BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG

PHÂN HIỆU TRƯỜNG ĐẠI HỌC THỦY LỢI

BÁO CÁO MÔN HỌC KHAI PHÁ DỮ LIỆU

Chest X-Ray Images (Normal and Pneumonia)

SINH VIÊN THỰC HIỆN :

2251068263 – Mai Văn Tiền

2251068275 – Lê Huỳnh Cẩm Tú

GIÁO VIÊN HƯỚNG DẪN : Ths.Vũ Thị Hạnh

Hồ Chí Minh, ngày 26 tháng 10 năm 2025

MỤC LỤC

MỤC LỤC HÌNH ẢNH

1. GIỚI THIỆU ĐỀ TÀI

Hình ảnh X-quang là thứ đã quá quen thuộc với mọi người, đặc biệt là những người bác sĩ chuyên môn. Họ dựa vào X-quang để mà đưa ra chẩn đoán ra bệnh cho bệnh nhân, giúp cho nhiều bệnh nhân phát hiện và kịp thời chữa trị.

Vì tầm quan trọng ấy, việc chẩn đoán hình ảnh X-quang là kỹ năng bắt buộc phải có của các bác sĩ chuyên môn. Do đó việc có một công cụ giúp ích cho việc chẩn đoán của các bác sĩ là một việc hữu ích và tiện lợi.

Cho nên đề tài này sẽ nghiên cứu và phát triển ra một công cụ chẩn đoán hình ảnh X-quang dựa trên mô hình học AI, tập trung vào phát hiện sự bất thường của phổi thông qua hình ảnh X-quang lồng ngực.

AI sử dụng là mô hình học sâu (deep learning), học sâu là một nhánh của học máy. Chúng mô phỏng lại cách hoạt động não bộ của con người. Sức mạnh đột phá của nó so với các mô hình khác là vấn đề học sâu về dữ liệu, nhờ vào khả năng trích xuất ra những đặc trưng phức tạp, cùng với cách học mô phỏng cách hoạt động của người, mô hình có thể đưa ra phán đoán tựa như là một con người.

Nhờ vào đó mà chúng vượt qua những thách thức của những bài toán phức tạp như: nhận dạng chó mèo, phân loại hình ảnh và thậm chí còn tạo ra hình ảnh giả dựa vào dữ liệu thật ngoài ra còn có nhiều thử thách phức tạp khác. Giúp các lập trình viên thay vì lập trình một cách tường minh, tìm ra một quy luật phức tạp thì mô hình học thông qua việc huấn luyện dữ liệu có thể đưa ra kết quả rất hiệu suất.

Bằng cách sử dụng mô hình học sâu, đưa dữ liệu chứa các hình ảnh của X-quang được các bác sĩ có chuyên môn phân loại bình thường và viêm phổi. Đưa chúng vào mô hình để mà huấn luyện và đưa ra dự đoán cho những hình ảnh X-quang mới, có thể giúp ích cho việc chẩn đoán của các bác sĩ.

2. MỤC TIÊU VÀ BÀI TOÁN ĐẶT RA

Mục tiêu:

- Nghiên cứu nhiều mô hình học sâu giúp phát hiện viêm phổi từ đầu vào là ảnh x-quang ngực.
- Tìm hiểu về nguyên lý của x-quang, cấu tạo của phổi và viêm phổi và các cách nhận biết viêm phổi sơ khai từ hình ảnh x-quang.
- Tiền xử lý dữ liệu cho việc huấn luyện cho từng mô hình học sâu.
- Nghiên cứu các kỹ thuật cần thiết để huấn luyện mô hình, chẳng hạn như class_weight, transfer learning, training from scratch.
- Tìm hiểu về các kỹ thuật để tăng tốc độ hội tụ và cải thiện quá tình huấn luyện mô hình như label smoothing, chính quy hóa.
- Nghiên cứu các phương pháp đánh giá mô hình.
- Tìm hiểu về kỹ thuật GradCam++ để trích xuất tổn thương của phổi từ hình ảnh x-quang.

Bài toán đặt ra:

- Bài toán lớn: Sử dụng mô hình học sâu để phát hiện ra tình trạng phổi là bình thường hay là viêm phổi.
- Bài toán nhỏ: Khi mô hình dự đoán là viêm phổi thì có thể dự đoán xem tác nhân của nó.

3. MÔ TẢ DỮ LIỆU VÀ CÁC BƯỚC TIỀN XỬ LÝ

Bộ dữ liệu: Chest X-Ray Images (Normal and Pneumonia)

Đường dẫn liên kết: https://www.kaggle.com/datasets/ghost5612/chest-x-ray-images-normal-and-pneumonia

Mô tả dữ liệu:

- Bộ dữ liệu từ các nhóm bệnh nhi từ 1 đến 5 tuổi tại Trung tâm Y tế phụ nữ và trẻ em Quảng Châu.
- Dữ liệu là những hình ảnh x – quang ngực được phân loại viêm phổi với bình thường.
- Kích cỡ toàn bộ dữ liệu là 5,863 hình ảnh (JPEG)
- Viêm phổi chia ra 2 loại vi khuẩn và virus dựa vào tên của nó (bacteria, virus)
- Mục đích chính là phân loại người bệnh và người không bệnh, có thể phân loại người bệnh theo nguyên nhân là vi khuẩn hay virus gây ra.
- Dữ liệu đã được loại bỏ những hình ảnh kém chất lượng và không đọc được.
- Các hình ảnh đã được hai bác sĩ chuyên môn đánh giá trước và còn xác nhận lại bởi chuyên gia thứ ba.

Các bước tiền xử lý:

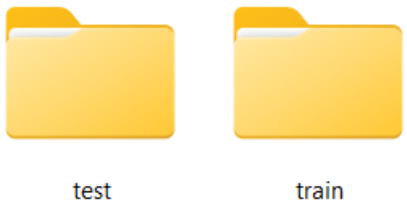
- Kích thước những hình ảnh không đồng bộ cần phải thống nhất về kích thước.
- Chia lại hình ảnh thành 3 nhãn: Bình thường, Viêm phổi virus, Viêm phổi vi khuẩn để có thể dự đoán được phổi có thể bị viêm và tác nhân của nó.
- Đưa hình ảnh về dạng chuẩn hóa cho từng mô hình mà mình lựa chọn.
- Sử dụng kỹ thuật tăng cường hình ảnh cho bộ train với các kỹ thuật: xoay, dịch trái phải, phóng to và lật để mô hình học đa dạng các khung ảnh.
- Hình ảnh khi chia thành 3 nhãn, số lượng của mỗi nhãn có sự chêch lệnh cần phải giúp cho mô hình ổn định các đối tượng. Kỹ thuật sẽ sử dụng class_weight, phương pháp này giúp cho mô hình giảm bớt chú ý lớp có số lượng lớn hơn lớp khác và ngược lại chú ý các lớp có số lượng thấp.

Hình ảnh code chia lại 2 lớp thành 3 lớp để có thể dự đoán tác nhân vi khuẩn hay virus.

```python
import os
import shutil
def makeThreeClass(path):
  # Thư mục gốc
  base_dir = path
  # Thư mục PNEUMONIA cũ
  pneu_dir = os.path.join(base_dir, "PNEUMONIA")

  # Thư mục mới cho từng loại
  virus_dir = os.path.join(base_dir, "PNEUMONIA_virus")
  bacteria_dir = os.path.join(base_dir, "PNEUMONIA_bacteria")

  # Tạo nếu chưa tồn tại
  os.makedirs(virus_dir, exist_ok=True)
  os.makedirs(bacteria_dir, exist_ok=True)

  # Duyệt qua tất cả file trong PNEUMONIA
  for filename in os.listdir(pneu_dir):
    src = os.path.join(pneu_dir, filename)

    # Kiểm tra file là virus hay vi khuẩn (bacteria)
    if "virus" in filename.lower():
        dst = os.path.join(virus_dir, filename)
        shutil.move(src, dst)
    elif "bacteria" in filename.lower():
        dst = os.path.join(bacteria_dir, filename)
        shutil.move(src, dst)
```

Hình 1. Hình ảnh code chia thành 3 lớp

Sau khi tạo thành thì gộp thư mục val vào train cho từng lớp. Tại vì số lượng ảnh val chưa đến 1% so với training.



