AI-PneumoScan: Early Pneumonia Detection System

Group ID: – DLP -1

Team Members & Rollno.:

1. Arihant Jain - 27

2. Ashwin Rahate – 29

Abstract

Pneumonia is a severe respiratory illness that targets mainly the lungs and can cause serious health issues if not diagnosed. Manual diagnosis using chest X-rays is subject to medical expertise and subjectivity. This project introduces a Convolutional Neural Network (CNN)-oriented deep learning approach to automatically identify pneumonia from grayscale chest X-ray images. Our suggested model contains several convolutional, pooling, dropout, and normalization layers optimized to extract applicable spatial features. The model is trained on preprocessed chest X-ray data, and it records high accuracy when classifying pneumonia versus normal lungs. The result shows the effectiveness of deep CNNs in classifying medical images, especially early pneumonia detection.

Introduction

Pneumonia is a condition that impacts millions of people around the globe and presents a serious challenge for healthcare systems, mainly because it can develop quickly and lead to life-threatening issues. Diagnosing it manually through chest X-rays can be a lengthy process and often varies from one radiologist to another. That's why we're working on creating an automated pneumonia detection system using a Convolutional Neural Network (CNN). This cutting-edge deep learning technique has shown remarkable success in classifying images.

- Input: Our algorithm takes in a grayscale chest X-ray image that measures 128x128 pixels.
- Output: The result is a simple binary label that tells us whether the image shows ‘Pneumonia’ or ‘Normal’.

It follows a comprehensive process that includes image preprocessing, model training, validation, evaluation, and interpretation.

Related Work

When it comes to detecting pneumonia, earlier research has looked into both traditional machine learning and deep learning strategies. A notable example is Kermany et al. (2018), who successfully applied transfer learning to chest X-ray images, achieving remarkable accuracy with pre-trained models. Other researchers have opted for classical image processing techniques, followed by classifiers like SVM or Random Forest, but these often struggled during the feature extraction process.

In contrast, deep learning methods, especially CNNs, have taken the lead by learning features directly from the image pixels. Recent studies have made use of advanced architectures like DenseNet, ResNet, and VGG16. While pre-trained models offer solid benchmarks, custom-designed CNNs can be both efficient and yield comparable results.

Our approach stands out because we utilize a custom CNN that we train from scratch. This allows us to have more control over the architectural decisions and optimization, while also helping to avoid the overfitting issues that can arise with large transfer learning models on smaller datasets.

Dataset and Features

For our project on detection of pneumonia using deep learning, we had used the publicly available dataset named "Chest X-Ray Images (Pneumonia)" from Kaggle. The dataset has been extensively used in medical imaging research and comprises labeled chest X-ray scans belonging to two classes named 'Pneumonia' and 'Normal'. The dataset includes a collection of 5,854 grayscale images, with every image belonging to either of the two categories. The distribution of the data is significantly imbalanced, with more pneumonia cases than normal cases.

Dataset Composition

- Total Images: 5,854
    - Pneumonia: 4,271 images
    - Normal: 1,583 images

In order to provide unbiased assessment and effective training of the models, we carried out a stratified split of the dataset with a 70:15:15 ratio, separately applied to each class to maintain the initial class distribution among all subsets:

Dataset Split	Pneumonia Images	Normal Images	Total
Training	2987	1101	4088
Validation	642	248	890
Test	642	234	876

This stratified method ensured that every subset had the same class imbalance ratio as in the original dataset, enabling the model to generalize more across unseen data.

Preprocessing Pipeline

Prior to inputting the images into our convolutional neural network (CNN), we undertook a number of preprocessing steps to normalize the input data and enhance model performance:

- Grayscale Conversion: Even though the images were already grayscale, we made sure all images were kept in this format to make input dimensions easier (i.e., single-channel images).

- Image Resizing: All images were resized to 128×128 pixels to minimize computational expense while keeping visual features required for classification.

- Normalization: Pixel values were normalized from the original range of [0, 255] to a normalized range of [0, 1]. This stabilized training and accelerated convergence by having consistent feature scaling between inputs.

