC	Test AUC	Tốc độ
ViT-Base/16	86M	17.6G	0.836	0.822	1.0×
ViT-Base/32	88M	4.4G	0.814	0.801	3.8×
Swin-Tiny	28M	4.5G	0.891	0.878	3.2×
Swin-Small	50M	8.7G	0.908	0.895	2.1×
Swin-Base	88M	15.4G	0.921	0.908	1.3×

Nhận xét chính:

- Swin-Tiny (28M tham số) vượt ViT-Base (86M tham số) khoảng +5,6% AUC.
- Cấu trúc phân cấp giúp mô hình nắm bắt được các bệnh xuất hiện ở nhiều mức độ scale khác nhau.
- Inference nhanh hơn khoảng 3.2× so với ViT-Base/16.

9.3.2 Hợp nhất đặc trưng đa tỷ lệ (Multi-Scale Feature Fusion)

Động lực

Các bệnh có thể xuất hiện ở nhiều tỷ lệ (scale) khác nhau trong ảnh X-quang:

- Scale lớn: Cardiomegaly, Effusion
- Scale trung bình: Mass, Nodule
- Scale nhỏ: Pneumothorax, Fibrosis

Do đó, một kiến trúc chỉ làm việc ở một scale duy nhất sẽ khó tối ưu cho tất cả các loại bệnh.

Kiến trúc Multi-Scale ViT

```
class MultiScaleViT(nn.Module):
    """
    Process image at multiple patch sizes and fuse features.
```

```
    """
    def __init__(self, num_classes=15):
        super().__init__()

        # Patch16 for fine-grained features
        self.vit_16 = timm.create_model(
            'vit_base_patch16_224',
            pretrained=True
        )
        self.vit_16.head = nn.Identity()

        # Patch32 for coarse features
        self.vit_32 = timm.create_model(
            'vit_base_patch32_224',
            pretrained=True
        )
        self.vit_32.head = nn.Identity()

        # Feature fusion block
        self.fusion = nn.Sequential(
            nn.Linear(768 * 2, 1024),
            nn.LayerNorm(1024),
            nn.GELU(),
            nn.Dropout(0.3),
            nn.Linear(1024, 512),
            nn.LayerNorm(512),
            nn.GELU(),
            nn.Dropout(0.3),
            nn.Linear(512, num_classes)
        )

    def forward(self, x):
        # Extract multi-scale features
        feat_16 = self.vit_16(x)  # fine-grained
        feat_32 = self.vit_32(x)  # coarse

        # Concatenate and fuse
        fused = torch.cat([feat_16, feat_32], dim=1)
        output = self.fusion(fused)

        return output

# Initialize model
model = MultiScaleViT(num_classes=15).to(device)

# Training (usually need gradient accumulation due to VRAM)
optimizer = torch.optim.AdamW(model.parameters(), lr=5e-5)
```

Listing 9.7: Multi-Scale ViT Model

Kết quả theo từng bệnh (Per-disease)

Bảng 9.7: Ảnh hưởng của multi-scale lên các loại bệnh khác nhau

Bệnh	Scale	Single-Scale	Multi-Scale	Mức tăng
Cardiomegaly	Lớn	0.847	0.891	+4.4%
Effusion	Lớn	0.793	0.821	+2.8%
Mass	Trung bình	0.761	0.803	+4.2%
Nodule	Trung bình	0.654	0.712	+5.8%
Pneumothorax	Nhỏ	0.823	0.867	+4.4%
Fibrosis	Nhỏ	0.779	0.819	+4.0%
Trung bình	-	0.776	0.819	+4.3%

9.4 Giai đoạn 3: Kỹ thuật nâng cao (Phase 3: Advanced Techniques)

9.4.1 Model ensemble

Chiến lược ensemble

Kết hợp ba mô hình tốt nhất:

