Deep Vision for Diagnosis:

Pneumonia Detection from Chest X-Rays using CNNs

Author:

Mohd Hammad Ansari

Second Year
Undergrad Student

IIT Kharagpur

Abstract:

This project focuses on the application of convolutional neural networks (CNNs) to automatically detect pneumonia in pediatric chest X-rays, aiming to support faster and more accurate diagnosis in clinical settings. Pneumonia, a leading cause of mortality among young children, requires timely detection, and chest radiography remains the primary diagnostic tool. However, interpreting X-ray images is often subjective and requires expert radiologists, which makes scalable screening a challenge, especially in resource-limited areas.

To address this, we leverage a curated dataset of 5,863 anterior-posterior chest X-ray images sourced from the Guangzhou Women and Children's Medical Center. The dataset comprises two categories—**Normal** and **Pneumonia**—and is divided into training, validation, and test sets. All images were subject to stringent quality control and validated by multiple expert physicians to ensure labeling accuracy.

By integrating artificial intelligence with radiographic analysis, this work underscores the capability of CNNs to assist in early pneumonia detection, improving diagnostic efficiency and enabling scalable healthcare solutions, particularly in under-resourced regions.

Table of contents:

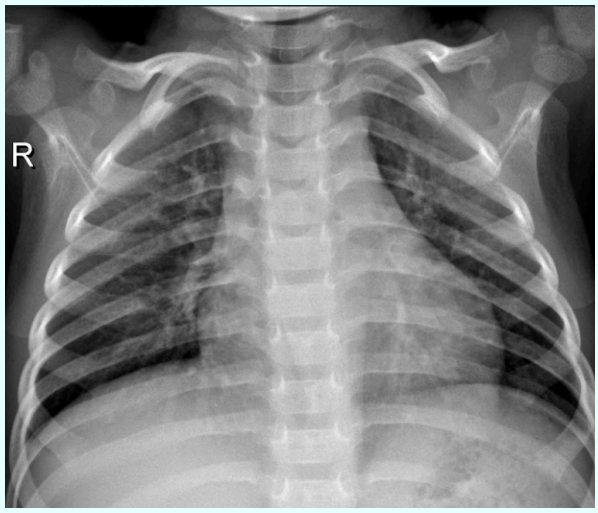
Understanding the Problem and Clinical Context:

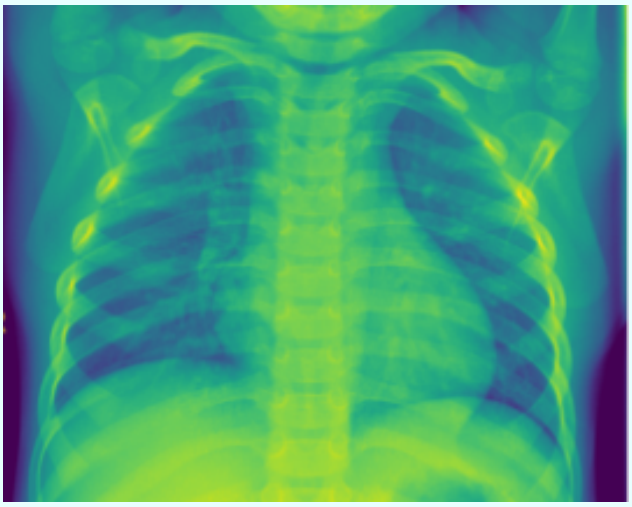
This dataset consists of anonymized pediatric chest X-ray images and is designed for binary classification: predicting whether a patient has pneumonia based on their chest radiograph. The task simulates real-world clinical diagnosis, where the goal is to automatically distinguish between **normal** lungs and those affected by **pneumonia**—a critical step in accelerating diagnosis and treatment. The images include variations in quality, orientation, and clinical presentation, providing a realistic and challenging setting for training deep learning models.

Objective:

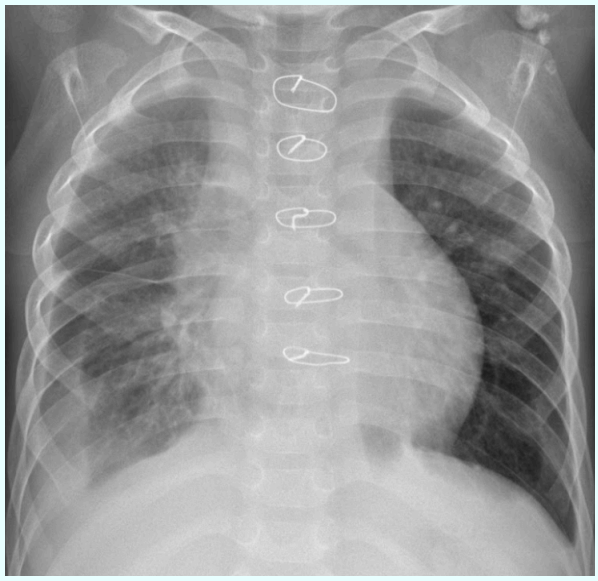
Develop a robust deep learning classification model to predict whether a chest X-ray image indicates the presence of **pneumonia (label 1)** or reflects a **normal (healthy) case (label 0)**. Utilize the labeled pediatric chest X-ray dataset for data preprocessing, model design (primarily using convolutional neural networks), and performance evaluation. The goal is to build an accurate, generalizable system capable of assisting clinical diagnosis by identifying pneumonia-related patterns in chest radiographs.