- Data Augmentation: In order to enhance the model's capability to generalize and prevent overfitting caused by the dataset's limited size, we utilized real-time data augmentation with Keras' ImageDataGenerator. The following augmentations were used:

    - Random Rotations (up to 15 degrees)

    - Zooming (up to 20%)

    - Shear Transformations (up to 15 degrees)

    - Horizontal Flipping

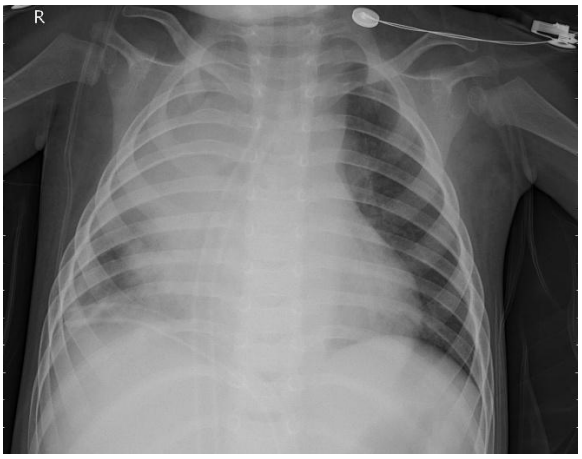
These transformations assisted the model in learning invariance to spatial orientations and minimal distortions that are inherent in real-world medical imaging.

Label Encoding

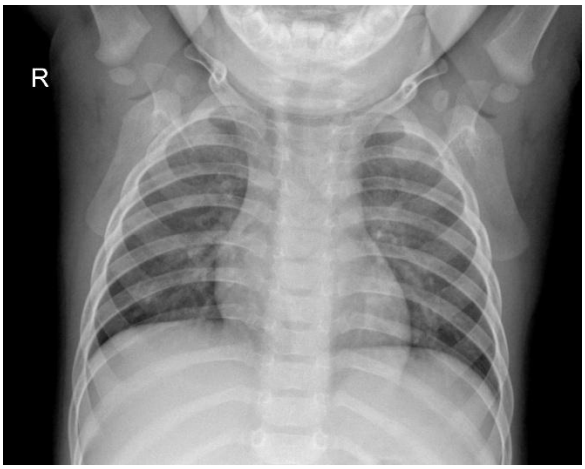
We encoded the categorical classes into numeric labels for compatability with binary classsification tasks:

- Label 0: Pneumonia
- Label 1: Normal

These labels were utilized during training with a binary cross-entropy loss function to inform the model's predictions.



Label 0: Pneumonia



Label 1: Normal

Feature Representation

In contrast to conventional image processing operations that are based on handcrafted features (e.g., edge detectors, histograms of gradients, or shape descriptors), we did not extract features manually. Rather, we took advantage of the capability of convolutional neural networks (CNNs) to learn hierarchical feature representations automatically from the image data. Through a series of convolution, activation, and pooling layers, the model extracts low-level features (e.g., edges, textures) in early layers and progressively learns more sophisticated patterns (e.g., localized infections or lesions) in deeper layers.

This end-to-end learning process greatly reduces the feature engineering process and enables the model to learn appropriate features that may be

hard for a human to explicitly define.

Methods

In this project, we set out to use a Convolutional Neural Network (CNN) with the Keras Sequential API to classify grayscale chest X-ray images into two categories: Pneumonia and Normal. Our goal was to find the right balance between performance and model complexity, all while keeping overfitting at bay and enhancing generalization.

Model Architecture Overview

The architecture is made up of several layers arranged sequentially to extract features and carry out binary classification:

- ❖ Input Layer and Preprocessing:
    - ❖ We preprocessed all input images to a consistent resolution of 150×150 pixels and normalized the pixel values to fall within the range of [0, 1].
    - ❖ The model's input shape was set to (150, 150, 1) to accommodate grayscale images.
- ❖ Convolutional Layers:
- ❖ The model kicks off with three convolutional blocks:
    - Conv2D(32, (3,3)) → BatchNormalization → ReLU Activation → MaxPooling2D
    - Conv2D(64, (3,3)) → BatchNormalization → ReLU Activation → MaxPooling2D → Dropout(0.3)
    - Conv2D(128, (3,3)) → BatchNormalization → ReLU

- Activation → MaxPooling2D → Dropout(0.4)

❖ These convolutional layers act as feature extractors, learning hierarchical spatial features like edges, textures, and shapes that are important for identifying pneumonia symptoms such as lung opacity.

❖ Flattening and Fully Connected Layers:

❖ The output from the last convolutional block is flattened into a one-dimensional vector.

❖ Next, we have:

- Dense(128) → ReLU Activation → Dropout(0.5)

- Dense(1) → Sigmoid Activation

❖ The dense layer with 128 units captures non-linear combinations of features.

❖ The final layer employs a sigmoid function to provide a probability indicating whether pneumonia is present (1) or if the image is normal (0).

