PNEUMONIA PREDICTION AND DECISION SUPPORT SYSTEM

A PROJECT REPORT

Submitted by

KARTHIKEYAN S	**412621243027**
MOHAN A	**412621243033**
NARMADHA V	**412621243034**
SUDHARSHAN M	**412621243050**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

SRI VENKATESWARAA COLLEGE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI-600 025 APRIL/MAY

2025

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"PNEUMONIA PREDICTION AND DECISION SUPPORT SYSTEM" is the bonafide work of "Karthikeyan S(412625104013), Mohan A(412625104015), Narmadha V (412625104018), Sudharshan M(412621243050)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Mr. S.Anburaman	**Mrs. Hemalatha**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Artificial Intelligence and Data Science, Sri Venkateswaraa College of Technology, Vadakal, Sriperumbudur.	Department of Artificial Intelligence and Data Science, Sri Venkateswaraa College of Technology, Vadakal, Sriperumbudur.

This project work was submitted for viva voce held on _____

at Sri Venkateswaraa College of Technology.

Subject Code:AD3811

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We wish to express our heartfelt gratitude to our beloved Chairman **Mr. B. Haribabu** and Vice Chairman **Dr. Srinivasan Haribabu** for providing us this facility to study integrating with values.

We express our sincere gratitude to our CEO **Mr. J. N. Senthil Kanna** for improving the quality education constantly.

We express our gratitude to **Dr. S. Lakshmi**, Principal, Sri Venkateswaraa College of technology for the encouragement given by her to the progress and completion of our project.

We thank **Mr. S. Anburaman**, Head of the Department, Computer Science and Engineering for his effective leadership, encouragement and guidance in the project.

We thank our Project Coordinator, **Mr. M. L. Alphin Ezhil Manuel**, Assistant Professor, Department of Computer Science and Engineering, for his valuable suggestion and motivation throughout our project work.

We thank **Mr. A. Abdul Wahid**, Head, Career Development Services Cell, for his valuable suggestion and motivation in our project work.

We also thank our Project Supervisor, **Mrs. Hemalatha D,** Assistant Professor, Department of Artificial Intelligence and Data Science, for mentoring and helping us immensely in all stages, henceforth leading to the completion of our project work successfully.

Finally, we would thank all teaching and non-teaching staff members of our Department of Computer Science and Engineering, our parents, friends, and well-wishers who have always supported us in all our works.

ABSTRACT

Pneumonia remains a leading cause of respiratory illness and mortality, particularly among the elderly and those with underlying health conditions such as asthma or diabetes. Chest X-rays are widely used for diagnosis, but manual interpretation is time- consuming and requires skilled radiologists. This study proposes a deep learning-based diagnostic framework that automates pneumonia detection using chest radiographs and relevant clinical data. The framework employs transfer learning with pre-trained CNN models—VGG16, DenseNet201, and XceptionNet—to classify X-ray images as normal or pneumonia-infected (bacterial or viral). To enhance model performance, optimization techniques such as Adam and Stochastic Gradient Descent (SGD) are applied and compared. Beyond image data, the model integrates clinical features including white blood cell count (WBC), C-reactive protein (CRP), neutrophil and lymphocyte percentages through a multimodal fusion layer. This fusion enables the network to learn from both visual and physiological cues. Experimental results show that DenseNet201 combined with the Adam optimizer achieves the best test accuracy of 91.49%. Incorporating clinical data significantly improves prediction, especially in borderline cases. The proposed system offers a reliable, efficient, and scalable tool for assisting healthcare professionals in early and accurate pneumonia diagnosis.

Keywords: Pneumonia Detection, Chest X-Ray, Deep Learning, CNN, Transfer Learning, Clinical Data Fusion, DenseNet201, Adam Optimizer.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
CXR	Chest X-ray
GPU	Graphical Processing Unit
MRI	Magnetic Resonance Imaging
RELU	Rectified Linear activation Unity
SCD	Stochastic Gradient Descent
VGG16	Visual Geometry Group16
ML	Machine Learning
LSTM	Long Short-Term Memory

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Pneumonia is one of the most common and potentially life-threatening respiratory diseases in the world. It is characterized by inflammation of the lungs, usually caused by an infection from bacteria, viruses, or fungi. The lungs react to these foreign agents through an inflammatory response that results in the filling of the alveoli (tiny air sacs) and bronchioles with pus or fluid, thereby making breathing painful and reducing oxygen intake. This not only hampers the normal respiration process but also creates a critical condition that requires immediate medical attention The impact of pneumonia is especially severe at both ends of the human age spectrum — infants and elderly individuals. According to global health data, approximately 450 million people contract pneumonia each year, resulting in over 4 million deaths globally. These alarming numbers highlight the urgency of effective and early diagnosis of the disease.