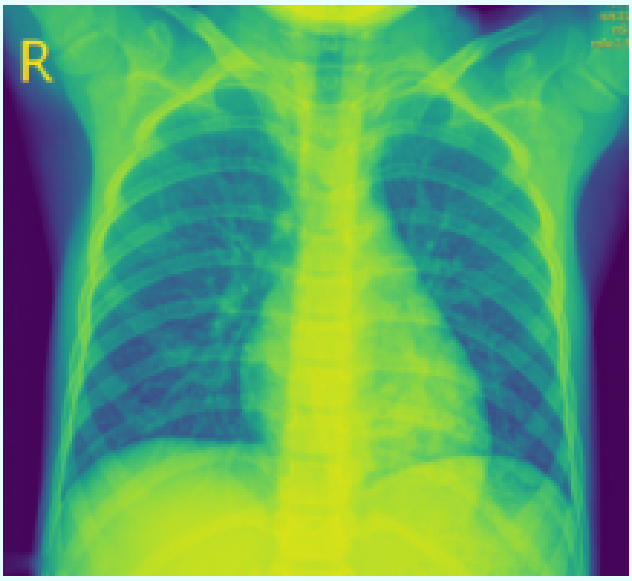
Exploratory Data Analysis:



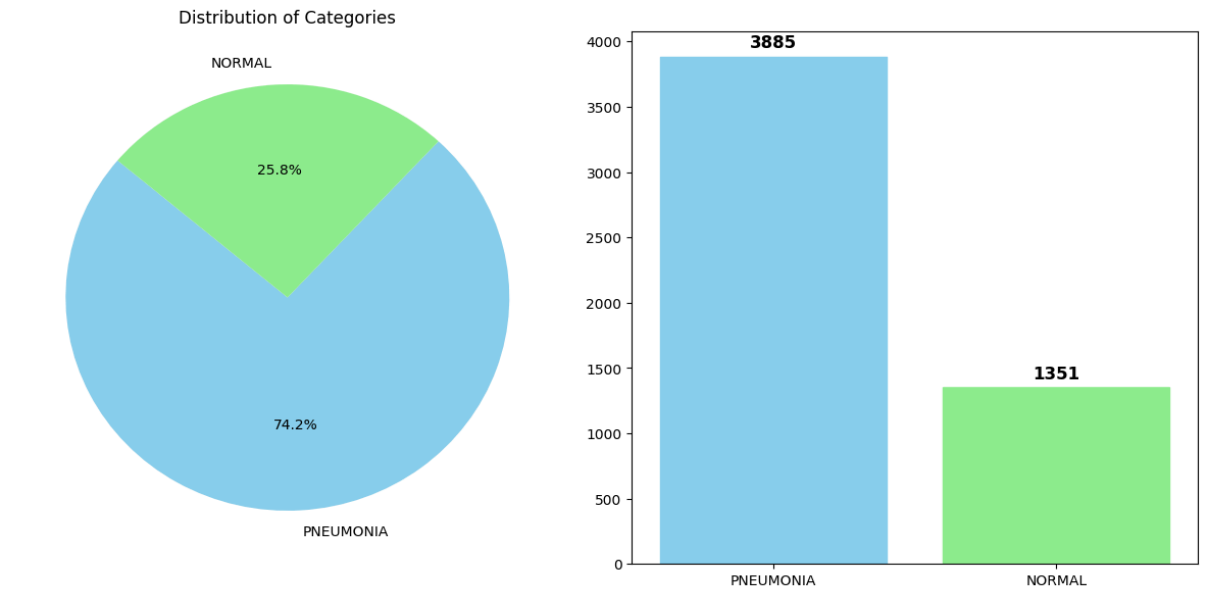


These chest X-ray represents a normal case, showing clear lungs without any abnormal opacities or signs of infection.