Hình 2. Hình ảnh sau khi chia nhãn

Tổng kết dữ liệu ta được:

- Thư mục train : 5.263 bức ảnh
  - Bình thường : 1.365 (26%)
  - Vi khuẩn : 2.547 (48%)
  - Virus : 1.351 (26%)
- Thư mục test: 624 bức ảnh

Hình ảnh tiền xử lý và tăng cường hình ảnh

```
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    brightness_range=[0.9, 1.1],
    fill_mode='nearest',
    validation_split=0.1
    )
```

Hình 3. Tiền xử lý và tăng cường hình ảnh

Sử dụng tiền xử lý ảnh theo từng mô hình mình lựa chọn.

- Các chỉ số tăng cường ảnh:
- Xoay ảnh ngẫu nghiên -10 độ hoặc 10 độ.
- Dịch sang trái – phải, trên – dưới 10% theo lần lượt chiều rộng và chiều cao
- Phóng to hoặc thu nhỏ 10% hình ảnh.
- Thay đổi độ sáng ngẫu nhiên từ 0.9 đến 1.1.
- Điền giá trị pixel khi xoay/ dịch chuyển ảnh được lấp đầy bởi các pixel gần nhất để lấp đầy chỗ trống.

4. PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU VÀ MÔ HÌNH ML

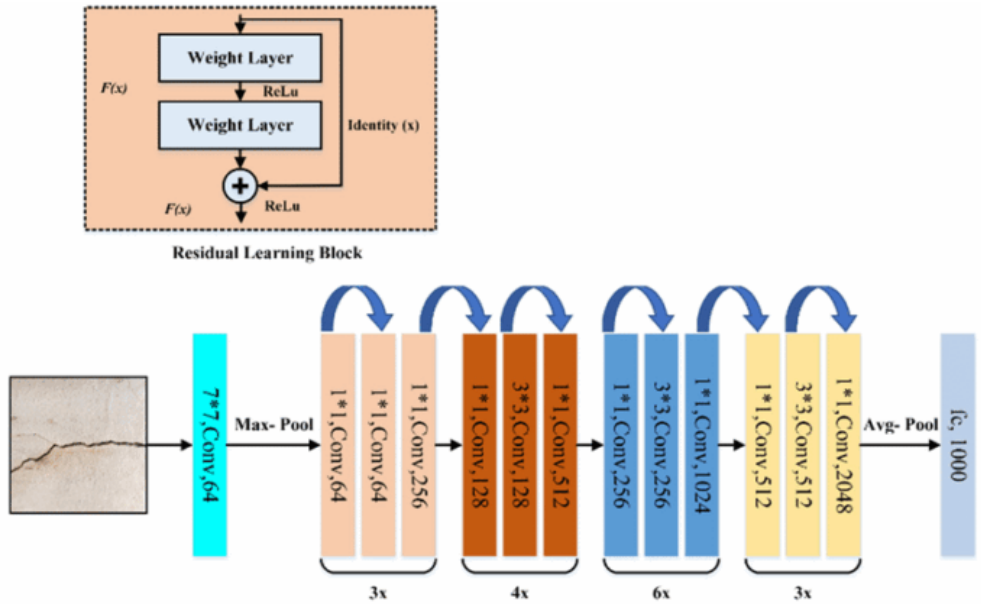
Mô hình được chọn:

- Resnet50
- DenseNet121
- EfficientNetB0

Vài nét về ba mô hình:

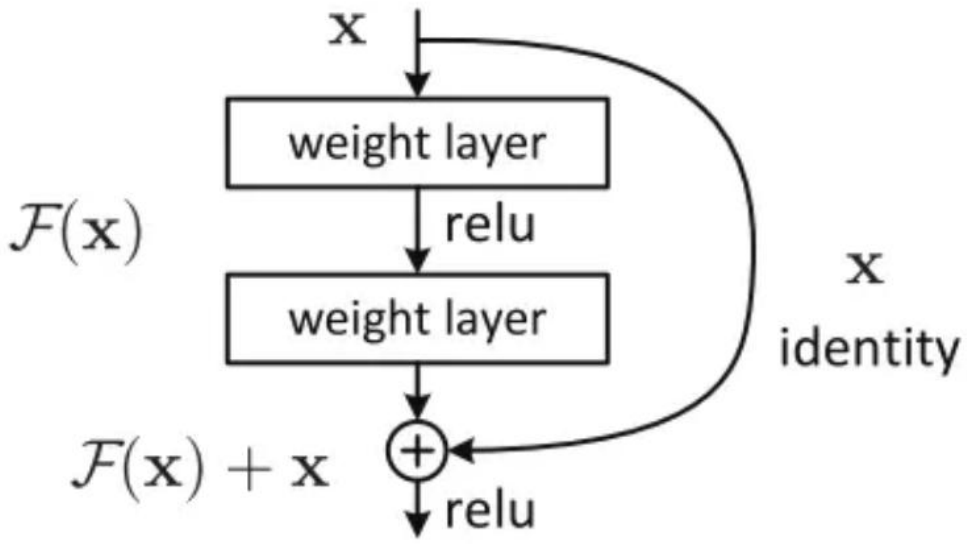
- Resnet50 – mô hình có đột phá lớn nhờ vào Residual Block giúp cho mô hình tránh được vấn đề vanishing gradient và exploding gradient.
- DenseNet121 – mô hình CheXNet phát triển bởi đại học Stanford kế thừa nó để mà huấn luyện 100.000 dữ liệu hình ảnh X-quang trên 14 bệnh lý, bài báo tập trung đánh giá về viêm phổi.
- EfficientNetB0 – mô hình giải quyết vấn đề Scalability, có độ chính xác cao hơn Resnet50 với bộ dữ liệu ImageNet (77.1%) cao hơn Resnet50 (76.0%), với hiệu suất như thế nhưng mô hình lại nhẹ hơn Resnet50 rất nhiều lần.

Kiến trúc mô hình Resnet50



Hình 4. Kiến trúc mô hình Resnet50

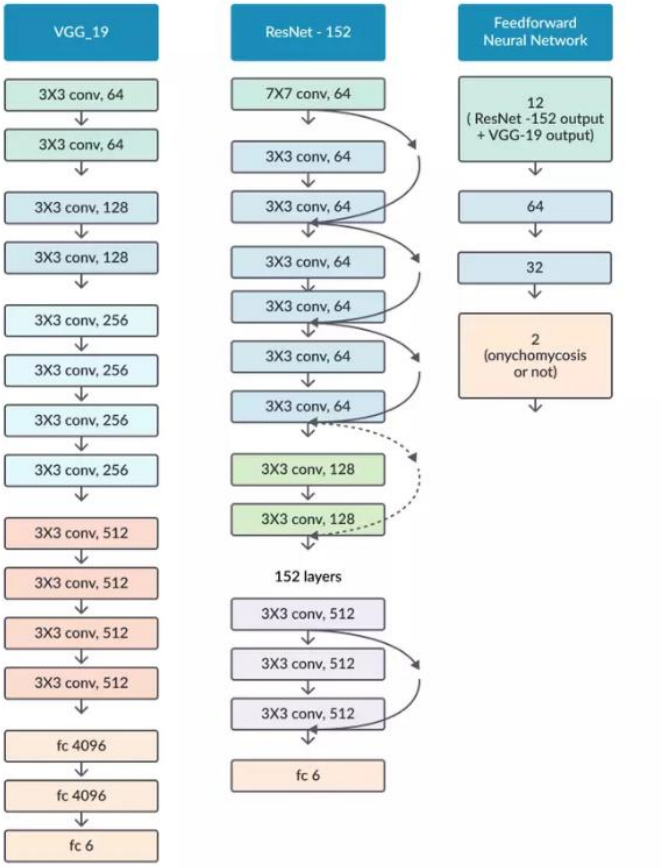
Nổi bật với cách giải quyết vấn đề mô hình mạng quá sâu có thể làm mất mát thông tin của dữ liệu và vấn đề vanishing gradient và exploding gradient khiến mô hình không học được gì từ dữ liệu cả.