There are three major categories of pneumonia:

Community-Acquired Pneumonia (CAP): Acquired outside hospitals or other healthcare settings.

Hospital-Acquired Pneumonia (HAP): Occurs in hospitalized patients, often due to prolonged exposure to ventilators or immunocompromised conditions.

Ventilator-Associated Pneumonia (VAP): A subset of HAP, affecting patients who require mechanical ventilation.

CAP is more prevalent and often diagnosed in outpatient clinics and emergency settings. In the United States, nearly 1.5 million people are hospitalized every year due to community-acquired pneumonia. Meanwhile, viral pneumonia affects nearly 200 million people annually, equally distributed between adults and children.

Bacterial pneumonia can be caused by:

Typical bacteria such as Streptococcus pneumoniae and Staphylococcus aureus, which are visible on Gram stain and grow on standard culture media.

Atypical bacteria such as Mycoplasma pneumoniae, Chlamydophila pneumoniae, and Legionella pneumophila, which do not show up on Gram stain and require specialized detection methods.

1.1 Global Health Burden

According to the World Health Organization (WHO) and UNICEF, pneumonia remains the leading cause of death in children under five years of age, despite being preventable and treatable. In India, the condition is especially concerning. The country accounted for 158,176 infant deaths in 2016 alone — the highest number globally. By 2030, projections indicate that nearly 11 million children under five could die due to pneumonia if preventive and diagnostic measures are not intensified. The traditional method of diagnosing pneumonia involves clinical examination supported by chest radiography (X-rays). However, challenges such as shortage of radiologists, varying interpretation of X-ray results, and delayed diagnosis often hinder the effectiveness of timely treatment. This problem is particularly acute in remote or underserved regions.

1.2 Need for Automation and AI in Healthcare

Given these challenges, the healthcare industry is turning towards Artificial Intelligence (AI) and Computer-Aided Diagnosis (CAD) tools. These systems have the potential to:

Assist doctors by analyzing medical images efficiently

Reduce diagnostic delays

Increase accuracy through data-driven decision-making

Deep learning, a subfield of AI, has shown exceptional promise in image analysis and medical diagnosis. Unlike traditional machine learning, which requires manual feature extraction, deep learning models learn directly from raw data — mimicking human learning.

1.3 Deep Learning and CNN in Medical Imaging

Deep learning enables machines to perform complex tasks such as object recognition, language translation, and medical image classification. A widely used deep learning technique for image analysis is the Convolutional Neural Network (CNN).

CNNs consist of multiple layers that automatically and adaptively learn spatial hierarchies of features from input images. The core layers in a CNN include:

Convolutional layers, which apply filters to extract important visual features like edges, textures, and patterns.

Activation functions, which introduce non-linearity into the model.

Pooling layers, which reduce the dimensions and computational load (types include Max Pooling and Average Pooling).

Fully connected layers, which compile the extracted features to classify images

Historically, CNNs were used for simple tasks such as recognizing handwritten digits on postal mail. However, their full potential was realized in 2012 when a CNN model named AlexNet achieved a breakthrough in the ImageNet Large Scale Visual Recognition Challenge, dramatically outperforming traditional algorithms.

AlexNet, developed by researchers at the University of Toronto, demonstrated 85% accuracy, while the nearest competitor achieved only 74%. This event marked the beginning of the deep learning revolution in computer vision and transformed CNNs into a standard for image classification tasks.

1.4 Role of CNN in Pneumonia Detection

CNNs are now widely applied in medical imaging to detect diseases such as:

- Lung cancer
- Tuberculosis
- Diabetic retinopathy
- COVID-19
- And most relevantly, Pneumonia

By training a CNN on large datasets of chest X-ray images, the model can learn to distinguish between normal and pneumonia-affected lungs with high accuracy. However, relying solely on visual data can limit the performance of diagnostic models.

