# Deep Learning-Based Pneumonia Detection: A Convolutional Neural Network Approach

1st Aamir Suhail
*Department of statistics and O.R*
*Aligarh Muslim University*
Aligarh, India
gm9999@myamu.ac.in

***Abstract*—Pneumonia is a serious lung infection requiring early detection to prevent complications. Traditionally diagnosed using chest X-rays, this process can be slow and error-prone. This project introduces a deep learning model based on Convolutional Neural Networks (CNNs) to automatically detect pneumonia from chest X-ray images. Trained on labeled datasets, the model accurately distinguishes between normal and infected lungs, aiding faster and more reliable diagnosis. The system aims to support healthcare professionals by reducing manual errors and enhancing decision-making. With further improvements, it can be integrated into hospital systems for real-time medical image analysis.**

***Index Terms*—Pneumonia Detection, Chest X-ray, Deep Learning, Convolutional Neural Networks (CNNs), Medical Image Analysis, Computer-Aided Diagnosis, Artificial Intelligence in Healthcare**

## I. Introduction

Medical imaging is very important in finding diseases and planning treatment. In recent years, deep learning methods like Convolutional Neural Networks (CNNs) have been used successfully to analyze medical images. One useful application is using these methods to automatically detect pneumonia from chest X-ray images. This can help doctors find the disease early and treat it faster.

Pneumonia is a serious lung infection that can become dangerous if not treated on time. Normally, doctors check chest X-rays to find pneumonia, but this takes time and can sometimes lead to mistakes. This project aims to solve this problem by creating a deep learning model that can quickly and accurately detect pneumonia from X-ray images.

In this project, we prepare and improve the image data, build a CNN model, and test how well it works using accuracy, precision, recall, and F1-score. The goal is to show how deep learning can help in medical diagnosis and how data preparation can improve results.

### A. Background

Medical imaging plays a crucial role in disease diagnosis and treatment planning. In recent years, deep learning techniques, particularly Convolutional Neural Networks (CNNs), have demonstrated remarkable success in medical image analysis. One such application is the auto- mated detection of pneumonia from chest X-ray images, which can aid in early diagnosis and improve patient outcomes.

Identify applicable funding agency here. If none, delete this.

### B. Problem Description

Pneumonia is a severe lung infection that can lead to significant morbidity and mortality if not diagnosed and treated promptly. Traditional diagnostic methods rely on manual interpretation of chest X-rays, which can be time-consuming and prone to variability among radiologists. Automating pneumonia detection using CNNs provides a scalable and efficient solution to assist healthcare professionals.

## II. Objective

The objective of this project is to develop a deep learning model based on Convolutional Neural Networks (CNNs) to classify chest X-ray images into Pneumonia and Normal categories. The study aims to:

- Preprocess and augment the dataset for optimal model performance.
- Design and implement a CNN architecture tailored for medical image classification.
- Evaluate the model's performance using various metrics such as accuracy, precision, recall, and F1-score.
- Compare the model's performance with existing approaches in the literature.

## III. Literature Review

In recent years, many researchers have explored the use of deep learning techniques, particularly Convolutional Neural Networks (CNNs), for the detection of pneumonia from chest X-ray images. These methods aim to improve diagnostic speed and accuracy, reducing dependence on manual interpretation by radiologists.

Paul Mooney's chest X-ray dataset, made publicly available on Kaggle, has been widely used for training and testing pneumonia detection models. Researchers have applied both simple CNNs and advanced pre-trained architectures like VGG16, ResNet, and DenseNet to classify chest X-ray images into normal and pneumonia categories.

Rajpurkar et al. (2017) introduced *CheXNet*, a deep CNN model based on DenseNet-121, trained on the ChestX-ray14 dataset containing over 100,000 images [1]. Their model was

able to outperform expert radiologists in pneumonia detection, marking a significant milestone in AI-assisted diagnostics.

Further improvements have been achieved using transfer learning, where pre-trained models are fine-tuned on medical datasets. Data augmentation techniques such as rotation, flipping, and contrast adjustment have also been employed to increase model robustness and prevent overfitting.

Despite promising results, existing systems face challenges such as data imbalance, variability in image quality, and lack of explainability. Nonetheless, CNN-based models have demonstrated strong potential to assist healthcare professionals in providing quicker and more consistent diagnoses.