Hình 5. Residual Block với Skip connection

Kỹ thuật Residual Block với Skip connection mạng sẽ học phần dư x thay vì mong đợi là H(x) sau đó ta sẽ thêm x vào cuối để làm sao cho H(x) và F(x) xấp xỉ nhau.

Cho nên với kỹ thuật học đó, mô hình Resnet50 thường có độ sâu rất nhiều so với các mô hình khác.

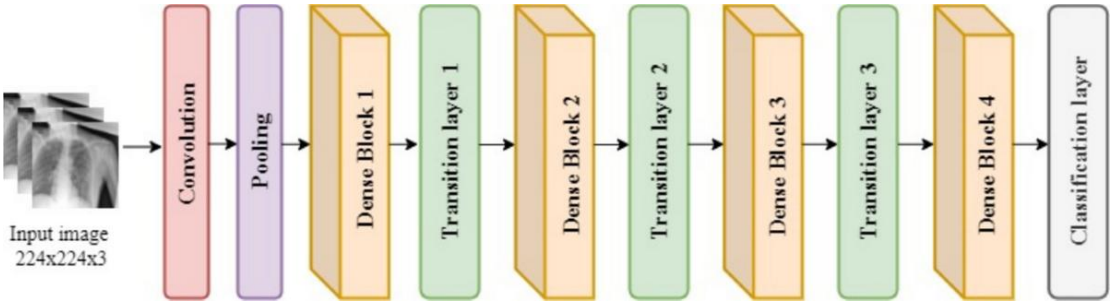


Hình 6. So sánh các mô hình khác với Resnet50

Thông số về Resnet50:

- Tổng tham số: khoảng 25 triệu tham số
- Tổng số layer: 50 layers
- Input size mặc định: 224x224x3
- Kích cỡ: 98MB

Kiến trúc mô hình DenseNet121

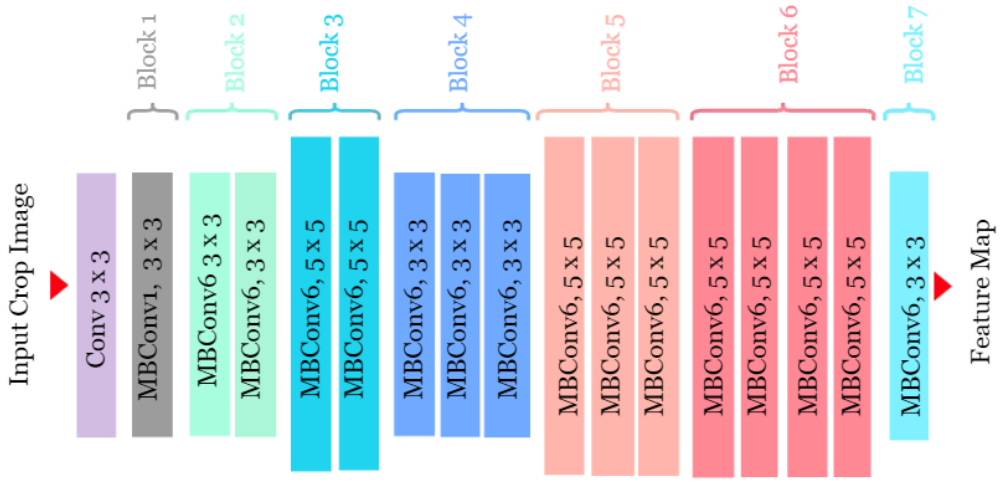


Hình 7. Kiến trúc mô hình DenseNet121

Thông số về DenseNet121:

- Tổng tham số: khoảng 7 triệu tham số
- Tổng số layer: 121 layers
- Input size mặc định: 224×224×3
- Kích cỡ: 28MB

Kiến trúc mô hình EfficientNetB0



Hình 8. Kiến trúc mô hình EfficientNetB0

Thông số về EfficientNetB0:

- Tổng tham số: khoảng 4 triệu tham số
- Tổng số layer: 82 layers
- Input size mặc định: 224x224x3
- Kích cỡ: 18MB

Kỹ Thuật Transfer Learning

Transfer Learning là học chuyển giao, dựa vào các mô hình đã được học sẵn trên bộ dữ liệu ImageNet để mà có thể tận dụng lại và điều chỉnh lại cho phù hợp với bài toán của mình.

```
Total params: 7,563,843 (28.85 MB)
Trainable params: 526,339 (2.01 MB)
Non-trainable params: 7,037,504 (26.85 MB)
```

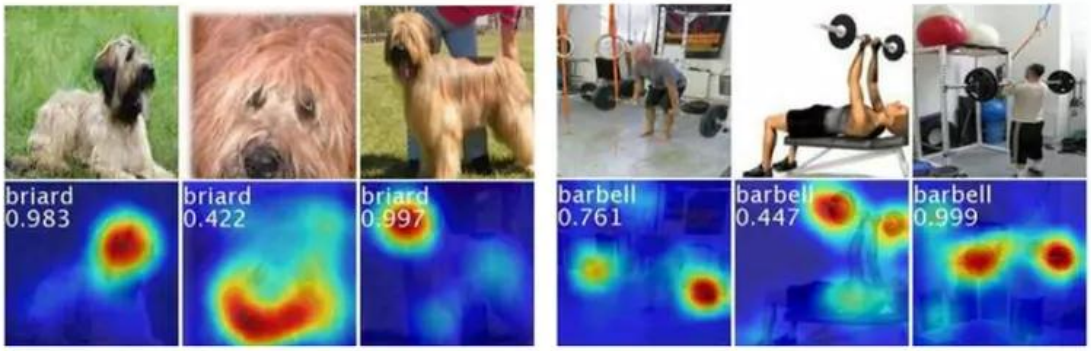
Hình 9. Kỹ thuật transfer learning

Có 2 loại học phổ biến:

- Feature Extraction: Giữ nguyên lại các tham số của các lớp CNN, chỉnh lại phần đầu ra.
- Fine-tuning: Mở khóa một vài layer cuối để huấn luyện lại.

Kỹ Thuật GRADCAM++

Grad-Cam++ là một kỹ thuật được sử dụng trong thị giác máy tính, giúp giải thích được mạng học sâu đang tập trung vào vị trí nào trên hình ảnh để mà mô hình đưa ra dự đoán.