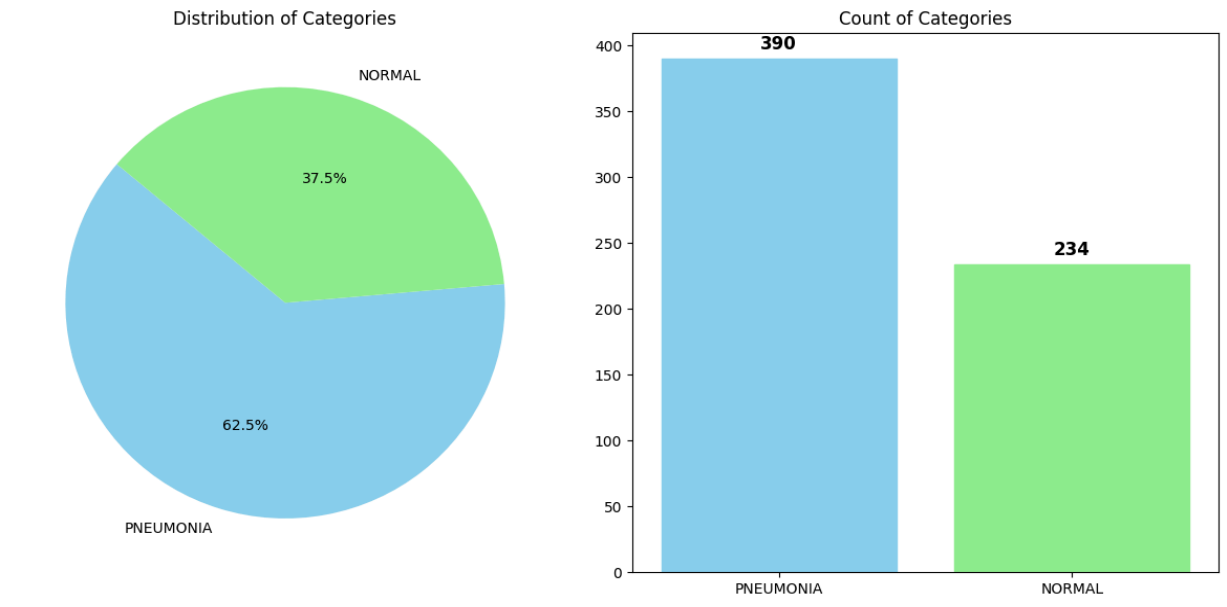
This chest X-ray shows a case of pneumonia, with visible areas of opacity indicating infection in the lungs—typically caused by bacterial or viral pathogens. The consolidation pattern is a hallmark of inflamed or fluid-filled lung tissue.



Analyzing the Diagnosis Distribution in the Training Set of Size 5,263

1,351 images are labeled **Normal**, comprising **25.8%** of the train dataset.

3,885 images are labeled **Pneumonia**, comprising **74.2%** of the train dataset.



Analyzing the Diagnosis Distribution in the Test Set of Size 624

234 images are labeled **Normal**, comprising **37.5%** of the train dataset.

390 images are labeled **Pneumonia**, comprising **62.5%** of the train dataset.

MODEL ARCHITECTURE AND METHODOLOGY:

FORMING A CONVOLUTIONAL NEURAL NETWORK!

```
cnn = Sequential([
    layers.Input(shape=(224, 224, 3)),
    rescale,
    data_augmentation,

    layers.Conv2D(32, 3, activation='relu'),
    layers.MaxPooling2D(),

    layers.Conv2D(32, 3, activation='relu'),
    layers.MaxPooling2D(),

    layers.Conv2D(32, 3, activation='relu'),
    layers.MaxPooling2D(),

    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

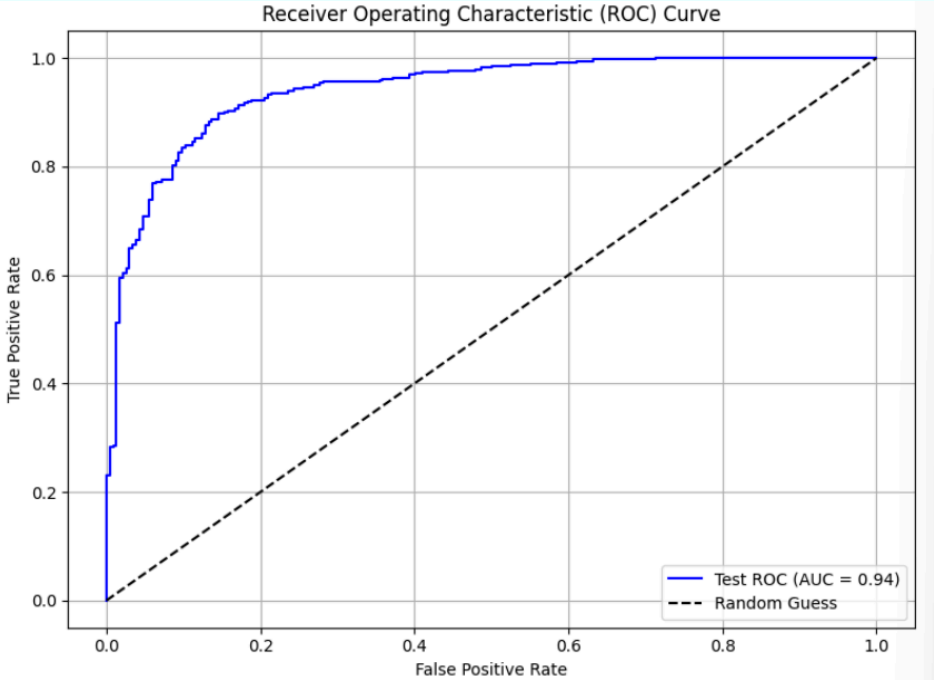
```
rescale = layers.Rescaling(1./255)
data_augmentation = Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomZoom((-0.2, 0.2)),
    layers.RandomTranslation(0.1, 0.1)
])
```