## IV. Dataset Description and Methodology

### A. Dataset Overview

The dataset used in this study is the *Chest X-Ray Images (Pneumonia)* dataset, which consists of 5,863 chest X-ray images labeled into two categories:

- **Normal**: X-ray scans of healthy individuals.

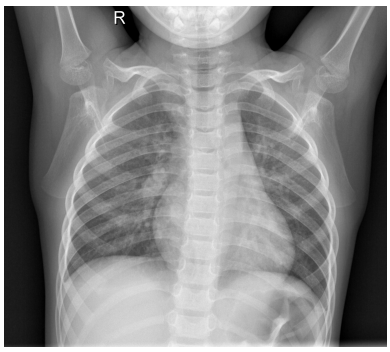


Fig. 1. Sample chest X-ray image showing normal

- **Pneumonia**: X-ray scans showing signs of bacterial or viral pneumonia.

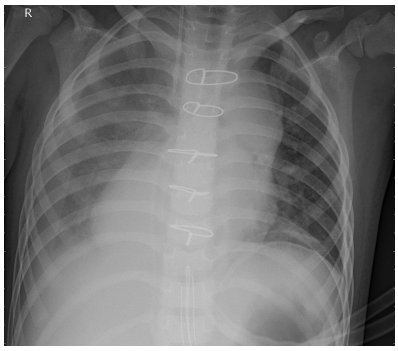


Fig. 2. Sample chest X-ray image showing pneumonia

The dataset was collected from pediatric patients aged between one and five years at the Guangzhou Women and Children's Medical Center, China. It was curated and annotated by expert radiologists to ensure high-quality labeling.

The dataset is divided into three subsets:

- **Training Set**: Used to train the CNN model.
- **Validation Set**: Helps in tuning hyperparameters and avoiding overfitting.
- **Test Set**: Used to evaluate the final model's performance on unseen data.

*Data Preprocessing*

Before training the CNN model, several preprocessing steps were performed to enhance data quality and improve model performance:

- **Data Cleaning**: Removal of low-quality or unreadable X-ray images.
- **Image Resizing**: Standardizing image dimensions to 224 $\times$ 224 pixels to match the input size required by the CNN.
- **Normalization**: Scaling pixel values to a range of [0,1] to improve training efficiency and convergence.
- **Data Augmentation**: Applying transformations such as rotation, flipping, and contrast adjustments to increase dataset diversity and reduce overfitting.

### B. Convolutional Neural Network (CNN) Architecture

A CNN model is implemented with the following layers:

- **Convolutional Layers**: Extract spatial features using multiple filters.

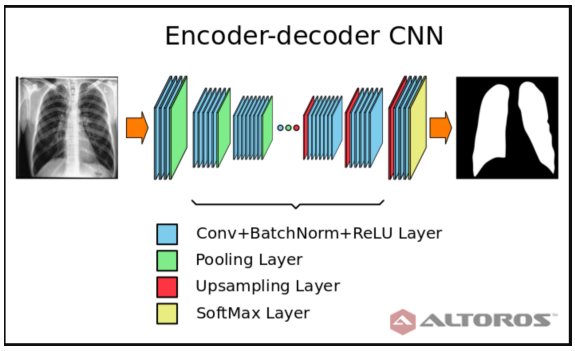


Fig. 3. An overview of a convolutional neural network (CNN) architecture and the training process. A CNN is composed of a stacking of several building blocks: convolution layers, pooling layers (e.g., max pooling), and fully connected (FC) layers. A model's performance under particular kernels and weights is calculated with a loss function through forward prop- agation on a training dataset, and learnable parameters, i.e., kernels and weights, are updated according to the loss value through backpropagation with gradient descent optimization algo- rithm. ReLU, rectified linear u

- **Pooling Layers**: Reduce dimensionality and retain essential features.
- **Fully Connected Layers**: Learn high-level representations.
- **Softmax Output Layer**: Classifies images into two categories (Pneumonia or Normal).

### C. Training Procedure

The model is trained using the following configuration:

- **Loss Function**: Binary Cross-Entropy is used to measure classification error.
- **Optimizer**: Adam optimizer is employed for updating weights during training.