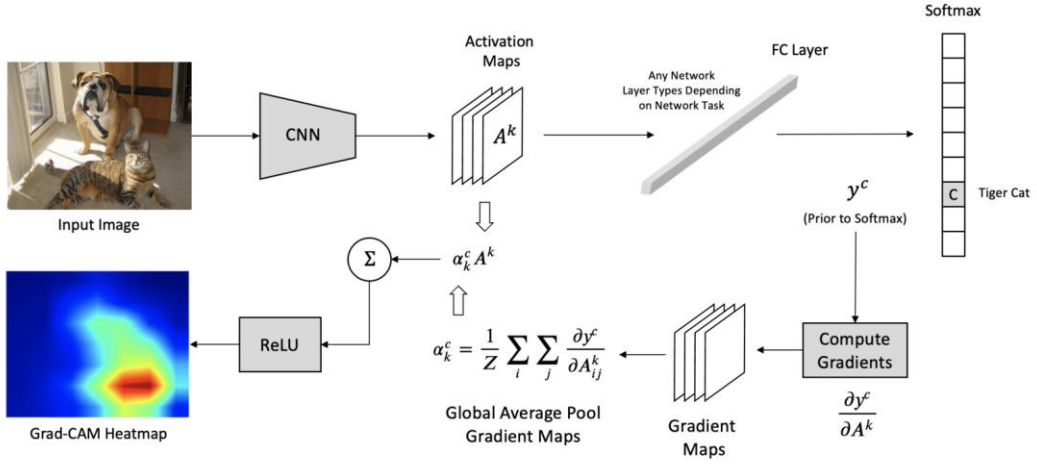


Hình 1. CAM cho class "briard" (trái) và class "barbell" (phải). Map hiển thị những vùng mang đặc trưng cao cho class đó, như đầu của con chó "briard" và 2 bên đĩa của "barbell"

Hình 10. Kỹ thuật CAM

Kỹ thuật này kế thừa từ Grad-Cam (Gradient-weight Class Activation Mapping) và cũng đồng kế thừa từ Cam.

Kỹ thuật này thường sử dụng lớp conv gần cuối bởi vì nó chứa nhiều đặc trưng của đối tượng nhất, sau đó chúng tính gradient cho hình ảnh nhờ vào class mà nó thuộc về rồi quay ngược trở lại lớp conv để mà trích xuất đặc trưng mà mô hình đưa ra dự đoán.



Hình 11. Quy trình GradCam

Nhờ vào kỹ thuật này có thể khai phá được bên trong mô hình học sâu đang học như thế nào và đặc điểm gì giúp mô hình có thể đưa ra dự đoán kết quả.

Điểm Chung Về Việc Huấn Luyện Mô Hình

Sử dụng chính quy hóa

Các mô hình đều sử dụng kỹ thuật chính quy hóa (regularization) để mà có thể giúp mô hình tránh hiện tượng quá khớp nhờ thêm vào loss một yếu tố.

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} W_j^2$$

Loss function

Regularization Term

Hình 12. Công thức chính quy hóa

Kỹ thuật Label Smoothing Loss Function

$$y = [1, 0, 0]$$

But with label smoothing (say, $\epsilon = 0.1$), this becomes:

$$y_{\text{smooth}} = [0.9, 0.05, 0.05]$$

In general, for any class i, the smoothed label is:

$$y_{\text{smooth},i} = (1 - \epsilon) \quad \text{for the true class}$$

$$y_{\text{smooth},j} = \frac{\epsilon}{k - 1} \quad \text{for all other classes}$$

Hình 13. Hình dung về label smoothing

Giúp mô hình học không cần khớp quá dữ liệu chỉ cần vẫn phù hợp với lớp mà nó thuộc về.

Sử dụng EarlyStopping

```
early_stop = EarlyStopping(
  monitor='val_loss',
  patience=5,
  restore_best_weights=True
)
```

Hình 14. Kỹ thuật Early Stopping

Kỹ thuật EarlyStopping: giúp quá trình huấn luyện ngừng lại nếu xem xét một chỉ số không đạt được yêu cầu thì ngắt quá trình huấn luyện lại.

Sử dụng ReduceLROnPlateau

```
reduce_lr = ReduceLROnPlateau(
  monitor='val_loss',
  factor=0.1,
  patience=3,
  min_lr=1e-7,
  verbose=1
)
```

Hình 15. Kỹ thuật ReduceLROnPlateau

Kỹ thuật ReduceLROnPlateau: cũng xem xét một chỉ số không đạt được yêu cầu, kỹ thuật này sẽ giảm learning rate xuống cho mô hình hội tụ tìm kiếm giải pháp tối ưu.

Sử dụng ModelCheckpoint

```
checkpoint = ModelCheckpoint(
  filepath=path,
  monitor='val_loss',
  save_best_only=True,
  save_weights_only=False,
  verbose=1
)
```

Hình 16. Kỹ thuật ModelCheckpoint

Kỹ thuật ModelCheckpoint: cũng xem xét một chỉ số nhưng mà đạt yêu cầu, thì kỹ thuật này sẽ lưu cho mình mô hình tốt nhất trong quá trình huấn luyện.

Transfer Learning với Resnet50

```
resnet = ResNet50( weights='imagenet',include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, IMG_CHANNEL))
resnet.trainable = False
x = layers.GlobalAveragePooling2D()(resnet.output)
x = Dense(512,kernel_regularizer=l2(0.01), bias_regularizer=l2(0.01))(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x= Dropout(0.2) (x)
output = Dense(3, activation='softmax')(x)
model_rn = Model(inputs=resnet.inputs, outputs=output)
model_rn.summary()
```

Hình 17. Transfer learning Resnet50

Cấu hình

- Tổng số tham số: 24,640,387
- Tham số huấn luyện: 1,051,651
- Chính quy hóa với: 1e-2
- Hàm tối ưu: Adam sử learning rate 1e-4
- Hàm loss: CategoricalCrossentropy với label_smothing 0.1
- Batch size: 32
- Epoch: 30
- Early: 3
- Reduce: 2

```
model_rn.compile(
    optimizer=tf.keras.optimizers.Adam(1e-4),
    loss=tf.keras.losses.CategoricalCrossentropy(label_smoothing=0.1),
    metrics=['accuracy']
)
```

```
BATCH_SIZE = 32
EPOCHS = 30
EARLY = 3
REDUCE = 2
hisfe_resnet50 = trainModel(model_rn,"resnet50.keras",resnet_preprocess)
```

Hình 18. Huấn luyện Resnet50

```
model_rn.compile(optimizer=tf.keras.optimizers.Adam(1e-5),
                 loss='categorical_crossentropy',
                 metrics=['accuracy'])
model_rn.fit(train_gen, epochs=5)
```

Hình 19. Fine tuning Resnet50

Transfer Learning với DenseNet121

```
densenet = DenseNet121(weights='imagenet', include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, IMG_CHANNEL))
fine= 50
for layer in densenet.layers[:-fine]:
    layer.trainable = False
for layer in densenet.layers[-fine:]:
    layer.trainable = True
x = layers.GlobalAveragePooling2D()(densenet.output)
x = Dense(256,kernel_regularizer=l2(5e-2))(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x= Dropout(0.5)(x)
output = Dense(3, activation='softmax')(x)

model_dense = Model(inputs=densenet.inputs, outputs=output)
model_dense.summary()
```

Hình 20. Transfer learning DenseNet121

Để giúp cho mô hình đạt hiệu suất cao, đã mở ra 50 lớp cuối để mà gộp hai giai đoạn feature extraction và fine tuning thành làm một.

Điều này giúp mô hình hội tụ đạt được độ chính xác cao có thể lên đến 92% nhưng với dữ liệu này mô hình có vẻ đang gặp vấn đề quá khớp, mặc dù độ chính xác cao nhưng hiệu suất không tốt.