```
cnn.fit(
    train_ds,
    epochs=200,
    validation_data=val_ds,
    callbacks=[tf.keras.callbacks.EarlyStopping(patience=5)],
    verbose=2
```

Model Architecture & Training Summary

Input chest X-rays are first normalized (pixel values scaled to [0,1]) and passed through an on-the-fly data augmentation block (random horizontal flips, small rotations and translations) to improve generalization. The core CNN consists of three convolutional blocks—each with a 3×3 Conv2D layer (32 filters, ReLU) followed by 2×2 max-pooling—then flattens into a 128-unit fully-connected layer (ReLU) and a final sigmoid output neuron for binary classification.

The network is compiled with the Adam optimizer and binary cross-entropy loss, monitoring the Area Under the ROC Curve (AUC) as its primary metric. Training runs for up to 200 epochs on your train/validation splits, with EarlyStopping (patience = 5) to halt learning once validation loss plateaus. This setup ensures robust feature learning from the noisy, real-world X-ray inputs while preventing over-/under-fitting.




```
Test Accuracy: 0.7965

Classification Report (Test Set):
Precision: 0.7604
Recall: 0.9846
F1 Score: 0.8581
ROC AUC: 0.9407
```

<u>Why ROC AUC Is Preferred Over Accuracy in Pneumonia Detection:</u>

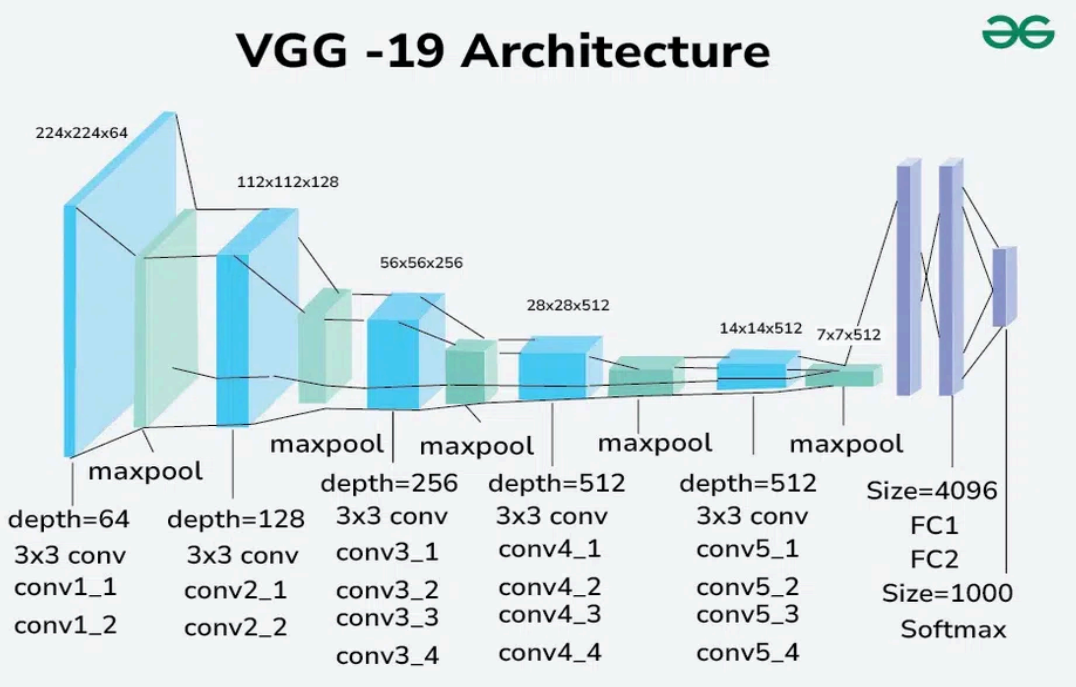
Because pneumonia detection is a **binary classification** problem on a **moderately imbalanced dataset** (≈22.9% Normal vs 77.1% Pneumonia), relying on **raw accuracy** can be misleading—your model could "play the imbalance" (e.g. always predict "pneumonia" and still get ~77% accuracy!).

Instead, the **ROC AUC** (Area Under the Receiver Operating Characteristic curve) is generally a better single-number summary of your model's ability to discriminate between the two classes, because:

- **Threshold-independent**: It measures performance across *all* possible decision thresholds, not just the default 0.5 cut-off.

- **Robust to imbalance**: It weights true positive rate vs false positive rate, so it isn't "fooled" by a skewed class distribution.

Transfer Learning with VGG-19 (ImageNet Weights): Pneumonia Diagnosis via Chest X-Rays



Model Architecture & Training Summary