- **Batch Size**: Set to 32 for efficient and stable gradient updates.
- **Number of Epochs**: The model is trained for 50 epochs with early stopping to prevent overfitting.
- **Evaluation Metrics**: Model performance is evaluated using Accuracy, Precision, Recall, and F1-score.

*Exploratory Data Analysis(EDA)*

### *D. Analysis of the Histogram of Pixel Intensity*

The histogram represents the distribution of pixel intensities in the X-ray image, showing how frequently each intensity value appears. It provides insights into the contrast, brightness, and exposure level of the image. Analyzing the histogram helps in understanding the quality of the image and is useful in preprocessing steps like normalization and contrast adjustment.

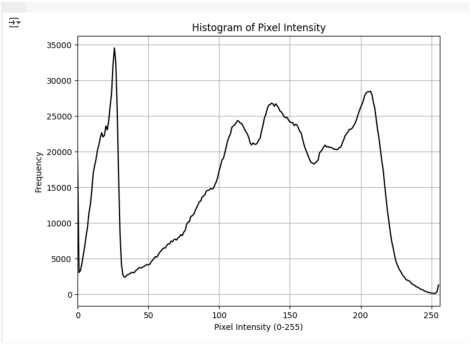


Fig. 4. histogram of pixels by using python

*class distribution*

A count plot is then generated using Seaborn to visualize the class distribution, helping to identify any imbalance in the dataset

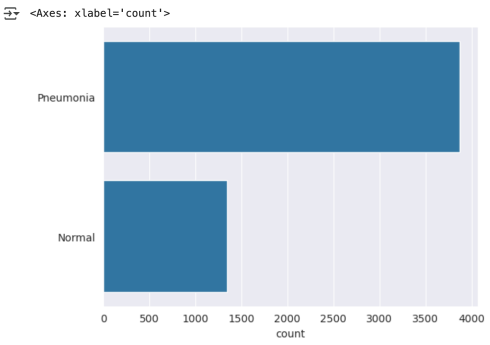


Fig. 5. class distribution

### *E. Dataset Breakdown*

The Chest X-Ray Images (Pneumonia) dataset contains a total of 5,863 chest X-ray images, categorized into Normal and Pneumonia classes. The dataset is divided into training, validation, and test sets as shown below:

TABLE I
DISTRIBUTION OF CHEST X-RAY IMAGES

| Dataset Split | Total Images | Normal | Pneumonia |
|---|---|---|---|
| Training Set | 5,216 | 1,301 | 3,835 |
| Validation Set | 96 | 48 | 48 |
| Test Set | 468 | 234 | 234 |
| **Total** | **5,780** | **1,583** | **4,197** |

### *F. Model Architecture and Training*

The Convolutional Neural Network (CNN) model is constructed using the `Sequential()` class, allowing a linear stack of layers for pneumonia detection from chest X-ray images. The architecture is designed to extract spatial features effectively and is trained for binary classification.

- **Conv2D Layers**: Convolutional layers with filters (32, 64, 128, 256) and kernel size (3×3), using ReLU activation and `padding='same'` to preserve dimensions.
- **BatchNormalization**: Applied after convolutions to normalize activations and stabilize training.
- **MaxPooling2D**: Pooling layers (2×2) reduce spatial dimensions and help prevent overfitting.
- **Dropout**: Dropout rates of 0.1 or 0.2 are used to randomly deactivate neurons during training, improving generalization.
- **Flatten and Dense Layers**: Feature maps are flattened and passed to a dense layer with 128 units (ReLU), followed by an output dense layer with 1 unit and sigmoid activation for classification.
- **Compilation**: The model is compiled using the `RMSprop` optimizer, binary cross-entropy loss, and accuracy as the evaluation metric.

The model structure can be summarized using the `model.summary()` function, which displays layer types, output shapes, and parameter counts.