To overcome this, the current project introduces a multimodal diagnostic model that integrates chest X-ray images with blood test results. This hybrid approach aims to enhance the model's decision-making process by incorporating clinical data, making it more comprehensive and reliable.

1.5 Objective

The main objective of this project is to design and implement a CNN-based deep learning model capable of detecting pneumonia by analyzing both chest X-ray images and blood test data. The model will employ a weighted fusion layer to combine visual features from CNN and clinical features from blood reports, thereby enhancing both diagnostic speed and accuracy. The solution aims to support radiologists and healthcare workers by providing fast and automated assessments, especially in areas lacking specialized medical personnel.

1.6 Project Report Overview

Chapter 1 introduces the problem, highlights the global burden of pneumonia, discusses the need for deep learning in healthcare, and presents the proposed solution.

Chapter 2 provides a detailed review of existing systems and related works in CNN-based disease detection, including architectures like VGG, ResNet, and DenseNet.

Chapter 3 outlines the methodology used in developing the model, from data preprocessing to architecture design and fusion of features.

Chapter 4 presents the experimental setup, performance evaluation metrics (accuracy, precision, recall, F1-score), and results obtained.

Chapter 5 concludes the report by discussing limitations, real-world applicability, and directions for future work and research extensions.

CHAPTER 2

LITERATURE SURVEY

Medical images utilizing CAD systems has been used for decades ever since the age of artificial intelligence. Since then AI technology has been elevated to machine learning and to deep learning that resulted in an improved acceleration through its performance. This developed technology turns to different techniques like classifying diseases of human-related, animals, and plant diseases. With an increased quantity of datasets, CNN has been the highly recommended technique for identifying and classifying numerous types of medical-related images.

1. Barhoom & Abu Naser (2022) – Diagnosis of Pneumonia Using Deep Learning

Source: International Journal of Academic Engineering Research, Vol.

Overview:

This paper explores the application of Convolutional Neural Networks (CNNs) in diagnosing pneumonia from chest X-ray images. Pneumonia is a serious respiratory condition that requires timely diagnosis, which can be delayed due to radiologist shortages. This study uses deep learning to automate and expedite the detection process.

Methodology:

The authors used a CNN-based model to classify chest X-rays as either normal or pneumonia-affected. They likely utilized a dataset such as ChestX-ray14 or a publicly available Kaggle dataset. Preprocessing steps included resizing, normalization, and data augmentation.

Results:

The CNN achieved high classification accuracy, demonstrating the effectiveness of deep learning in medical imaging diagnostics. The model reduced false positives and negatives, outperforming traditional ML techniques.

Significance:

This work showcases how AI can support healthcare by reducing diagnostic workload and increasing accuracy, especially in under-resourced areas.

2. Albawi, Mohammed & Al-Zawi (2017) – Understanding of a Convolutional Neural Network

Source: 2017 International Conference on Engineering and Technology (ICET),

Overview:

This is a tutorial-style paper providing a conceptual breakdown of CNNs, aimed at beginners. It walks the reader through the architecture and functioning of CNNs used in image classification tasks.

Key Components Explained:

Convolutional Layers: Filters learn local features from the input image.

Pooling Layers: Reduce spatial dimensions and computation.

Activation Functions: ReLU is introduced for non-linearity.

Fully Connected Layers: Integrate features for final classification.

Applications:

Though the paper does not apply CNNs to a specific problem, it lays the groundwork for understanding how CNNs are used in medical and general image recognition tasks.

Significance:

Acts as a reference guide and educational resource for students and early researchers in deep learning.

3. Razavian et al. (2014) – CNN Features Off-the-Shelf: An Astounding Baseline for Recognition

Source: CVPR Workshops, pp. 806–813

Overview:

This paper introduces the idea of using CNNs pretrained on large datasets (e.g., ImageNet) as generic feature extractors for unrelated tasks, without fine-tuning.

Methodology:

Use of AlexNet (trained on ImageNet).

Features are extracted from the fully connected layers and used with simple classifiers like SVM for new tasks (e.g., object or scene recognition).