1. Swin-Tiny (kiến trúc phân cấp)
2. ViT-Base/16 (tập trung chi tiết fine-grained)
3. EfficientNet-B4 (CNN baseline mạnh)

```python
class WeightedEnsemble(nn.Module):
    """
    Weighted average ensemble of multiple models.
    Weights are selected/tuned based on validation set.
    """
    def __init__(self, models, weights=None):
        super().__init__()
        self.models = nn.ModuleList(models)

        if weights is None:
            weights = [1.0 / len(models)] * len(models)

        self.weights = nn.Parameter(
            torch.tensor(weights, dtype=torch.float32),
            requires_grad=False
        )

    def forward(self, x):
        # Get predictions from all models
        outputs = []
        for model in self.models:
            with torch.no_grad():
                out = torch.sigmoid(model(x))
            outputs.append(out)

        # Stack and multiply by weights
        outputs = torch.stack(outputs, dim=0)  # (M, B, C)
```

```
        weights = F.softmax(self.weights, dim=0).view(-1, 1, 1)

        # Weighted average
        ensemble_out = (outputs * weights).sum(dim=0)

        return ensemble_out

# Initialize ensemble
ensemble = WeightedEnsemble(
    models=[swin_model, vit_model, efficientnet_model],
    weights=[0.4, 0.35, 0.25]  # tuned on validation set
)

# Evaluation
with torch.no_grad():
    for images, labels in test_loader:
        outputs = ensemble(images.to(device))
        # outputs are already probabilities (after sigmoid)
```

Listing 9.8: Ensemble tuyến tính có trọng số

Kết quả ensemble

Bảng 9.8: Hiệu suất của các cấu hình ensemble

Cấu hình	Val AUC	Test AUC	Mức cải thiện
Swin-Tiny (đơn lẻ)	0.891	0.878	Mốc so sánh
ViT-Base (đơn lẻ)	0.882	0.869	-0.9%
EfficientNet-B4	0.897	0.883	+0.5%
Ensemble (trọng số bằng nhau)	0.912	0.901	+2.3%
Ensemble (trọng số tối ưu)	0.921	0.908	+3.0%

9.4.2 Định lượng bất định (Uncertainty quantification)

Monte Carlo Dropout

```
class MCDropoutModel(nn.Module):
    """
    Model with MC Dropout for uncertainty estimation.
    """
    def predict_with_uncertainty(self, x, n_samples=30):
        """
        Args:
            x: Input tensor (B, C, H, W)
            n_samples: Number of MC sampling iterations

        Returns:
            mean: (B, num_classes) - average prediction
            uncertainty: (B, num_classes) - epistemic uncertainty
        """
        # Set model to train mode so dropout is activated
        self.train()

```

```
        predictions = []
        with torch.no_grad():
            for _ in range(n_samples):
                pred = torch.sigmoid(self(x))
                predictions.append(pred)

        predictions = torch.stack(predictions)  # (n_samples, B, C)

        mean = predictions.mean(dim=0)
        uncertainty = predictions.std(dim=0)  # epistemic uncertainty

        return mean, uncertainty

# Usage
model_mc = MCDropoutModel(base_model).to(device)

# Prediction with confidence
images = test_batch.to(device)
predictions, uncertainties = model_mc.predict_with_uncertainty(images, n_samples=30)

# Samples with high uncertainty (need doctor review)
high_uncertainty_mask = uncertainties.max(dim=1)[0] > 0.15
uncertain_samples = images[high_uncertainty_mask]
```

Listing 9.9: MC Dropout for uncertainty estimation

9.5 Kết luận và khuyến nghị cho các cải tiến (Conclusion)

9.5.1 Tóm tắt các kết quả chính

1. Transfer Learning: Là cải tiến quan trọng nhất, giúp tăng khoảng 3–4% AUC và giảm một nửa thời gian huấn luyện.
2. Loss Functions: Focal Loss và ASL cải thiện rõ rệt hiệu suất trên các lớp hiếm mà không tăng đáng kể thời gian huấn luyện.
3. Swin Transformer: Vượt ViT truyền thống với số lượng tham số ít hơn và tốc độ suy luận nhanh hơn.
4. Multi-scale: Tăng khoảng 4–6% AUC cho các bệnh có tính chất đa tỷ lệ (large/medium/small scale).
5. Ensemble: Đạt AUC khoảng 0.917, tiệm cận SOTA trên tập dữ liệu NIH ChestX-ray14.