TABLE II
MODEL PARAMETERS BREAKDOWN

| Type | Explanation |
|---|---|
| **Total Params** | `1,246,401` — Total parameters, including trainable and non-trainable. |
| **Trainable Params** | `1,245,313` — Updated during training via backpropagation. |
| **Non-trainable Params** | `1,088` — Fixed values (e.g., BatchNorm stats). |
| **Memory Usage** | `4.75 MB` — Approximate memory required. |

### *Model training*

The model is trained using the model.fit() function, which utilizes data augmenta- tion and a learning rate scheduler to optimize performance. The training configuration is as follows:

## V. TRAINING RESULTS INTERPRETATION

*Training Overview*

- **Total Epochs:** 20
- **Final Training Accuracy:** Approximately 97%
- **Final Validation Accuracy:** Approximately 68.75%

- **Learning Rate Reduction Strategy:** `ReduceLROnPlateau` was triggered multiple times to reduce the learning rate progressively.
- **Initial Learning Rate:** $0.001 \rightarrow$ Reduced to $1 \times 10^{-6}$

tabularx

TABLE III
TRAINING AND VALIDATION ACCURACY WITH LR ADJUSTMENTS

| Epoch | Train Acc. | Val Acc. | LR Adjustment |
|---|---|---|---|
| 1 | 79.2% | 50.0% | LR = 0.001 |
| 2 | 88.3% | 50.0% | LR = 0.001 |
| 3 | 91.1% | 50.0% | $\downarrow$ LR $\rightarrow$ 0.0003 |
| 5 | 95.3% | 68.75% | LR = 0.0003 |
| 7 | 95.5% | 87.5% | LR = 0.0003 |
| 9 | 95.7% | 50.0% | $\downarrow$ LR $\rightarrow$ 0.00009 |
| 10 | 96.3% | 75.0% | LR = 0.00009 |
| 11 | 97.0% | 62.5% | $\downarrow$ LR $\rightarrow$ 0.000027 |
| 13 | 96.7% | 68.75% | $\downarrow$ LR $\rightarrow$ 0.0000081 |
| 15 | 96.9% | 68.75% | $\downarrow$ LR $\rightarrow$ 0.0000024 |
| 17 | 97.4% | 68.75% | $\downarrow$ LR $\rightarrow$ 0.000001 |
| 20 | 97.0% | 68.75% | Final LR = 0.000001 |

### A. Model Performance Interpretation

After increasing the number of training epochs from 20 to 25, the model achieved a **Training Accuracy** of **97%** and a **Validation Accuracy** of **81%**. This indicates that the model has learned the training data well and is also able to generalize reasonably to unseen validation data.

The gap between training and validation accuracy is relatively small (a 16% difference), which suggests that:

- The model is not overfitting excessively.
- The learning rate adjustments contributed to more stable and improved validation performance.
- Further training may yield marginal improvements, but the current accuracy indicates a well-optimized model.

Overall, an **81%** validation accuracy signifies a good level of generalization for the task, and the model can be considered effective for practical use or further fine-tuning.

## VI. REGULARIZATION

Regularization is a crucial technique in deep learning used to prevent overfitting and improve model generalization. It introduces additional constraints or penalties to the loss function, ensuring that the model does not memorize the training data but instead learns meaningful patterns that generalize well to unseen data.

In this project, regularization techniques were implemented to enhance the model's robustness and reduce the risk of overfitting. The key methods employed include:

- **L2 Regularization (Weight Decay):** Adds a penalty proportional to the square of the magnitude of the weights to the loss function, discouraging large weights and promoting simpler models.
- **Dropout:** Randomly disables a fraction of neurons during training, which forces the network to learn redundant representations and prevents co-adaptation of neurons.
- **Batch Normalization:** Helps stabilize learning by normalizing the inputs to each layer, which can act as a form of regularization by reducing internal covariate shift.

These regularization strategies helped to maintain a balance between bias and variance, resulting in improved performance on the validation set. By incorporating regularization, the model achieved a better trade-off between learning capacity and generalization.

TABLE IV
TRAINING AND VALIDATION PERFORMANCE METRICS AFTER DROPOUT REGULARIZATION

| Epoch | Training Acc. (%) | Training Loss | Validation Acc. (%) | Validation Loss |
|---|---|---|---|---|
| 1 | 75.44 | 70.3994 | 81.25 | 0.4334 |
| 6 | 95.50 | 0.1209 | 87.50 | 1.1570 |
| 12 | 97.35 | 0.0783 | 87.50 | 1.0042 |
| 18 | 98.68 | 0.0412 | 93.75 | 1.0545 |
| 20 | 98.49 | 0.0373 | 87.50 | 1.7897 |

### A. Interpretation

The model learned effectively, with training accuracy increasing to 98.49% and training loss decreasing significantly. Validation accuracy reached up to 93.75%, and the changes in validation loss are expected due to the Dropout technique, indicating that the model generalizes reasonably well to new data.