Findings:

The results show that these pretrained features outperform traditional feature descriptors like SIFT and HOG in many recognition tasks.

Significance:

This was one of the first papers to popularize transfer learning, which is especially valuable in medical imaging where labeled data is scarce.

4. Anthopoulos et al. (2016) – Lung Pattern Classification for Interstitial Lung Diseases Using a Deep CNN

Source: IEEE Transactions on Medical Imaging, Vol. 35(5), pp. 1207–1216

Overview:

This study targets the automatic classification of lung patterns in HRCT (High-Resolution Computed Tomography) scans for diagnosing Interstitial Lung Diseases (ILDs), which are difficult to diagnose due to visual complexity.

Methodology:

A custom CNN was trained on annotated CT scans. Various image preprocessing

and augmentation techniques were used to optimize the model's performance.

Results:

The CNN model was able to distinguish between different ILD patterns (e.g., honeycombing, ground-glass opacities) with high accuracy and consistency.

Significance:

Improves diagnostic reliability and assists radiologists by automating part of a traditionally manual and expert-driven task.

5. Demner-Fushman et al. (2016) – Preparing a Collection of Radiology Examinations for Distribution and Retrieval

Source: J Am Med Inform Assoc, Vol. 23(2), pp. 304–310 Overview:

This paper discusses the development and curation of large datasets of radiological images and their associated reports, which are critical for training and evaluating machine learning models.

Methodology:

Creation of structured datasets like MIMIC-CXR.

De-identification of patient data and metadata tagging.

Mapping radiology reports to corresponding images.

Importance:

The dataset preparation addressed key issues in data sharing, such as privacy, format standardization, and usability for researchers.

Significance:

These datasets are widely used in the research community to train deep learning models for automated disease detection and classification.

6. Das et al. (2013) – ML Approach for Automated Screening of Malaria Parasite Using Light Microscopic Images

Source: Micron 45, pp. 97–106

Overview:

This paper proposes a classical machine learning approach for detecting malaria parasites in blood smear images. It is a pioneering effort before deep learning became dominant.

Methodology:

Microscopic images were enhanced and segmented to isolate infected cells.

Feature extraction techniques were applied.

Classifiers like SVM and ANN were used for detection.

Results:

High classification accuracy was achieved using handcrafted features, which significantly reduced the time required for manual inspection.

Significance:

Demonstrated the potential of AI in parasitology and point-of-care diagnostics, especially in low-resource settings.

7. Glorot & Bengio (2010) – Understanding the Difficulty of Training Deep Feedforward Neural Networks

Source: Proceedings of AISTATS, pp. 249–256

Overview:

This paper addresses the vanishing/exploding gradient problem in deep networks and proposes a new method for weight initialization—Xavier Initialization—to stabilize training.

Key Concepts:

Analysis of activation functions and gradient flow in deep networks.

Comparison of initialization strategies.

Impact:

Their initialization method allowed deeper networks to converge faster and more reliably, making modern deep learning architectures feasible.

Significance:

A foundational work that contributed to the deep learning revolution by solving a major bottleneck in training.

8. Hua et al. (2015) – Computer-Aided Classification of Lung Nodules via Deep Learning

Source: OncoTargets and Therapy, Vol. 8, pp. 506–513

Overview:

This study applies CNNs to classify lung nodules from CT scans as benign or malignant. Early detection of cancerous nodules is critical for improving patient outcomes.

Methodology:

CT scan slices were fed into a CNN.

The model was trained to distinguish between benign and malignant patterns based on texture and shape.

Results:

The CNN-based system achieved high sensitivity and specificity, demonstrating practical applicability in clinical decision support systems.

Significance:

An example of how deep learning can be effectively integrated into radiology workflows to aid early cancer detection.

2.1 Deep Learning

Learning from the Deep Learning point of view is a method of cultivating the actions based on knowledge and experience. Deep Learning is an artificial intelligence and machine learning subgroup which expands the output of countless machine learning applications like AI. Deep architecture technique organized with many processing parts that started in the early 1960s has affected non-linearity and computing power based on the GPU via acceleration, allowing for deeper networks that have better used their resources[9]. The authors in have developed numerous models to confirm an accurate result of the top model in detecting pneumonia. Their study trained various types of CNN models namely AlexNet, LeNet, GoogleNet, ResNet, and VGGNet of using a 226x226 image resolution with a dataset of 26,684 images. The result of their study reached 92% accuracy rate for VGGNet and 50% is the lowest rate attained by the ResNet model[4].