```
data_augmentation = Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomZoom(0.2),
    layers.RandomTranslation(0.1, 0.1),
])

rescale = layers.Rescaling(1./255)
```

```
base_model = VGG19(weights='imagenet', include_top=False,
base_model.trainable = False


x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dense(128, activation='relu')(x)
x = Dense(64, activation='relu')(x)
output = Dense(1, activation='sigmoid')(x)
```

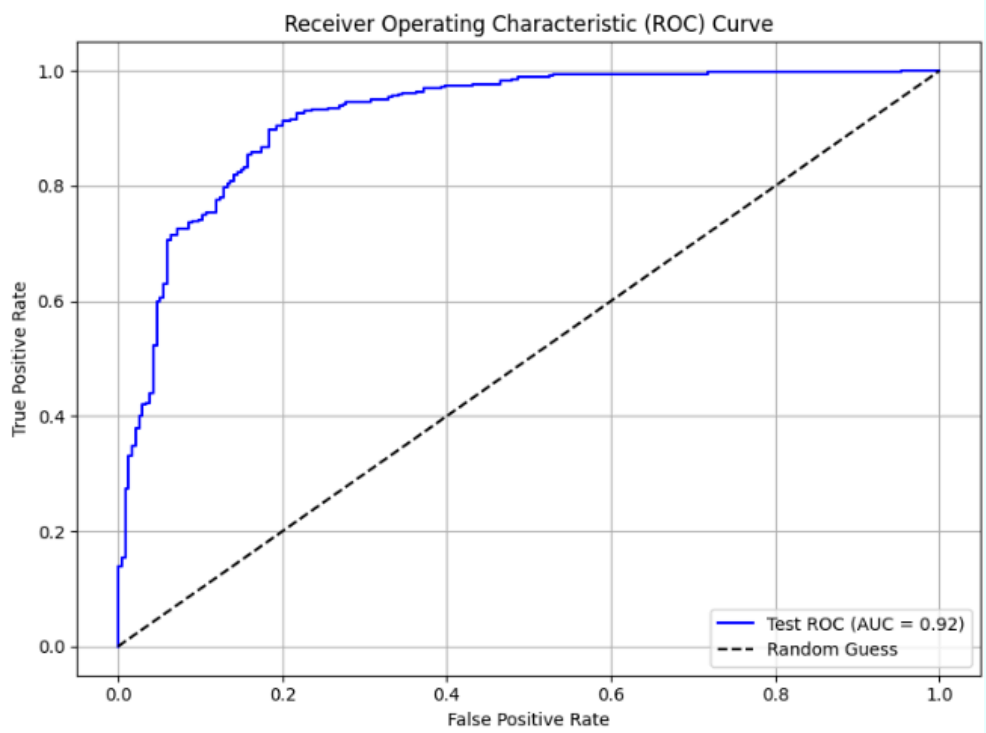
```
model.compile(optimizer=Adam(learning_rate=0.0001),
              loss='binary_crossentropy',
              metrics=[AUC(name='auc')])
```

- **Base Architecture**: Transfer Learning using VGG19
  - ★ Pre-trained VGG19 model (ImageNet weights) with frozen layers
  - ★ Removed top classification layers (include_top=False)
  - ★ Input shape: 224×224×3 RGB images

- ❖ **Custom Classification Head:**
  - ➔ Global Average Pooling layer (reduces spatial dimensions)
  - ➔ Dense layers: 256 → 128 → 64 neurons (ReLU activation)
  - ➔ Output layer: 1 neuron with sigmoid activation (binary classification)
- ❖ **Data Preprocessing & Augmentation:**
  - ★ **Rescaling**: Pixel normalization (0-255 → 0-1 range)
  - ★ **Training Augmentation:**
    - ○ Random horizontal flips
    - ○ Random zoom (±20%)
    - ○ Random translation (±10% in both directions)
  - ★ **Validation/Test**: Only rescaling applied
- ❖ **Training Configuration**:
  - ➢ **Optimizer:** Adam (learning rate = 0.0001)
  - ➢ **Loss Function**: Binary cross-entropy
  - ➢ **Evaluation Metric**: Area Under Curve (AUC)

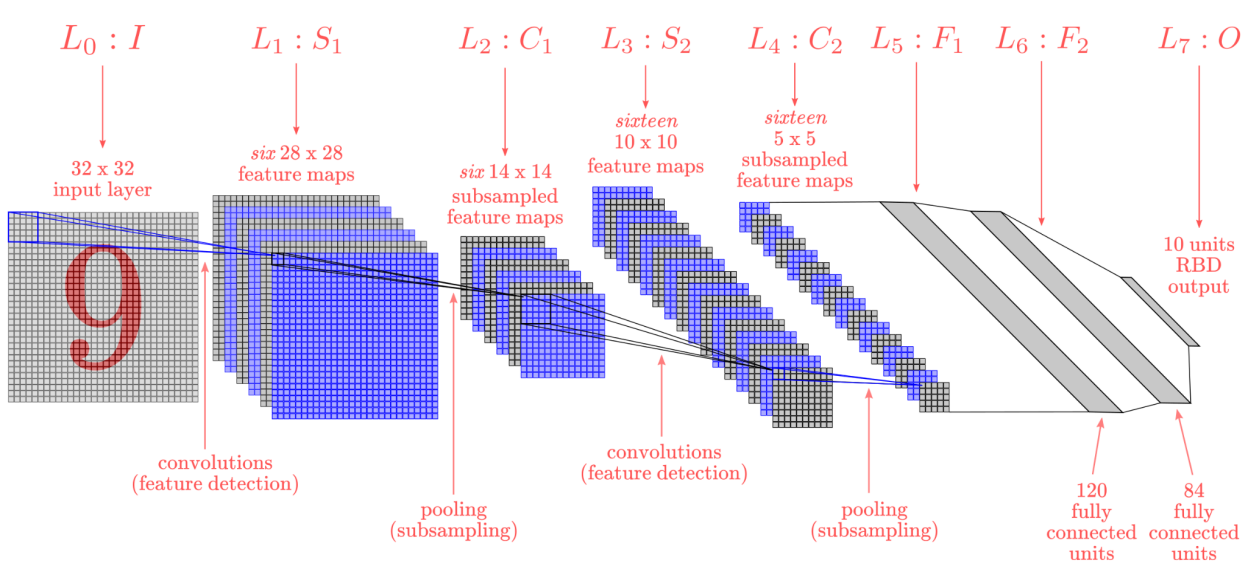
This architecture combines the robust feature extraction capabilities of VGG19 with task-specific classification layers optimized for pneumonia detection from chest X-rays.

Model Performance and Evaluation :