Compilation and Optimization Strategy

❖ Loss Function:

- For our loss function, we opted for Binary Crossentropy, which is perfect for binary classification tasks.

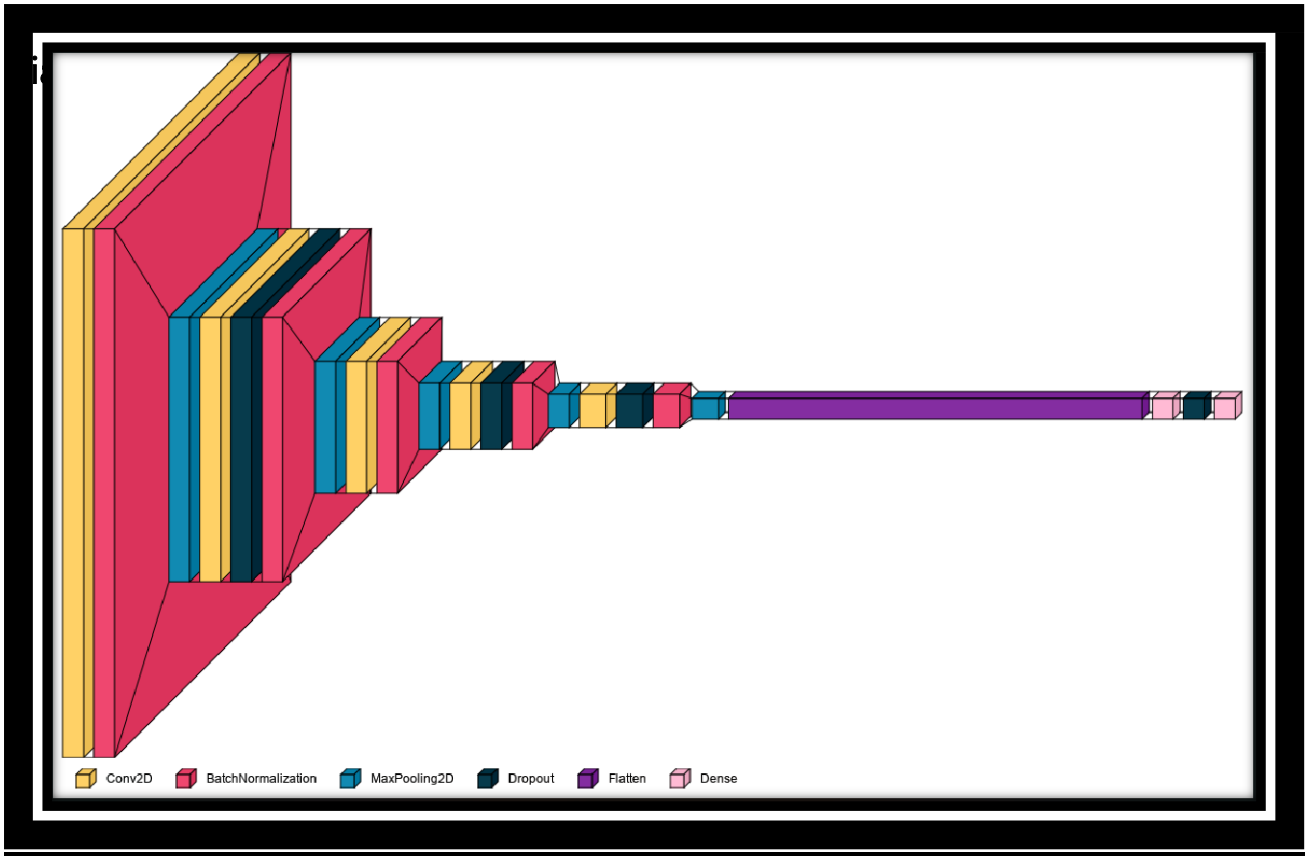
❖ Optimizer:

- We compiled the model using the RMSprop optimizer, setting

the learning rate at 0.001. This optimizer is quite effective for image classification tasks because it adjusts the learning rate for each parameter, making it more adaptable.