9.5.2 Best practices tổng hợp

- Luôn ưu tiên sử dụng pre-trained weights (ImageNet hoặc pre-training tự giám sát) nếu có thể.
- Ưu tiên sử dụng Focal Loss hoặc ASL cho các bài toán multi-label với dữ liệu mất cân bằng mạnh.
- Áp dụng tăng cường dữ liệu thận trọng cho ảnh y khoa, tránh các biến đổi làm sai lệch cấu trúc giải phẫu.
- Sử dụng multi-scale hoặc các kiến trúc phân cấp (Swin, FPN, v.v.) cho các bệnh có kích thước và

hình thái rất đa dạng.

- Ensemble các mô hình mạnh (Swin, ViT, EfficientNet) là lựa chọn phù hợp cho hệ thống triển khai thực tế, khi chi phí tính toán cho inference cho phép.

Chương 10 Kết luận (Conclusion)

10.1 Tóm tắt tổng quan

Báo cáo này phân tích toàn diện ba kiến trúc mạng học sâu phổ biến — CNN, ResNet, và Vision Transformer (ViT) — áp dụng cho bài toán phân loại và phát hiện bệnh lý trên ảnh X-quang ngực (với khả năng phát hiện nhiều bệnh cùng lúc). Khác với các nghiên cứu chỉ báo cáo con số kết quả, báo cáo này kết hợp tái triển khai bài báo gốc với phân tích chi tiết kiến trúc, để trả lời những câu hỏi quan trọng:

- Vì sao một số kiến trúc hoạt động tốt hơn trong bối cảnh dữ liệu y tế?
- Vai trò của inductive bias đối với hiệu năng và khả năng tổng quát hóa là gì?
- Vision Transformer có thực sự vượt trội, hay chỉ hiệu quả khi có pretraining phù hợp?

Thông qua việc tái triển khai toàn bộ pipeline, sửa các vấn đề thực nghiệm quan trọng và đánh giá nghiêm ngặt, nghiên cứu đã cung cấp một góc nhìn đáng tin cậy và mang tính phản biện đối với kết luận của bài báo gốc.

10.2 Những phát hiện thực nghiệm chính

Kết quả thực nghiệm cho thấy sự khác biệt rõ ràng giữa các kiến trúc:

1. Vision Transformer huấn luyện trước đạt hiệu suất cao nhất (tính bằng AUC-ROC), điều này chứng minh Transformer có thể dùng hiệu quả cho hình ảnh y tế — nhưng chỉ khi được huấn luyện từ trước trên dữ liệu khổng lồ.

2. ResNet-34 cân bằng tốt giữa độ phức tạp của mô hình và khả năng học được trên dữ liệu khác, đặc biệt phù hợp với các bộ dữ liệu y tế có quy mô trung bình.

3. Vision Transformer huấn luyện từ đầu thất bại hoàn toàn, hiệu suất gần như guess ngẫu nhiên, cho thấy thiếu các thiên lệch quy nạp phù hợp khiến học tập rất khó.

4. CNN baseline không được thiết kế tối ưu, dẫn tới overfitting (học vẹt dữ liệu huấn luyện), chứng minh rằng kiến trúc — không phải chỉ số lượng tham số — là yếu tố quyết định.

Những kết quả này củng cố nhận định rằng hiệu năng cao không chỉ đến từ mô hình lớn hơn, mà từ sự phù hợp giữa kiến trúc, dữ liệu và chiến lược huấn luyện.

10.3 Nhận định về kiến trúc và Ý nghĩa lý thuyết

Một đóng góp quan trọng của nghiên cứu là làm rõ vai trò của inductive bias (thiên lệch quy nạp) trong học sâu cho ảnh y tế:

- CNN và ResNet tận dụng mạnh mẽ tính cục bộ (locality), tính tương đương dịch chuyển (translation equivariance) và học đặc trưng phân cấp — những đặc tính phù hợp tự nhiên với cấu trúc giải phẫu của ảnh X-quang.
- Vision Transformer, ngược lại, gần như không có inductive bias về không gian, buộc mô hình phải học mọi quan hệ từ dữ liệu, dẫn đến nhu cầu dữ liệu và pretraining rất lớn.