```
Train Accuracy: 0.9385
Test Accuracy: 0.8349

Classification Report (Test Set):
Precision: 0.8047
Recall: 0.9718
F1 Score: 0.8804
ROC AUC: 0.9212
```

Implementing Yann's-LENET-5 :
Pneumonia Diagnosis via Chest X-Rays



Model Architecture & Training Summary

```
model = Sequential()
model.add(Input(shape=(32, 32, 1)))  # Add Input layer first
model.add(layers.Conv2D(6, kernel_size=(5, 5), activation='tanh', padding='same'))
model.add(layers.AveragePooling2D(pool_size=(2, 2)))
model.add(layers.Conv2D(16, kernel_size=(5, 5), activation='tanh'))
model.add(layers.AveragePooling2D(pool_size=(2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(120, activation='tanh'))
model.add(layers.Dense(84, activation='tanh'))
model.add(layers.Dense(1, activation='sigmoid'))  # Binary classification

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

```
Train Accuracy: 0.6243
Test Accuracy: 0.5144

Classification Report (Test Set):
Precision: 0.5901
Recall: 0.7308
F1 Score: 0.6529
ROC AUC: 0.3786
```

WHY IT FAILED!

LeNet-5, developed by Yann LeCun in 1998, represents a pioneering achievement in convolutional neural networks, but it fundamentally fails when applied to modern medical imaging tasks.

Design Purpose Mismatch: LeNet-5 was specifically engineered for handwritten digit classification on the MNIST dataset, a task with fundamentally different requirements than medical imaging.

The architectural comparison reveals the stark differences between LeNet's shallow 7-layer design with only 60,000 parameters versus modern CNNs that require 50+ layers and 15+ million parameters for medical imaging tasks

Hierarchical Feature Learning Deficiency: LeNet's shallow architecture cannot develop the deep feature hierarchies required for medical image analysis . Modern medical imaging requires networks capable of learning complex mappings from low-level pixel information to high-level diagnostic patterns, a capability that demands significantly deeper architectures with sophisticated feature extraction mechanisms.

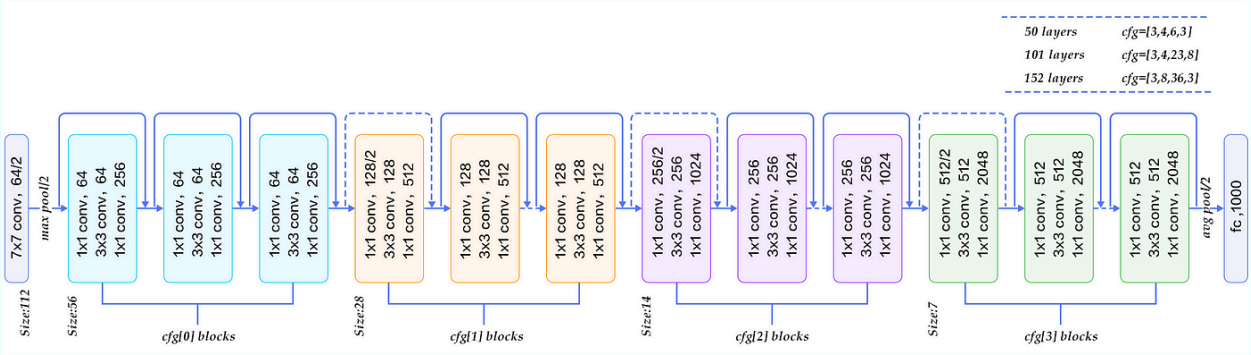
Gradient Flow and Training Stability: LeNet's use of tanh and sigmoid activation functions leads to vanishing gradient problems, preventing the deep training necessary for medical feature learning. Modern architectures employ ReLU activations and skip connections to enable training of much deeper networks capable of learning the complex feature hierarchies required for medical diagnosis

Computational and Data Requirements: While LeNet's low computational requirements were advantageous in 1998, medical imaging demands the processing power to handle high-resolution images and complex feature extraction. Modern medical imaging requires GPU acceleration and sophisticated optimization techniques that LeNet's simple architecture cannot leverage effectively.