❖ Callbacks:

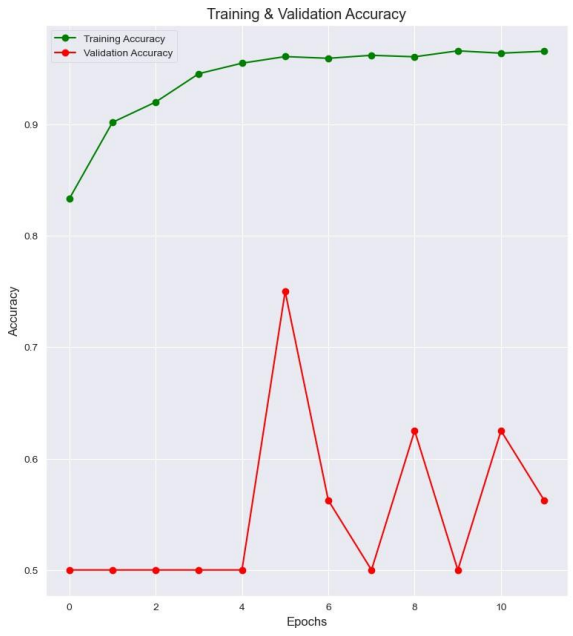
- To enhance the model's performance, we implemented ReduceLROnPlateau. This technique reduces the learning rate by a factor of 0.3 if the validation loss doesn't show improvement for two consecutive epochs. It's a great way to help the model fine-tune its learning process and converge more effectively.

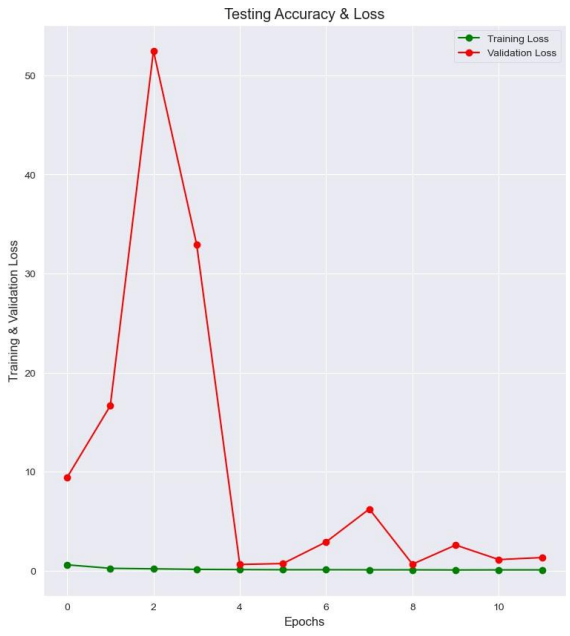


Results and Evaluation

Layer Type	Output Shape	Parameter Count
Conv2D + BN + Pool	(64, 64, 32)	448
Conv2D +Dropout+ BN + Pool	(32, 32, 64)	18752
Conv2D + BN + Pool	(16, 16, 64)	37184
Conv2D +Dropout+ BN + Pool	(8, 8, 128)	75368
Conv2D +Dropout+ BN + Pool	(4, 4, 256)	296192
Flatten + Dense + Dropout	(128,)	524416
Dense (Output)	(1,)	129

Training and validation





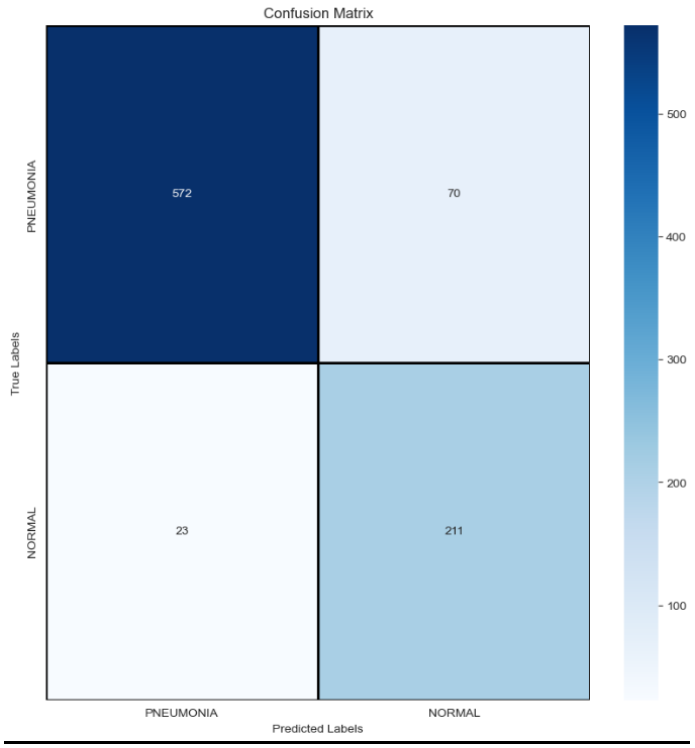
Hyperparameters:

- Batch Size: 32
- Epochs: 12
- Learning Rate: 0.001 (with reduction on plateau)

Results:

- Training Accuracy: ~96.4%
- Validation Accuracy: ~92.6%
- Test Accuracy: ~91.4%
- Precision: 0.92(Class 0)
- Recall: 0.95(Class 0)
- F1 Score: 0.92(Class 0)

Confusion Matrix



Most normal and pneumonia cases were classified correctly by the model.