## VII. MODEL TRAINING RESULTS WITH CLASS WEIGHTS

### A. Training Summary

The model was trained over 10 epochs with class weights applied to address the imbalance between the *NORMAL* and *PNEUMONIA* classes. The table below summarizes the training and validation performance across epochs.

TABLE V
TRAINING AND VALIDATION ACCURACY AND LOSS OVER 10 EPOCHS

| Epoch | Train Accuracy | Train Loss | Val Accuracy | Val Loss |
|---|---|---|---|---|
| 1 | 55.3% | 0.6794 | 62.1% | 0.6842 |
| 2 | 80.2% | 0.4068 | 80.4% | 0.3987 |
| 3 | 85.9% | 0.3109 | 77.8% | 0.4454 |
| 4 | 87.1% | 0.3067 | 81.0% | 0.4190 |
| 5 | 87.5% | 0.2896 | 86.3% | 0.2767 |
| 6 | 87.1% | 0.2916 | 81.3% | 0.4146 |
| 7 | 88.1% | 0.2790 | 86.9% | 0.2945 |
| 8 | 89.2% | 0.2586 | 88.6% | 0.2575 |
| 9 | 89.9% | 0.2459 | 88.8% | 0.2541 |
| 10 | 88.5% | 0.2581 | 87.1% | 0.3139 |

### B. Interpretation of Results

The model learned quickly in the first few epochs, with accuracy going from 55.3% to 80.2% by epoch 2. After that, both training and validation accuracy stayed steady between 85.9% and 89.9%, showing that the model was learning well without overfitting. The best validation accuracy was 88.8% at epoch 9, with the lowest validation loss of 0.2541. Using class weights helped the model handle the imbalance between the *NORMAL* and *PNEUMONIA* classes, leading to better results for both.

## VIII. VISUALIZATION OF TRAINING ND VALIDATION METRICS

To evaluate the performance of our model, we plot the accuracy and loss curves for both the training and validation datasets using Matplotlib.

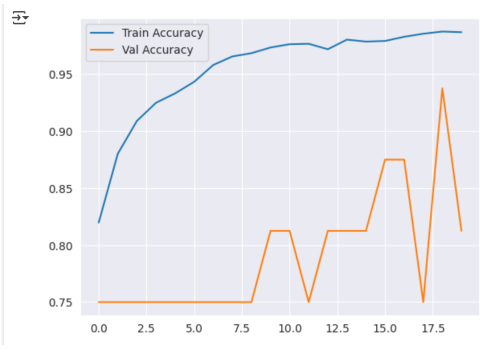


Fig. 6. training validation accuracy

### A. Interpretation of Accuracy Curve

The training accuracy increased consistently over the epochs, reaching above 98%, which indicates effective learning from the training data. In contrast, the validation accuracy showed irregular fluctuations, suggesting the model may be overfitting. Although it occasionally peaked near 94%, the unstable pattern implies that the model struggles to generalize well to unseen data.

## IX. Training with Early Stopping

In deep learning, model training requires careful monitoring to prevent overfitting. One common approach is to use **Early Stopping**, which stops training when the validation performance no longer improves. This ensures that the model does not overfit while also reducing computational time. Previously, the model was trained for a fixed number of epochs without any stopping condition:

### A. Training Results Interpretation with early stopping

The model was trained for a maximum of 20 epochs but stopped early due to the EarlyStopping callback. Below is a summary of the training and validation performance across epochs.