Đây là nhật ký mà mình đã tinh chỉnh mô hình

EPOCH =30
== Feature + Finetuning(30 layer cuối), lr: 1e-4, batchsize:32, l2: 5e-3 , dropout: 0.2 : 0.92%
(mô hình có vẻ overfitting ở epoch: 18 tỉ lệ gap: 8%)
== Feature + Finetuning(30 layer cuối), lr: 1e-4, batchsize:32, l2: 1e-2 , dropout: 0.2 : 0.91%
(mô hình có vẻ overfitting ở epoch : 10)
== Feature + Finetuning(30 layer cuối), lr: 1e-4, batchsize:32, l2: 2e-2 , dropout: 0.2 : 0.92%
(mo hinh co ve overfitting 10)
== Feature + Finetuning(30 layer cuối), lr: 1e-4, batchsize:32, l2: 5e-2 , dropout: 0.2 : 0.92 %
(mô hình có vẻ học tốt hơn nhưng bị 1 lần overfit ở 17)
== Feature + Finetuning(30 layer cuối), lr: 5e-4, batchsize:32, l2: 1e-2 , dropout: 0.2 : 0.94%
(mô hình có vẻ học tốt hơn nhưng bị 1 lần overfit ở 5,10)
== Feature + Finetuning(30 layer cuối), lr: 5e-4, batchsize:32, l2: 1e-2 , dropout: 0.4 : 0.92%
(mô hình có vẻ học tốt hơn nhưng bị 1 lần overfit ở)
== Feature + Finetuning(20 layer cuối), lr: 1e-4, batchsize:32, l2: 1e-2 , dropout: 0.2 : 0.90%
(overfitting)
== Feature + Finetuning(40 layer cuối), lr: 1e-4, batchsize:32, l2: 1e-2 , dropout: 0.2 : 0.91%
(Học có vẻ ổn hơn chêch lệnh không quá nhiều)
== Feature + Finetuning(40 layer cuối), lr: 1e-4, batchsize:32, l2: 1e-2 , dropout: 0.5 : 0.91%
(ở epoch 24 là ổn: 87%, 87%)

Ở các tham số trên mặt dù tốt trong quá trình training nhưng mà nó khá tệ khi test và sự chêch lệch giữa việc xác thực và huấn luyện cao hơn 5% ngưỡng mà mình đặt ra là hiện tượng quá

khớp. Với lại khi test lại với các hình ảnh không huấn luyện thì độ chính xác còn giảm mạnh loanh quoanh [60% - 80%]

Cấu hình

- Tổng số tham số: 7,301,699
- Tham số huấn luyện: 1,339,331
- Chính quy hóa với: 5e-2
- Hàm tối ưu: Adam sử learning rate 1e-4
- Hàm loss: CategoricalCrossentropy với label_smothing 0.1
- Batch size: 32
- Epoch: 20
- Early: 5
- Reduce: 3
- Finetuning lấy 50 layers cuối.

```
model_dense.compile(
    optimizer=tf.keras.optimizers.Adam(1e-4),
    loss=tf.keras.losses.CategoricalCrossentropy(label_smoothing=0.1),
    metrics=['accuracy']
)
```

```
BATCH_SIZE = 32
EPOCHS = 20
EARLY = 5
REDUCE = 3
his_dense = trainModel(model_dense,"DenseNet121.keras",densenet_preprocess)
```

Hình 21. Huấn luyện mô hình DenseNet121

Training from scratch với EfficientNetB0

```
effnet = EfficientNetB0(weights=None, include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, IMG_CHANNEL))
for layer in effnet.layers:
  layer.trainable = True
x = layers.GlobalAveragePooling2D()(effnet.output)
x = Dense(160,kernel_regularizer=l2(1e-3))(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x= Dropout(0.2)(x)
output = Dense(3, activation='softmax')(x)

model_effnet = Model(inputs=effnet.inputs, outputs=output)
model_effnet.summary()
```

Hình 22. Trainning from scratch EfficientNetB0

Cấu hình

- Tổng số tham số: 4,255,654
- Tham số huấn luyện: 4,213,311
- Chính quy hóa với: 1e-3
- Hàm tối ưu: Adam sử learning rate 1e-3
- Hàm loss: CategoricalCrossentropy với label_smothing 0.1
- Batch size: 16
- Epoch: 50
- Early: 10
- Reduce: 5

```
model_effnet.compile(
    optimizer='adam',
    loss=tf.keras.losses.CategoricalCrossentropy(label_smoothing=0.1),
    metrics=['accuracy']
)
```

```
BATCH_SIZE = 16
EPOCHS = 50
EARLY = 10
REDUCE = 5
his_effnet = trainModel(model_effnet,"EfficientNetB0.keras",efficientnet_preprocess)
```

Hình 23. Huấn luyện mô hình EfficientNetB0

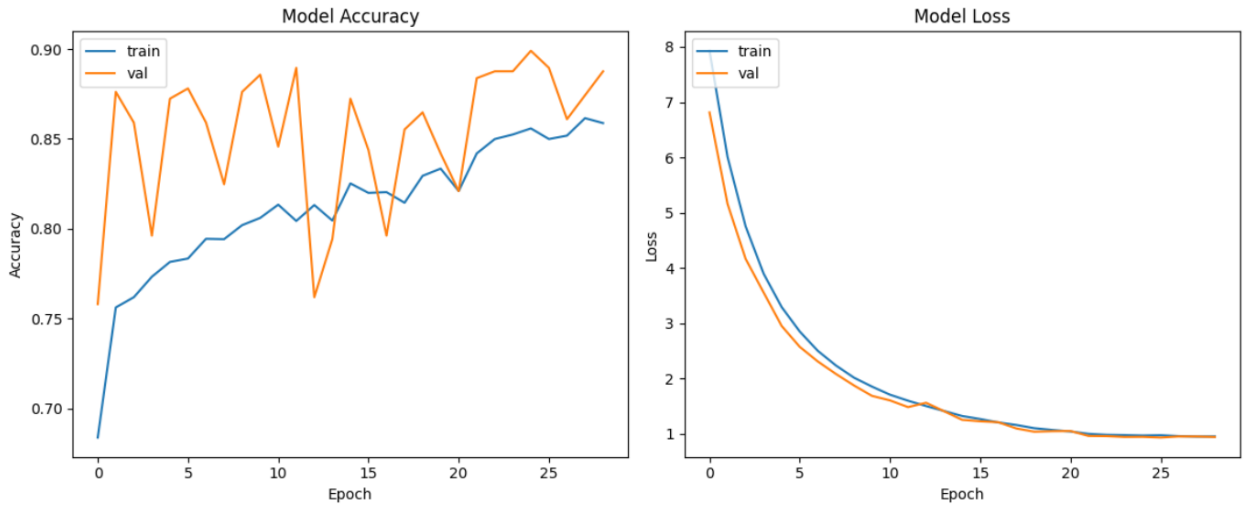
5. KẾT QUẢ VÀ ĐÁNH GIÁ MÔ HÌNH (ĐỘ CHÍNH XÁC, BIỂU ĐỒ, CONFUSION MATRIX, GIAO DIỆN TRÊN WEB)

Độ chính xác tập training

- Resnet50: 87%
- DenseNet121: 86%
- EfficientNetB0: ~82%

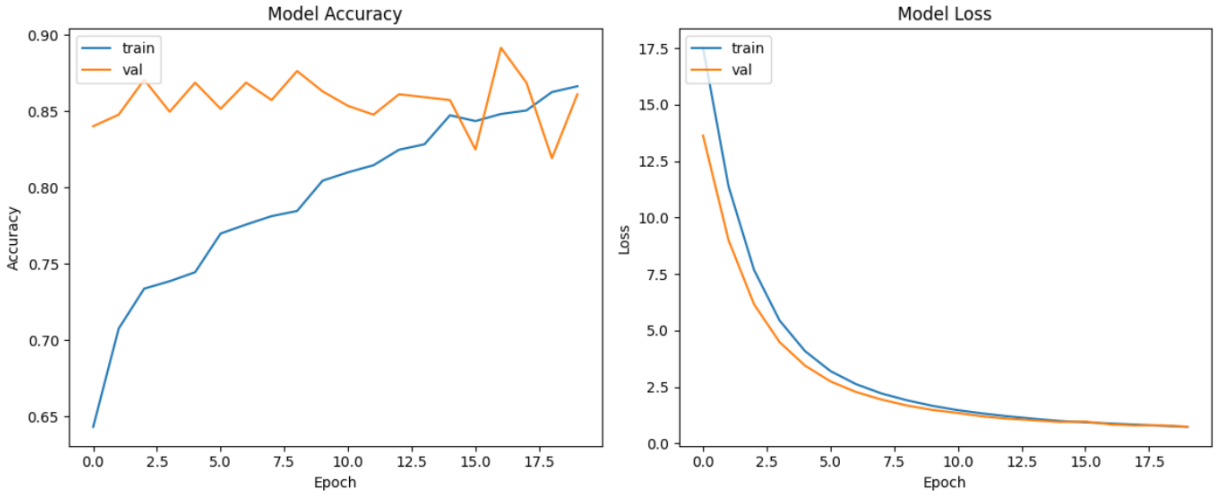
Dựa trên đồ thị học

Mô hình Resnet50



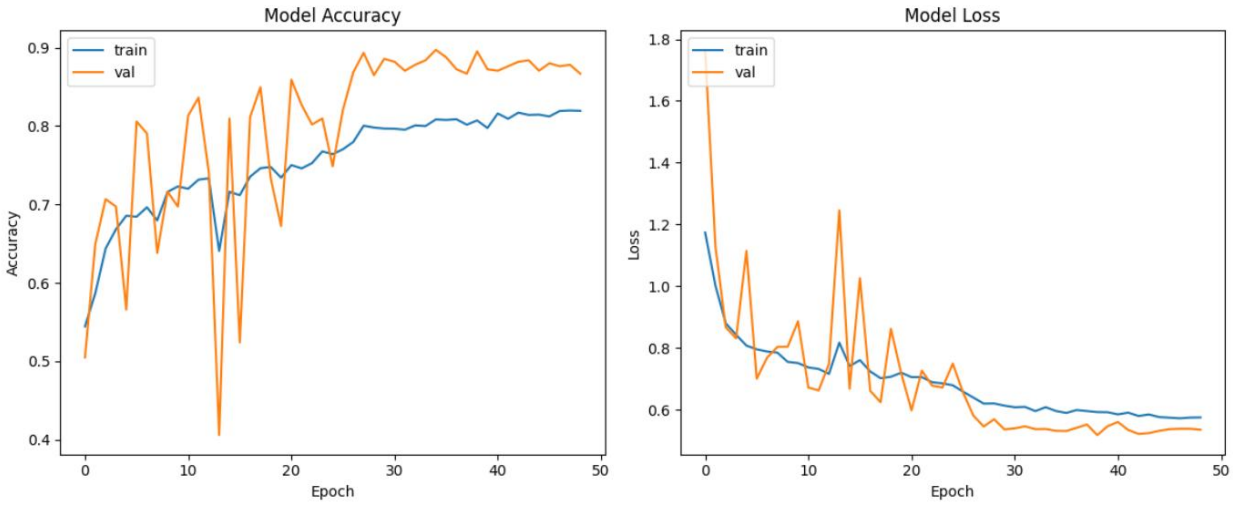
Hình 24. Đồ thị học Resnet50

Mô hình DenseNet121



Hình 25. Đồ thị học DenseNet121

Mô hình EfficientNetB0



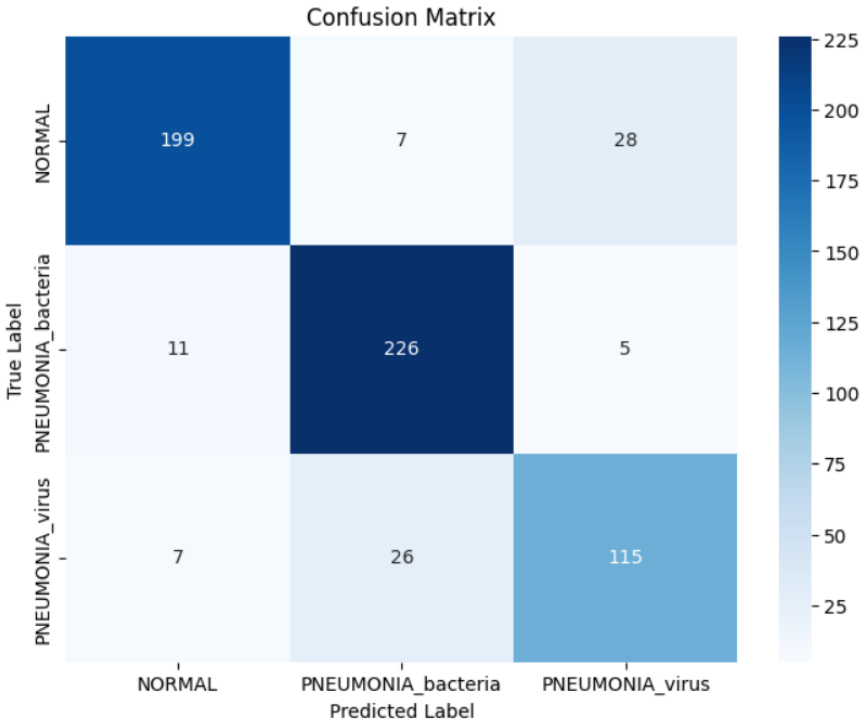
Hình 26. Đồ thị học EfficientNetB0

Nhận xét

- Dựa vào mô hình học dễ thấy Resnet50 và DenseNet121 học ổn định hơn và độ chính xác khá cao.
- Mô hình EfficientNetB0 học giao động nhiều nhưng về cuối tại epoch 30 trở lên, giá trị xác thực tốt hơn giá trị huấn luyện