2.2 Quality of the Data

Deep learning is applied to unstructured data like images, video, sound or text. An image is just a blob of pixels, a message is just a blob of text. This data is not organized in a typical, relational database by rows and columns. So, once the data is collected, they are pre-processed into a format the deep learning algorithm can use for the model.

Problems that can arise in data collection such as collected data could be unrelated to the problem statement. Data may be imbalanced. Some classes or categories in the data may have a disproportionately high or low number of corresponding samples[5]. As a result, they risk being under-represented in the model, and the size of a dataset affects the memory and processing required for iterations during training. Normalization reduces the size by reducing the order and magnitude of data.

2.3 Image Augmentation

Deep learning models usually require enormous amounts of data to train for them to correctly function. In the case of CNNs, thousands upon thousands of images are required which, pragmatically speaking, is a difficult prospect. To solve this, a technique named image augmentation is used [3]. Image augmentation virtually increases the size of an existing dataset to a large extent with techniques like standardization of features (pixel values), whitening transforms [5], random rotations and shifts, flipping, re scaling, shearing, zooming, etc. The Author have rescaled, shear, zoomed and flipped horizontally the training image for a wider variety and augmentation of training data.

Image segmentation is the process of partitioning a digital image into multiple image segments, also known as image regions or image objects (sets of pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze [10].

Image segmentation is typically used to locate objects and boundaries (lines,curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

2.4 Data partitioning

Data partitioning in data mining is the division of the whole data available into two or three non-overlapping sets: the training set, the validation set, and the test set. If the data set is very large, often only a portion of it is selected for the partitionsa [2].

Data splitting is the act of partitioning available data into. two portions, usually for cross- validation purposes. One. portion of the data is used to develop a predictive model. and the other to evaluate the model's performance[7]. It takes the chest- X ray images as input and divides the data set into practice (98637) images, 28744 patients), validation (6351 images, 1672 Patients), and testing (420 images, 389 patients) for training the model. Data might be split in as 80-20 or a 67-33 ratio of training, testing data.

Before inputting the images into the network the images have been resized to 224 x 224 and normalized it. The system produces the likelihood of pneumonia including a heat-map that locates the most likely area of pneumonia.

2.5 Data pre-processing

Data partitioning in data mining is the division of the whole data available into two or three non-overlapping sets, the training set, the validation set, and the test set. If the data set is very large, often only a portion of it is selected for the partitions[8]. Partitioning is normally used when the model for the data at hand is being chosen from a broad set of models. To build useful Deep Learning models, the validation error must continue to decrease with the training error. Data Augmentation is a very powerful method of achieving this. The augmented data will represent a more comprehensive set of possible data points, thus minimizing the distance between the training and validation set, as well as any future testing sets.This technique enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data and improve performance. Many other strategies for increasing generalization performance focus on the model’s architecture itself. This has led to a sequence of progressively more

complex architectures from AlexNet, VGG-16, ResNet, Inception-V3 and DenseNet. Functional solutions such as dropout regularization, batch normalization, transfer learning, and pre-training have been developed to try to extend Deep Learning for application on smaller datasets.

2.6 Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other stimulated by the human brain. The convolutional neural network is a feed- forward neural network[2]. It is capable of extracting characterized features and classifies them. The standard convolutional layer acquires image input that creates map features through the kernel. It works by taking each neuron from the preceding layer. Weight sharing reduces the number of parameters. Every connected convolutional layer to non-linear activation layer accelerates CNN's conventional architecture as illustrated in