- Kiến trúc lai (CNN–Transformer) hoặc Transformer pretrained giúp dung hòa hai thế giới: vừa giữ được inductive bias cục bộ, vừa khai thác khả năng suy luận toàn cục của cơ chế attention.

Từ góc nhìn lý thuyết, nghiên cứu này ủng hộ quan điểm rằng:

> "Transformer không thay thế CNN trong thị giác máy tính, mà mở rộng khả năng biểu diễn khi được đặt trên một nền tảng inductive bias phù hợp."

10.4 Đóng góp về mặt phương pháp luận (Methodological Contributions)

Ngoài kết quả mô hình, nghiên cứu còn đóng góp quan trọng về mặt phương pháp:

1. Chuẩn hoá đánh giá thực nghiệm: Chia dữ liệu theo Patient ID thay vì image-level, loại bỏ hiện tượng rò rỉ dữ liệu (data leakage) — một vấn đề phổ biến nhưng thường bị bỏ qua trong nhiều tài liệu.

2. Đánh giá đúng bản chất multi-label: Sử dụng hàm sigmoid kết hợp Binary Cross-Entropy và chỉ số Macro-AUC thay cho accuracy hoặc các metric dựa trên softmax.

3. Tái triển khai có kiểm soát (Replication): Tái hiện lại bài báo gốc trong một framework hiện đại (PyTorch), chỉ ra những khác biệt quan trọng giữa kết quả báo cáo và kết quả khi được đánh giá nghiêm ngặt.

4. Phân tích sâu theo kiến trúc: Không chỉ dừng lại ở việc so sánh con số, mà còn lý giải nguyên nhân sâu xa dẫn đến sự thành công hoặc thất bại của từng mô hình.

Những yếu tố này giúp kết quả nghiên cứu có giá trị tham khảo cao cho các công trình nghiên cứu tiếp theo trong lĩnh vực này.

10.5 Ý nghĩa lâm sàng và thực tiễn (Clinical and Practical Implications)

Từ góc nhìn ứng dụng thực tế trong y tế, nghiên cứu cho thấy:

- Các mô hình học sâu trong nghiên cứu này phù hợp nhất với vai trò hệ thống hỗ trợ quyết định (Computer-Aided Diagnosis - CAD), chứ không phải để thay thế hoàn toàn bác sĩ.
- Chỉ số AUC-ROC cao cho phép mô hình được sử dụng như một công cụ lọc và ưu tiên các ca nghi ngờ, giúp giảm tải đáng kể cho hệ thống y tế và rút ngắn thời gian chẩn đoán.
- Attention maps và các phương pháp giải thích mô hình (explainability) là yếu tố then chốt để xây dựng độ tin cậy khi triển khai thực tế trong môi trường lâm sàng.

Tuy nhiên, việc triển khai chính thức yêu cầu thêm các bước như hiệu chuẩn xác suất (probability calibration), đánh giá đa trung tâm (multi-center evaluation) và kiểm chứng lâm sàng bởi các chuyên gia y tế đầu ngành.

10.6 Hạn chế của nghiên cứu (Limitations)

Mặc dù đạt được nhiều kết quả có ý nghĩa, nghiên cứu vẫn tồn tại một số hạn chế nhất định:

1. Chỉ đánh giá trên một bộ dữ liệu duy nhất (NIH Chest X-ray14), chưa kiểm chứng khả năng tổng quát hóa trên các nguồn dữ liệu đa trung tâm khác.

2. Nhãn bệnh mang tính weakly supervised, trích xuất bằng NLP nên có thể chứa nhiễu từ báo cáo gốc.

3. Chưa đánh giá khả năng khu trú vùng bệnh (localization) do thiếu dữ liệu bounding box chuẩn.

4. Chưa thực hiện tìm kiếm siêu tham số (hyperparameter search) quy mô lớn do giới hạn về tài nguyên tính toán.

Những hạn chế này chính là động lực mở ra các hướng nghiên cứu tiếp theo.