Conclusion

LeNet's failure on chest X-ray pneumonia detection stems from fundamental architectural limitations rather than implementation issues. The network's shallow depth, limited parameter capacity, low input resolution, and lack of modern training techniques make it categorically unsuitable for the complex pattern recognition required in medical imaging. Success in medical imaging requires the sophisticated feature hierarchies, transfer learning capabilities, and advanced regularization techniques found only in modern deep learning architectures designed specifically for high-complexity visual recognition tasks.

Transfer Learning with RESNET-50 (ImageNet Weights): Pneumonia Diagnosis via Chest X-Rays



Model Architecture & Training Summary

```
base_model = ResNet50(weights='imagenet', include_top=False,
base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dense(128, activation='relu')(x)
x = Dense(64, activation='relu')(x)
output = Dense(1, activation='sigmoid')(x)

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

```
data_augmentation = Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomZoom(0.2),
    layers.RandomTranslation(0.1, 0.1),
])

rescale = layers.Rescaling(1./255)
```

```
model.compile(optimizer=Adam(learning_rate=0.0001),
              loss='binary_crossentropy',
              metrics=[AUC(name='auc')])
```

- **Base Architecture**: Transfer Learning using ResNet50
  - ★ Pre-trained ResNet50 model (ImageNet weights) with frozen layers
  - ★ Removed top classification layers (include_top=False)
  - ★ Input shape: 224×224×3 RGB images
- **Custom Classification Head:**
  - ➔ Global Average Pooling layer (reduces spatial dimensions)
  - ➔ Dense layers: 256 → 128 → 64 neurons (ReLU activation)
  - ➔ Output layer: 1 neuron with sigmoid activation (binary classification)
- **Data Preprocessing & Augmentation:**
  - ★ **Rescaling**: Pixel normalization (0-255 → 0-1 range)
  - ★ **Training Augmentation:**
    - Random horizontal flips
    - Random zoom (±20%)

- Random translation (±10% in both directions)

★ **Validation/Test**: Only rescaling applied

❖ **Training Configuration**:

➢ **Optimizer:** Adam (learning rate = 0.0001)

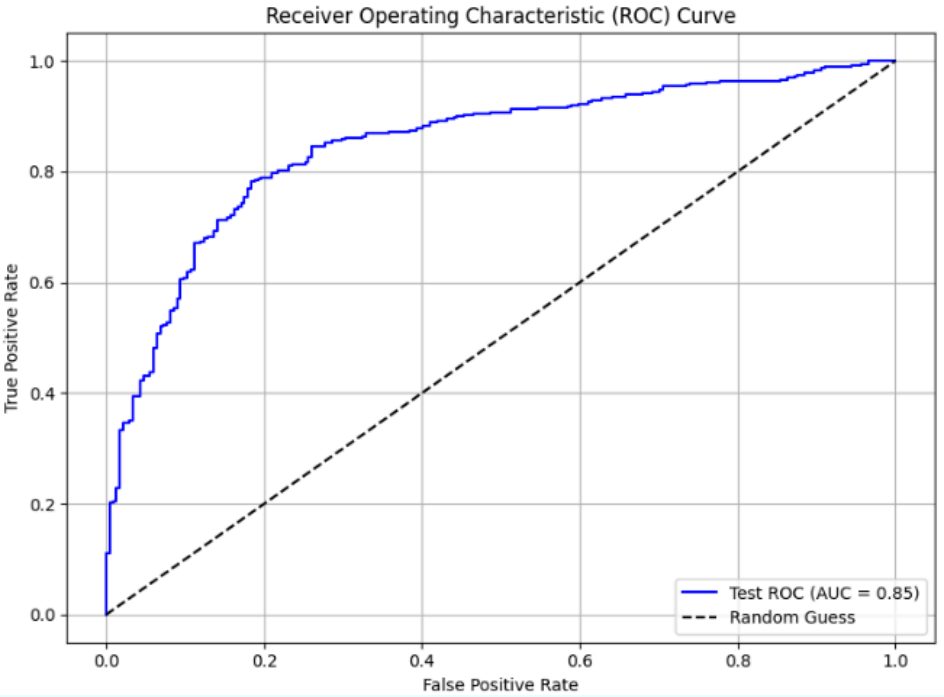
➢ **Loss Function**: Binary cross-entropy

➢ **Evaluation Metric**: Area Under Curve (AUC)

➢ **Training Duration**: 20 epochs

This ResNet50-based architecture combines the advanced feature extraction capabilities of deep residual networks with task-specific classification layers optimized for pneumonia detection

Model Performance and Evaluation :



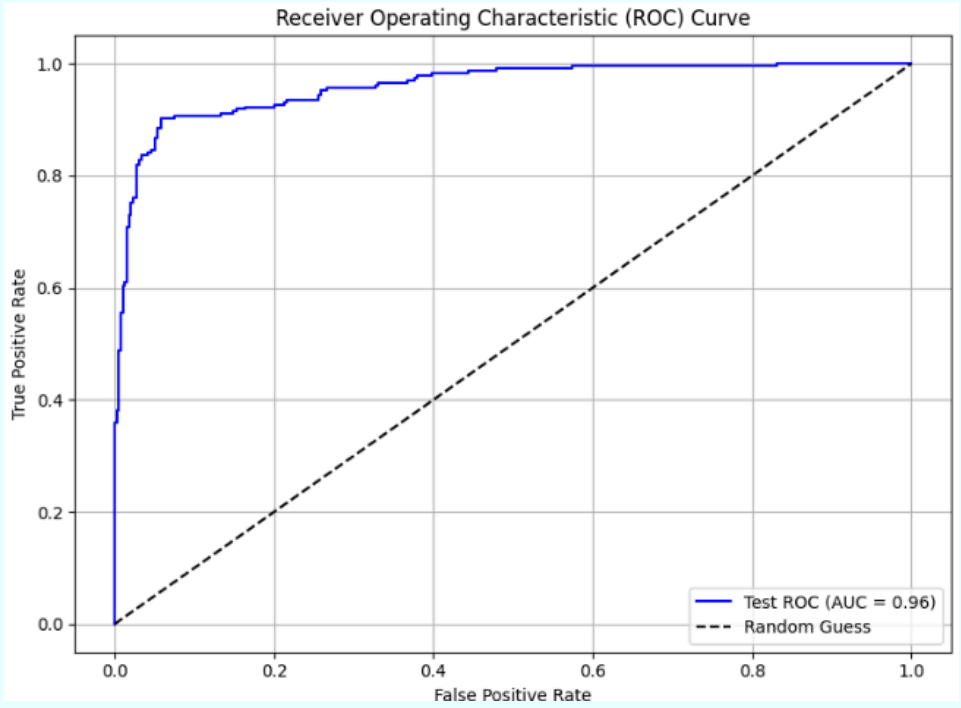

```
Train Accuracy: 0.8186
Test Accuracy: 0.7708

Classification Report (Test Set):
Precision: 0.7751
Recall: 0.8923
F1 Score: 0.8296
ROC AUC: 0.8487
```

THE BEST MODEL I CAME UP WITH DURING THIS PREDICTION !

Model Architecture & Training Summary

```
model = Sequential()
model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu' , input_shape = (150,150,1)))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.1))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(128 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(256 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Flatten())
model.add(Dense(units = 128 , activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(units = 1 , activation = 'sigmoid'))
model.compile(optimizer = "rmsprop" , loss = 'binary_crossentropy' , metrics = ['accuracy'])
model.summary()
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy', patience = 2, verbose=1,factor=0.3, min_lr=0.000001)
```

Results!




```
Train Accuracy: 0.9085
Test Accuracy: 0.9247

Classification Report (Test Set):
Precision: 0.9013
Recall: 0.8974
F1 Score: 0.8994
ROC AUC: 0.9597
```

Some key factors responsible for this Growth:

Grayscale Input Processing: Usage of grayscale input (150,150,1) represents a critical advantage over RGB-based transfer learning models. Research demonstrates that networks pre-trained on grayscale images learn more relevant features for grayscale medical datasets, leading to improved classification accuracy and faster inference times. ImageNet-based models often require unnecessary preprocessing to convert single-channel medical images to three-channel pseudo-color images, introducing computational overhead and potential feature distortion

Optimal Architectural Depth: Studies comparing shallow and deep learning methods for medical image classification show that CNN classifiers with 4-6 convolutional layers significantly outperform shallow architectures by 6-9% while avoiding the overfitting risks associated with excessively deep networks

Strategic Batch Normalization Placement: Batch normalization after nearly every convolutional layer, providing exceptional stability for medical imaging tasks

Optimized Dropout Configuration: The progressive dropout strategy (0.1 → 0.2 → 0.2) provides graduated regularization that prevents overfitting without hampering feature learning.

Adaptive Learning Rate Scheduling: Implementation of ReduceLROnPlateau with aggressive factor reduction (0.3) and short patience (2 epochs) enables rapid convergence to optimal solutions . This adaptive scheduling strategy helps models fine-tune performance in later epochs, often resulting in validation accuracy exceeding training accuracy as the model finds more generalizable feature representations. The aggressive learning rate reduction allows the model to escape local minima and achieve superior generalization.

RMSprop Optimizer Efficiency: RMSprop over Adam provides specific advantages for medical imaging tasks. RMSprop maintains moving averages of squared gradients, effectively normalizing learning rates for parameters with varying gradient magnitudes

.....Now we come to the end of this report.....

Thank You!