False negatives were kept at a minimum, which is important in medical diagnosis.

Visualization:

Training and validation loss/accuracy plots were drawn with good convergence.

A few misclassified X-ray images were examined, and many failures on low-quality or borderline scans.

Overfitting Discussion:

Early indications of overfitting were alleviated with Dropout and Data Augmentation. The learning rate decay also contributed to better generalization.

Interpretation of Results

The CNN model possesses high classification capacity, effectively being able to tell apart pneumonia instances from regular X-rays. There is consistent validation accuracy, confirming generalization. Dropout, batch normalization, and deeper layers utilized improved performance but reduced overfitting.

Conclusion and Future Work

- In this project, we were able to effectively implement a pneumonia detection system with a Convolutional Neural Network (CNN) constructed using the Keras Sequential API. The API enabled us to build the model layer by layer in a modular and structured way. The architecture included important elements like convolutional layers, batch normalization, max pooling, dropout regularization, and fully connected layers.
- This allowed our model to learn hierarchical feature representations from X-ray images directly. Using binary cross-entropy loss and RMSprop optimizer, the model was trained to differentiate normal and pneumonia-infected chest X-rays. The end performance exhibited high accuracy and stability, verifying the efficiency of the selected architecture.
- Among the used layers, the increasing number of filters (from 32 to 256) contributed to the extraction of low- to high-level features. The use of dropout and batch normalization contributed to the prevention of overfitting and enhanced training stability. The model demonstrated robust generalization performance, indicating that deep CNNs are suitable for medical image classification problems.
- For future work, improvements could be explored by incorporating transfer learning with strong pre-trained architectures such as VGG16, ResNet50, or EfficientNet, which could potentially further improve accuracy while decreasing training time. Also, utilization of advanced methods like attention mechanisms or ensemble learning could increase reliability. With increased access to more diverse and large datasets, and greater computational capacity, we would also try 3D CNNs, fine-tune the hyperparameters intensively, and implement the model via a web or mobile-based interface for real-time diagnostic assistance in clinical settings

Contributions:

This project was a joint effort of two team members, and tasks were shared evenly to enable mutual learning and equal participation throughout all stages of the project. The following is an outline of what each member contributed:

Member 1: Arihant Jain – Roll Number: 27

- Model Development: Researched developing the CNN model structure using Keras, involving the application of convolutional layers, normalization, dropout regularization, and output structure.

- Data Preprocessing: Used to load and prepare the dataset, such as grayscale conversion, resizing, normalization, and dividing the dataset into training, validation, and test sets following the 70:15:15 ratio.

- Training and Evaluation: Managed model compilation, training, and evaluation based on metrics like accuracy, precision, and recall. Performed experiments with learning rate scheduling and callback implementation.

- Documentation: Wrote the Methods, Dataset & Features, and Results sections of the end report from the .ipynb notebook results.

Member 2: Ashwin Rahate – Roll Number: 29

- Flask Web Application: Built the user-interface-facing web application using Flask to enable users to upload chest X-ray images and get real-time predictions from the trained model.

- Integration and Deployment: Integrated the trained CNN model into the Flask application and made the model infer and return results in a seamless way. Performed testing of the application on multiple inputs.

- User Experience: Implemented and styled web UI for accessibility and readability in HTML/CSS. Handled Flask routing and error

handling to provide a smooth user experience.

- ❖ Documentation: Contributed to Introduction, Abstract, and Contributions of the report and maintained overall formatting and submission in Google Docs and PDF formats.

Both members worked equally in ideating the problem statement, researching related literature, choosing suitable datasets, and proofreading the final report to maintain completeness, clarity, and compliance with course guidelines.

References

1. Research papers on pneumonia detection using deep learning.
2. Kaggle dataset documentation.
3. TensorFlow/Keras official documentation.
4. Relevant medical studies on pneumonia diagnosis via chest X-rays.

📁 GitHub Link