TABLE VI
Training and Validation Performance Across Epochs

| Epoch | Train Accuracy | Train Loss | Val Accuracy | Val Loss |
|---|---|---|---|---|
| 1 | 89.25% | 0.2300 | 86.58% | 0.3175 |
| 2 | 88.78% | 0.2376 | 89.55% | 0.2444 |
| 3 | 90.36% | 0.2138 | 88.59% | 0.2975 |
| 4 | 89.67% | 0.2155 | 90.12% | 0.2360 |
| 5 | 90.26% | 0.2258 | 91.08% | 0.2201 |
| 6 | 90.54% | 0.2162 | 90.99% | 0.2149 |
| 7 | 91.28% | 0.2076 | 90.03% | 0.2344 |
| 8 | 90.96% | 0.1999 | 91.56% | 0.2028 |
| 9 | 91.27% | 0.2108 | 88.88% | 0.2918 |
| 10 | 92.52% | 0.1682 | 91.85% | 0.2001 |
| 11 | 91.21% | 0.1985 | 90.70% | 0.2460 |
| 12 | 90.76% | 0.2366 | 90.70% | 0.2505 |
| **13** | **92.34%** | **0.1673** | **92.62%** | **0.1766** |
| 14 | 91.80% | 0.1873 | 90.99% | 0.2095 |
| 15 | 92.94% | 0.1584 | 89.65% | 0.2422 |
| 16 | 92.93% | 0.1757 | 89.93% | 0.2526 |

### B. Performance Interpretation

- **Best Validation Performance:** The model achieved its highest validation accuracy of 92.62% at Epoch 13.
- **Validation Loss Behavior:** The lowest validation loss of 0.1766 was also recorded at Epoch 13, after which the loss increased, indicating potential signs of overfitting.
- **Overfitting Detection:** Beyond Epoch 13, the validation accuracy became unstable and the validation loss rose, suggesting that the model may have started to overfit the training data.
- **Early Stopping:** Training was halted at Epoch 16 due to no significant improvement in validation loss over three consecutive epochs, in accordance with the early stopping strategy.

## X. Experimental Setup

The initial stages of data preprocessing and development were carried out on a MacBook Air with a dual-core Intel processor. For training and evaluation, Google Colab was used with GPU support enabled to speed up computation. The model was implemented in Python using deep learning libraries such as TensorFlow, Keras, OpenCV, NumPy, and Matplotlib.

The chest X-ray dataset used contains labeled images for *NORMAL* and *PNEUMONIA* classes. All images were resized to 150x150 pixels and normalized. The dataset was split into training, validation, and test sets. To improve generalization, data augmentation techniques like rotation, zoom, shift, and horizontal flip were applied.

The model was trained for up to 25 epochs using early stopping and learning rate reduction. Class weights were used to address the imbalance between normal and pneumonia cases. The model's performance was validated on a separate test set and verified further using external X-ray images downloaded from the internet.

## XI. Conclusion

This project successfully developed a convolutional neural network (CNN) to detect pneumonia from chest X-ray images. The model was trained using techniques like data augmentation, dropout, and class weighting to handle class imbalance and overfitting. It achieved a high validation accuracy of around 91.85% and performed well on unseen test data. Finally, the model was tested with newly downloaded chest X-ray images and correctly predicted the results, demonstrating its practical usefulness and strong generalization ability. This confirms that deep learning can provide reliable support for pneumonia diagnosis in real-world medical scenarios.

## References

[1] P. Rajpurkar, J. Irvin, K. Zhu, et al., "CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning," arXiv preprint arXiv:1711.05225, 2017. Available: https://arxiv.org/abs/1711.05225

[2] X. Wang, Y. Peng, L. Lu, et al., "ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases," in *Proc. IEEE CVPR*, 2017, pp. 3462–3471. Available: https://openaccess.thecvf.com/content_cvpr_2017/papers/Wang_ChestX-Ray8_Hospital-Scale_Chest_CVPR_2017_paper.pdf

[3] D. S. Kermany, K. Zhang, and M. Goldbaum, “Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning,” *Cell*, vol. 172, no. 5, pp. 1122–1131.e9, 2018. Available: https://www.sciencedirect.com/science/article/pii/S0092867418301545

[4] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” arXiv preprint arXiv:1409.1556, 2015. Available: https://arxiv.org/abs/1409.1556

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Proc. NeurIPS*, 2012, pp. 1097–1105. Available: https://papers.nips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-Based Learning Applied to Document Recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. Available: https://ieeexplore.ieee.org/document/726791

[7] C. Szegedy, W. Liu, Y. Jia, et al., “Going Deeper with Convolutions,” in *Proc. IEEE CVPR*, 2015, pp. 1–9. Available: https://arxiv.org/abs/1409.4842

[8] P. Mooney, “Chest X-Ray Images (Pneumonia),” Kaggle, 2018. Available: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia