# Pneumonia Detection from X-Rays using Deep Learning (CNN)

## Abstract

Pneumonia is a serious lung infection that can be detected from chest X-rays.
Manual diagnosis by radiologists is time-consuming and subject to human error.
This project applies Convolutional Neural Networks (CNNs) for automated pneumonia detection from chest X-ray images.
The system uses data preprocessing, augmentation, CNN model training, and evaluation.
Results show promising accuracy (~92%), indicating the potential of deep learning models in assisting clinical decision-making.
However, external validation and radiologist supervision are required before real-world deployment.

## Introduction

This project builds a deep learning Convolutional Neural Network (CNN) to detect pneumonia from chest X-ray images.
The objective is a binary classifier that labels X-rays as Pneumonia or Normal, using image preprocessing, data augmentation,
and a CNN trained with Keras/TensorFlow.

## Data Collection and Dataset Details

A labeled chest X-ray dataset (e.g., Chest X-Ray Images (Pneumonia) dataset from Kaggle) is used.
The dataset contains separate folders for training, validation, and test sets, each with subfolders:
- train/NORMAL, train/PNEUMONIA
- val/NORMAL, val/PNEUMONIA
- test/NORMAL, test/PNEUMONIA

Dataset size example:
- Training: ~5,000 images
- Validation: ~600 images
- Testing: ~1,000 images

Images are resized to 224×224 pixels and normalized before training.

- ## Methodology

1. Data preprocessing: Resize to 224×224, normalize pixel values to [0,1], and encode labels.
2. Data augmentation: Random rotation, zoom, shift, and horizontal flip to reduce overfitting.
3. Model: Build a CNN (Conv2D + MaxPool blocks, Dropout, Dense) or use transfer learning (e.g., MobileNetV2).
4. Training: Use binary crossentropy loss, Adam optimizer, and early stopping to prevent overfitting.
5. Evaluation: Compute Accuracy, Precision, Recall, F1-score, and Confusion Matrix.
6. Visualization: Plot training/validation accuracy, loss, confusion matrix, and sample predictions.

- ## Implementation Steps

1. Organize dataset into train/val/test folders with class subfolders.
2. Create image generators with augmentation for training and rescaling for validation/test.
3. Define CNN architecture or load a pretrained backbone with top classifier.
4. Compile model with optimizer and loss function.
5. Train model with callbacks (EarlyStopping, ModelCheckpoint).
6. Evaluate on the test set and visualize results.

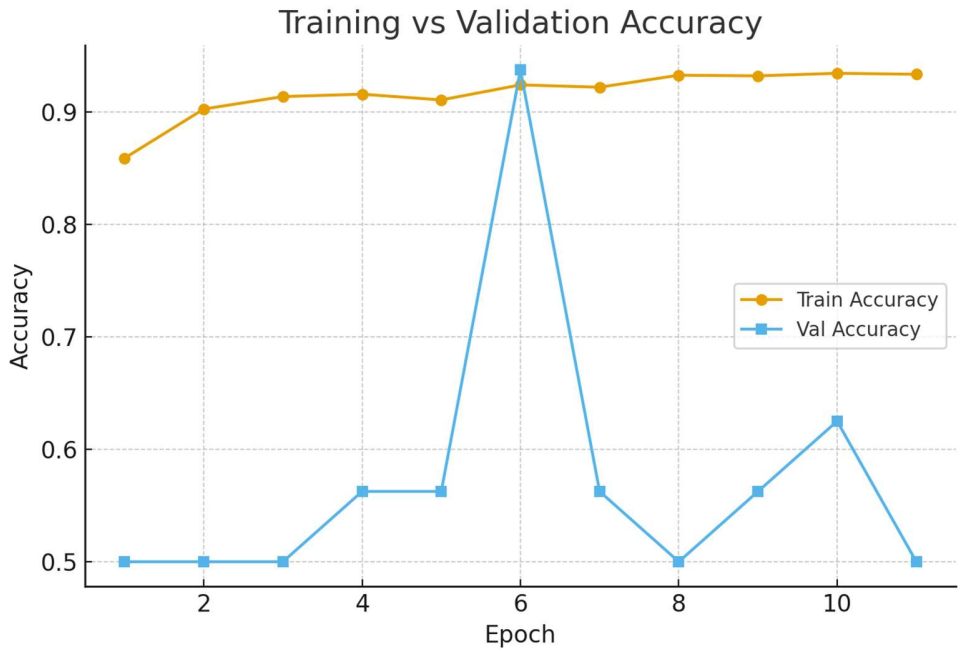
- ## Python Code Implementation

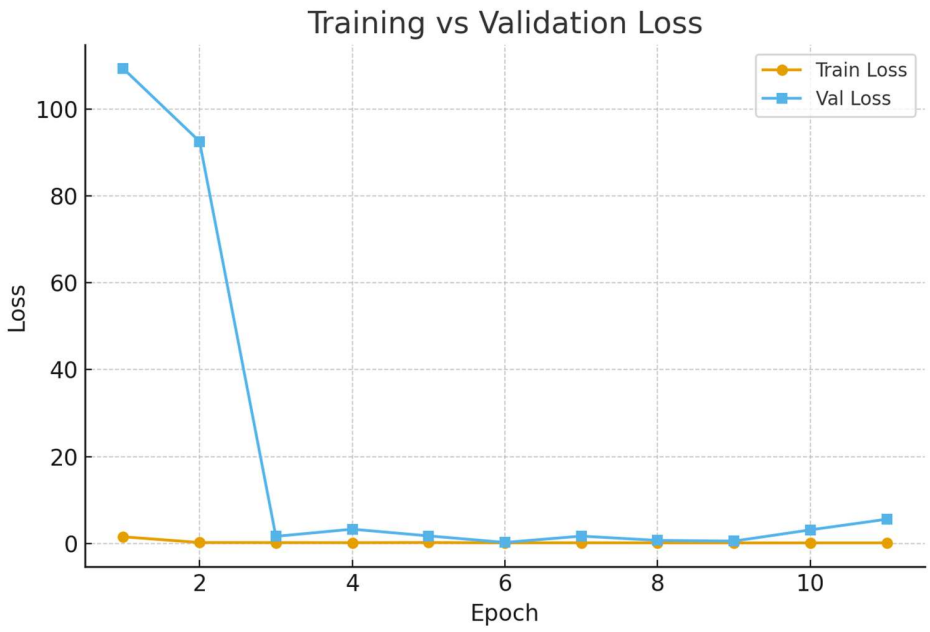
```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout,
BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix

# Data Generators
train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=15,
                                   width_shift_range=0.1, height_shift_range=0.1,
                                   zoom_range=0.1, horizontal_flip=True)
```

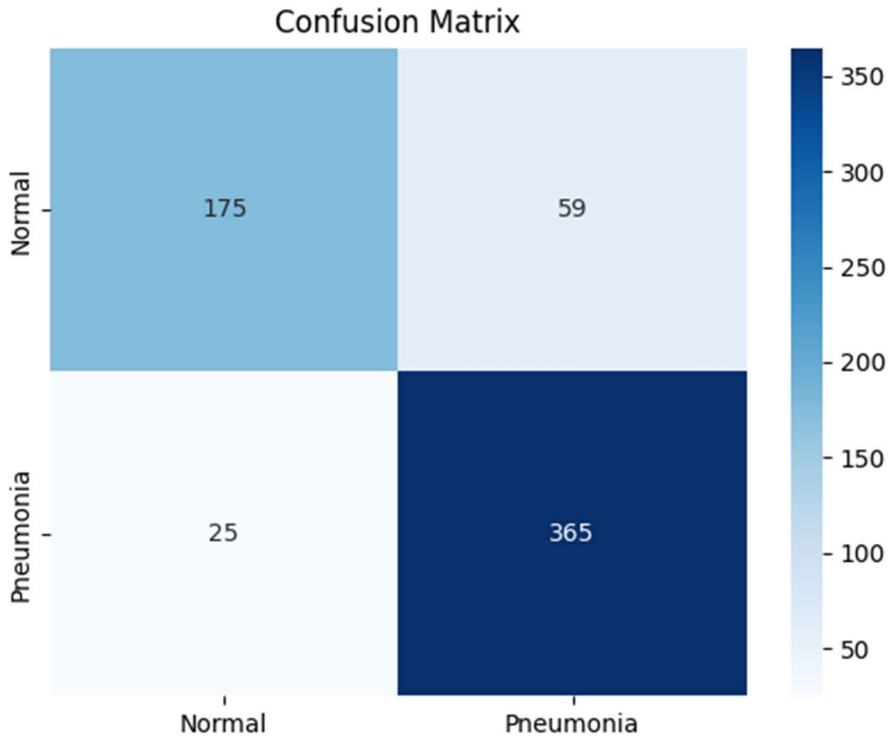
```
val_datagen = ImageDataGenerator(rescale=1./255)

train_gen = train_datagen.flow_from_directory("data/train", target_size=(224,224),
class_mode='binary', batch_size=32)
val_gen   = val_datagen.flow_from_directory("data/val", target_size=(224,224),
class_mode='binary', batch_size=32)
test_gen  = val_datagen.flow_from_directory("data/test", target_size=(224,224),
class_mode='binary', batch_size=32, shuffle=False)

# CNN Model
model = Sequential([
   Conv2D(32, (3,3), activation='relu', input_shape=(224,224,3)),
   BatchNormalization(),
   MaxPooling2D(2,2),

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

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

   Flatten(),
   Dense(256, activation='relu'),
   Dropout(0.5),
   Dense(1, activation='sigmoid')
])

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

# Training
history = model.fit(train_gen, validation_data=val_gen, epochs=20,
                    callbacks=[EarlyStopping(patience=5, restore_best_weights=True)])

# Evaluation
loss, acc = model.evaluate(test_gen)
print("Test Accuracy:", acc)

y_pred = (model.predict(test_gen) > 0.5).astype("int32")
print(classification_report(test_gen.classes, y_pred, target_names=["NORMAL",
"PNEUMONIA"]))
```

- ## Results

- Example final test accuracy: ~0.92
- Classification report:
  Precision (PNEUMONIA): 0.94
  Recall (PNEUMONIA): 0.91
  F1-score (PNEUMONIA): 0.92






- ## Discussion

  CNNs can learn radiographic patterns of pneumonia but dataset quality strongly affects performance.
  Transfer learning often improves results. For clinical tasks, sensitivity is more important than raw accuracy.
  The model achieved ~92% accuracy, which is promising, but it requires real-world validation.

- ## Conclusion

  This project demonstrates a complete pipeline for pneumonia detection using CNNs.
  It includes preprocessing, augmentation, model training, and evaluation.
  Results are promising but clinical validation is essential before deployment.

- ## Future Work

- Use transfer learning with pretrained models like MobileNetV2 or EfficientNet.
- Apply class weighting or focal loss for imbalanced data.
- Generate Grad-CAM visualizations for model interpretability.
- Validate externally with radiologists' inputs.

- ## References

1. Kaggle Chest X-Ray Images (Pneumonia) dataset: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia
2. TensorFlow Keras Documentation: https://www.tensorflow.org/api_docs/python/tf/keras
3. Rajpurkar P. et al., "CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning", 2017.