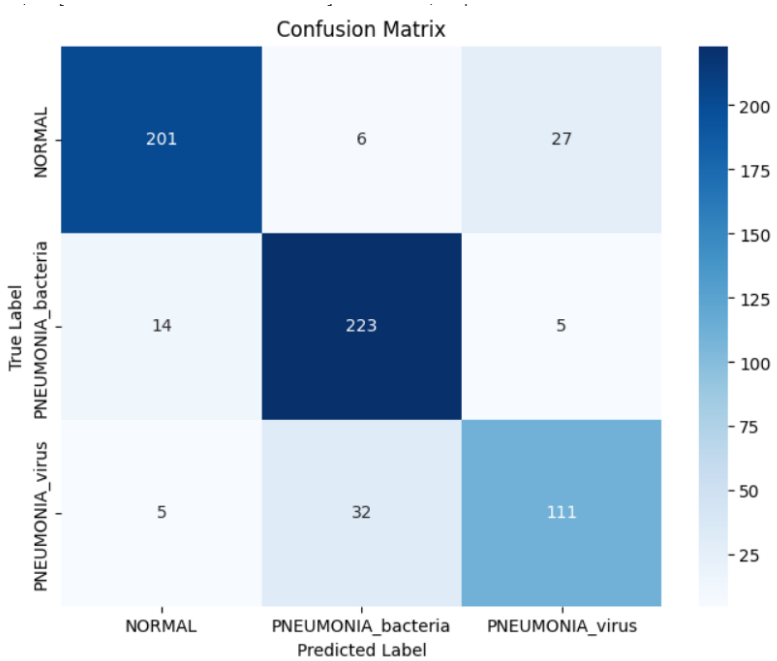
Dựa trên ma trận nhầm lẫn

Mô hình Resnet50



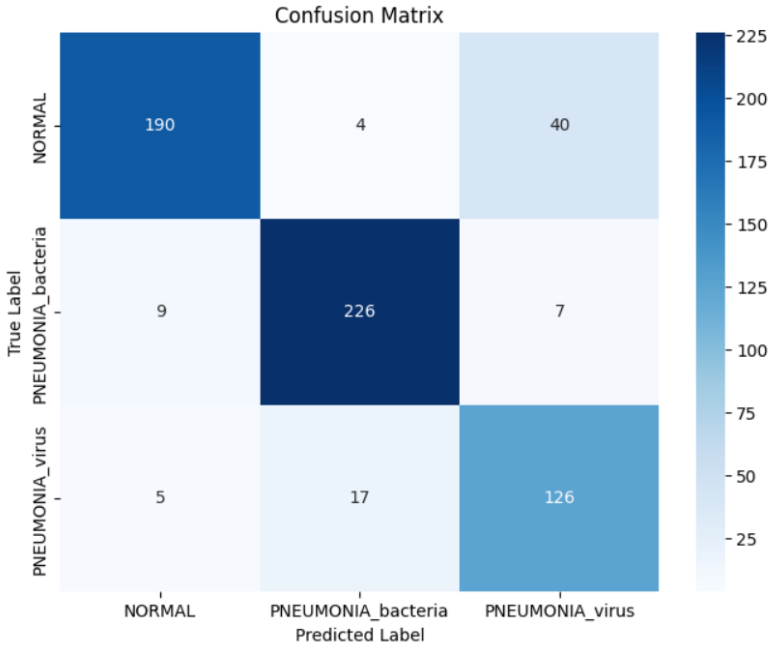
Hình 27. Ma trận nhầm lẫn Resnet50

Mô hình DenseNet121



Hình 28. Ma trận nhầm lẫn DenseNet121

Mô hình EfficientNetB0



Hình 29. Ma trận nhầm lẫn EfficientNetB0

Nhận xét:

- Dễ thấy ma trận nhầm lẫn của EfficientNetB0 đang làm tốt với mục đích phân loại giữa virus và vi khuẩn. Tuy nhiên độ nhầm lẫn giữa bình thường và viêm phổi đang cao hơn. Bài toán nhỏ được thỏa nhưng bài toán lớn không đạt so với hai mô hình còn lại.
- Còn lại hai mô hình ta dễ thấy Resnet50 phân loại vi khuẩn và virus tốt hơn DenseNet121.

Dựa trên các chỉ số

Mô hình Resnet50

- Độ chính xác (accuracy): 0.87
- Độ chính xác (precision)
  - Bình thường: 0.92
  - Viêm phổi vi khuẩn: 0.87
  - Viêm phổi virus: 0.78
- Khả năng thu hồi (recall)
  - Bình thường: 0.85
  - Viêm phổi vi khuẩn: 0.93
  - Viêm phổi virus: 0.78
- Điểm F1
  - Bình thường: 0.88
  - Viêm phổi vi khuẩn: 0.90

    - Viêm phổi virus: 0.78

Mô hình DenseNet121

- Độ chính xác (accuracy): 0.86
- Độ chính xác (precision)
    - Bình thường: 0.91
    - Viêm phổi vi khuẩn: 0.85
    - Viêm phổi virus: 0.78
- Khả năng thu hồi (recall)
    - Bình thường: 0.86
    - Viêm phổi vi khuẩn: 0.92
    - Viêm phổi virus: 0.75
- Điểm F1
    - Bình thường: 0.89
    - Viêm phổi vi khuẩn: 0.89
    - Viêm phổi virus: 0.76

Mô hình EfficientNetB0

- Độ chính xác (accuracy): 0.87
- Độ chính xác (precision)
    - Bình thường: 0.93
    - Viêm phổi vi khuẩn: 0.91
    - Viêm phổi virus: 0.73
- Khả năng thu hồi (recall)
    - Bình thường: 0.81
    - Viêm phổi vi khuẩn: 0.93
    - Viêm phổi virus: 0.85
- Điểm F1
    - Bình thường: 0.87
    - Viêm phổi vi khuẩn: 0.92
    - Viêm phổi virus: 0.79

Nhận xét

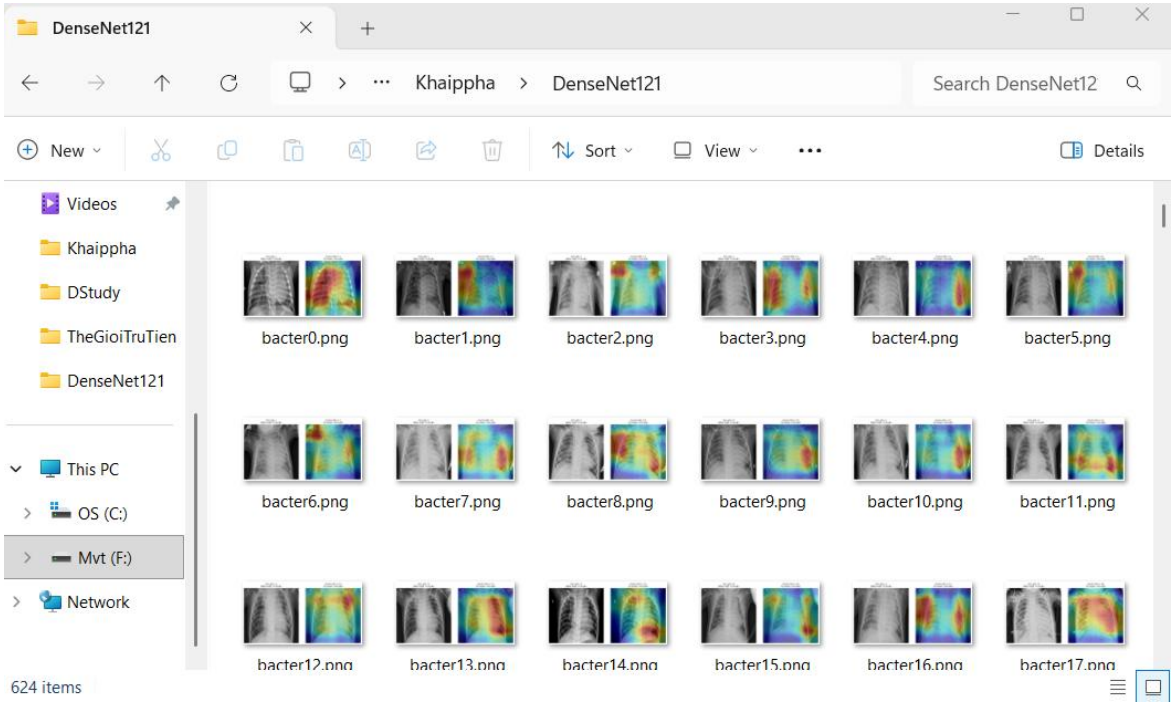
- Kết hợp với ma trận nhầm lẫn và chỉ số trên ta dễ thấy EfficientNetB0, như đã nói từ trước là việc phân loại bài toán phụ rất tốt nhưng nhầm lẫn bài toán chính thì hơn so với hai mô hình DenseNet121 và Resnet50.
- Còn lại hai mô hình DenseNet121 và Resnet50 thì dễ thấy ở điểm F1-score ta dễ thấy Resnet50 đang làm tốt hơn về bài toán phụ so với mô hình DenseNet121.

Đánh giá dựa trên GradCam++

Kỹ thuật khai phá xem mô hình đang sử dụng đặc trưng gì trên bức ảnh để mà đưa ra dự đoán.

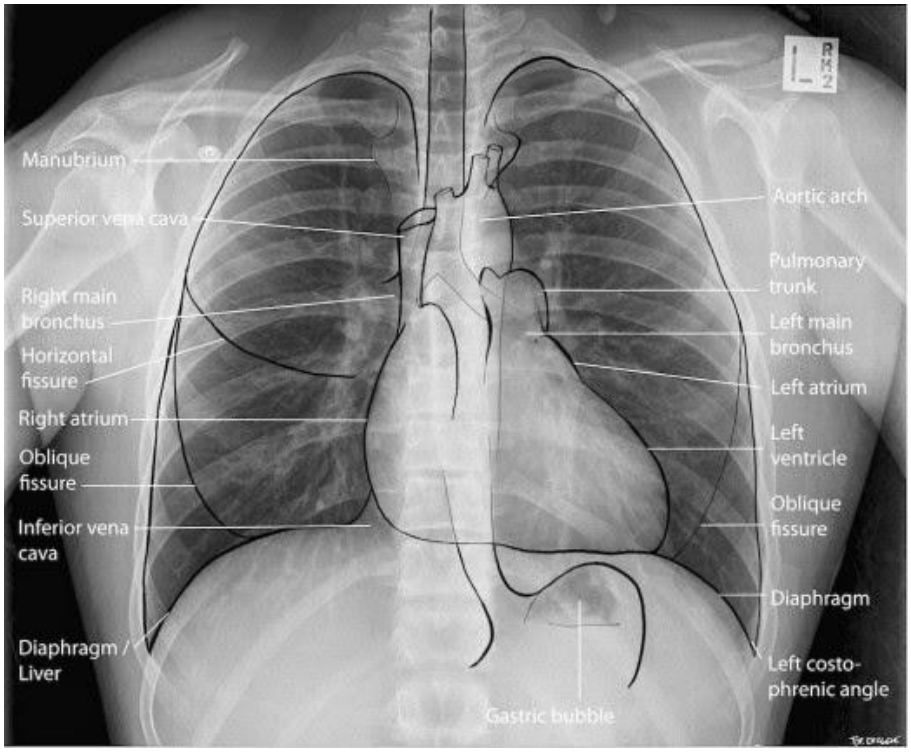
Với bài toán viêm phổi, thì mô hình học phải tập trung vào vị trí của phổi để mà đưa ra dự đoán đồng nghĩa là trích xuất được tổn thương ở đâu vị trí phổi.

Mỗi mô hình có 624 hình ảnh.



Hình 30. Chạy gradcam++ cho dữ liệu test

Yêu cầu đặt ra : Gradcam++ chỉ cần màu vàng hoặc màu đỏ (đặc sắc) vào vị trí phổi thì được tính.



Hình 31. Vị trí của phổi trong x-quang

Liệt kê xem mỗi mô hình có bao nhiêu hình ảnh đúng với yêu cầu.

Với dữ liệu test kích thước là : 624 hình ảnh

- Resnet50 : 463 (74%)
- DenseNet121 : 485 (77%)
- EfficientNetB0 : 480 (76%)

Triển khai website

Các công nghệ sử dụng

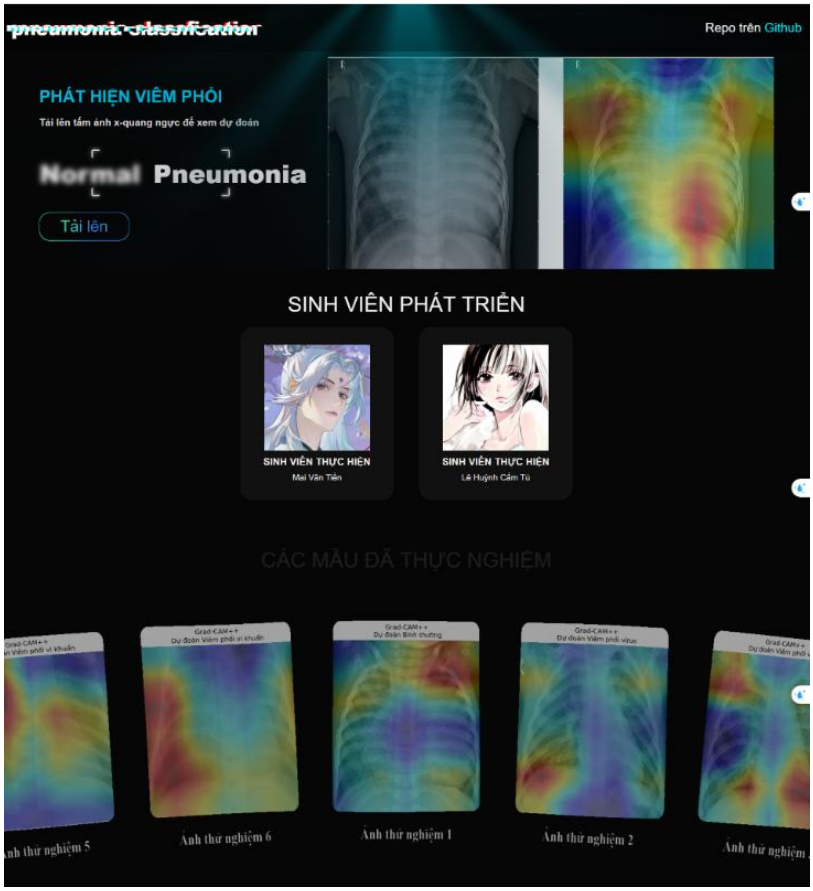
Client:

- Công cụ build: Vite
- Framework: ReactJS
- Ngôn ngữ sử dụng: TypeScript
- Giao diện sử dụng: Reactbits

Backend:

- Framework: Flask
- Ngôn ngữ sử dụng: Python
- Các thư viện sử dụng: opencv, matplotlib, tensorflow, keras

Giao diện người dùng



Hình 32. Hình ảnh giao diện website

6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết luận

Mô hình được lựa chọn là mô hình DenseNet121

Với độ chính xác khi huấn luyện là 86%, và được đánh giá với tập test với độ chính xác là 86%.

DenseNet121 thỏa mãn hai yêu cầu mà đề tài đặt ra đó là 2 mục đích.

- ***Mục đích chính*:** Sử dụng mô hình học sâu để phát hiện ra tình trạng phổi là bình thường hay là viêm phổi
- ***Mục đích phụ*:** Khi mô hình dự đoán là viêm phổi thì có thể dự đoán xem tác nhân của nó.

Tuy nhìn về các chỉ số có thể thấy Resnet50 hơn về độ chính xác và f1-score so với DenseNet121

Ngoài ra còn là một mô hình nhẹ hơn Resnet50 và khai phá dữ liệu mô hình học hỏi trích xuất từ Gradcam++ ta dễ thấy DenseNet121 với 77% toàn bộ hình ảnh được cho là tìm đúng vị trí phổi so với 74% mà Resnet50 mang lại.

Trang website cũng đã triển khai với hiệu suất mô hình cũng ổn định đã sẵn sàng trở thành một công cụ hỗ trợ cho các bác sĩ chẩn đoán viêm phổi.

Đường dẫn dự án website: https://github.com/mvtvn78/pneumonia_classification

Đường dẫn file ipynb:

https://drive.google.com/file/d/1JNuaBBXfUEiYKli_wH1Ns41u2nllprR/view?usp=sharing

Hướng phát triển

- Công cụ có thể mở rộng ra nhiều bệnh về lồng ngực hơn chứ không chỉ là viêm phổi đặc thù.
- Xây dựng phần mềm đa nền tảng có thể giúp cho bác sĩ linh hoạt sử dụng.
- Hợp tác với các bác sĩ chuyên gia để mà thu thập, mở rộng thêm dữ liệu cho mô hình, thí dụ dữ liệu này chỉ là trẻ em thì mở rộng thêm người lớn và người già.
- Mong sự nghiên cứu và phát triển có thể đóng góp một phần nào đó cho y tế và góp phần cứu giúp mọi người trong việc điều trị bệnh của mình.

7. TÀI LIỆU THAM KHẢO

[1] Cleveland Clinic (2025), *"Tổng quan về viêm phổi"*

https://my.clevelandclinic.org/health/diseases/4471-pneumonia

[2] Ths.BS.Chu Văn Đặng (2010), Nhà xuất bản giáo dục Việt Nam, *"Chẩn Đoán Hình Ảnh Xquang",*

[3] Bệnh viện Quân Y 103 (2025), *"Chẩn đoán X Quang lồng ngực"*

https://benhvien103.vn/chan-doan-x-quang-long-nguc/

[4] Grad-CAM++: Improved Visual Explanations for Deep Convolutional Networks (2017)

https://arxiv.org/abs/1710.11063

[5] CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning (2017)

https://arxiv.org/abs/1711.05225