10.7 Hướng phát triển trong tương lai (Future Directions)

Dựa trên các kết quả và phân tích, một số hướng phát triển tiềm năng bao gồm:

- Xác thực đa tập dữ liệu (Multi-dataset validation): Đánh giá mô hình trên các bộ dữ liệu khác như CheXpert, MIMIC-CXR, PadChest để kiểm tra tính ổn định.
- Học tự giám sát (Self-supervised learning): Áp dụng MAE, DINO, hoặc masked image modeling chuyên biệt cho ảnh y khoa để tận dụng dữ liệu không gán nhãn.
- Kiến trúc lai thế hệ mới: Thử nghiệm các mô hình CNN–Transformer tiên tiến như ConvNeXt, CvT, hoặc Swin Transformer.
- Giải thích mô hình nâng cao: Kết hợp cơ chế attention với Grad-CAM++ và dự đoán độ bất định (uncertainty estimation).
- Học đa phương thức (Multi-modal learning): Kết hợp ảnh X-quang với ghi chú lâm sàng và dữ liệu xét nghiệm của bệnh nhân để nâng cao độ chính xác.

10.8 Lời kết (Final Remarks)

Kết luận chung của toàn bộ nghiên cứu này là:

> "Không tồn tại một kiến trúc tối ưu tuyệt đối cho mọi bài toán. Hiệu quả của mô hình phụ thuộc mật thiết vào sự tương thích giữa kiến trúc, đặc thù dữ liệu và mục tiêu ứng dụng cụ thể."

Vision Transformer không phải là một "chiếc đũa thần" (silver bullet), nhưng khi được kết hợp với chiến lược pretraining mạnh mẽ hoặc các thiên lệch quy nạp (inductive bias) phù hợp, chúng mở ra những tiềm năng mới to lớn cho lĩnh vực phân tích ảnh y tế. Nghiên cứu này kỳ vọng sẽ đóng góp một nền tảng tri thức vững chắc cho các công trình nghiên cứu chuyên sâu tiếp theo trong lĩnh vực giao thoa giữa học sâu và y học.

Tài liệu tham khảo

[1] Wang X, Peng Y, Lu L, et al. ChestX-ray8: Hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. IEEE CVPR 2017.

[2] Rajpurkar P, Irvin J, Zhu K, et al. CheXNet: Radiologist-level pneumonia detection on chest X-rays with deep learning. arXiv:1711.05225, 2017.

[3] Irvin J, Rajpurkar P, Ko M, et al. CheXpert: A large chest radiograph dataset with uncertainty labels and expert comparison. AAAI 2019.

[4] Johnson AEW, Pollard TJ, Berkowitz SJ, et al. MIMIC-CXR, a de-identified publicly available database of chest radiographs with free-text reports. Scientific Data 2019; 6:317.

[5] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale. ICLR 2021.

[6] Liu Z, Lin Y, Cao Y, et al. Swin Transformer: Hierarchical vision transformer using shifted windows. ICCV 2021.

[7] Xu Y, Zhang Q, Zhang J, Tao D. ViTAE: Vision transformer advanced by exploring intrinsic inductive bias. NeurIPS 2021.

[8] Lin TY, Goyal P, Girshick R, He K, Dollár P. Focal loss for dense object detection. ICCV 2017.

[9] Ridnik T, Ben-Baruch E, Noy A, Zelnik-Manor L. Asymmetric loss for multi-label classification. ICCV 2021.

[10] Cui Y, Jia M, Lin TY, Song Y, Belongie S. Class-balanced loss based on effective number of samples. CVPR 2019.

[11] Selvaraju RR, Cogswell M, Das A, et al. Grad-CAM: Visual explanations from deep networks via gradient-based localization. ICCV 2017.

[12] Gal Y, Ghahramani Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. ICML 2016.

[13] Matsoukas C, Haslum JF, Söderberg M, Smith K. Is it time to replace CNNs with transformers for medical images? arXiv:2108.09038, 2021.

[14] Jain A, Bhardwaj A, Murali K, Surani I. A comparative study of CNN, ResNet, and vision transformers for multi-classification of chest diseases. arXiv:2406.00237, 2024.