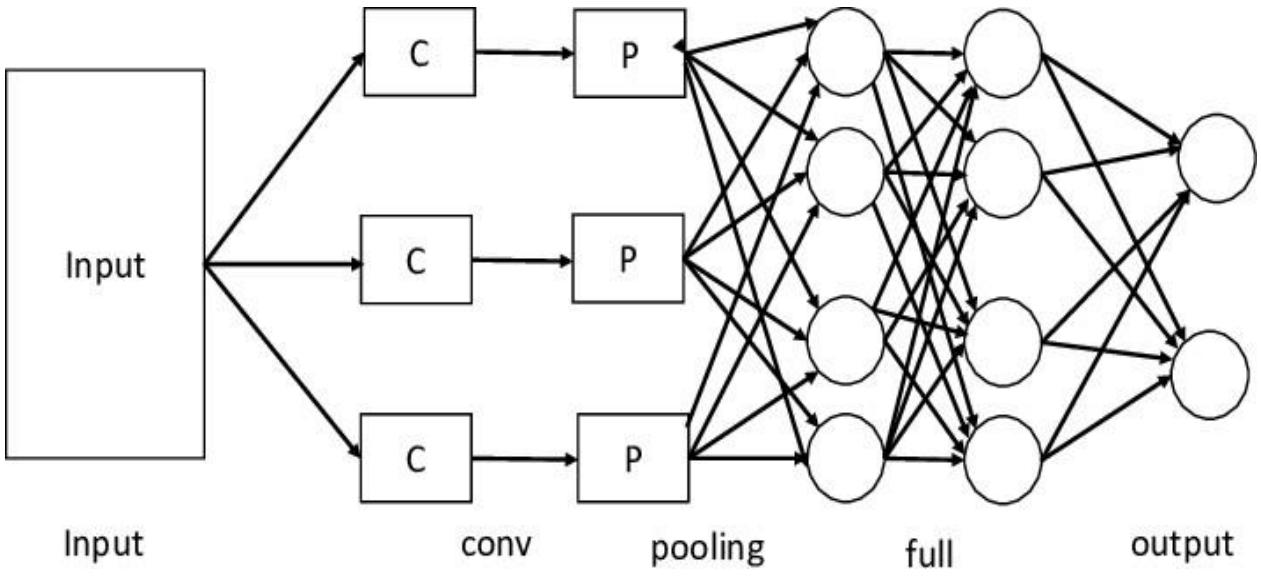

Figure 2.1 Simple Convolutional Neural Network Architecture Diagra

Numerous authors have been using several CNN models in their research. The plain and simple architecture was developed using the LeNet model was introduced in the works of Le Cun et.al. They modified layers, kernels, optimizers, and hyper-parameters to train their model. Their proposed model, composed of several kernels in one convolution learns extra complex functions that compute multiple map features with activation and pooling layers[10]. On ResNet18, Xception, InceptionV3, DenseNet121, and MobileNetv3, Hashmi presented a state- of-the-art based deep learning technique where certain pre-trained convolutional architectures and weighted classifier are utilized for transfer learning utilizing fine-tuning. There were also some partial data augmentation approaches utilized. Finally, using a weighted classifier, data from the Guangzhou Women and Children's Medical Center pneumonia dataset yielded a testing accuracy of 82.43 percent .

The Authors work is based on the well-established pre trained DL model (VGG16) and the attention module.

VGG-16 helps to overcome the over fitting problem as there are limited amount of CXR images for training purpose. The proposed method (also called Attentionbased VGG-16) consists of four main building blocks such as Attention module, Convolution module.

The scale-invariant convolution module captures the interesting clues of the image. The interesting clues are extracted from the mid level layer that is more appropriate to CXR images. However, the features from other layers (higher or lower) are not appropriate to CXR images because such images are neither more general nor more specific.

The standard convolutional layer acquires image input that creates map features through the kernel. It works by taking each neuron from the preceding layer. Increased performance of improved illustration of images because of multiple neuron connections and overlapping with each other [15]. Weight sharing reduces the number of parameters. Every connected convolutional layer to nonlinear activation layer accelerates CNN in more complex functions that may lead to overfitting and pooling is used to overcome the overfitting and also reduce the number of parameters. Fully-connected layers summarize the trained features to provide the classified result.

The limitation is that despite the power and resource complexity of CNNs, they provide in-depth results. At the root of it all, it is just recognizing patterns and details that are so minute and inconspicuous that it goes unnoticed to the human eye. But when it comes to understanding the contents of an image it fails.

These limitations are more than evident when it comes to practical applications. For example, CNNs were widely used to moderate content on social media. But despite the vast resources of images and videos that they were trained on it still is not able to completely block and remove inappropriate content.

Several studies have shown that CNNs trained on ImageNet and other popular datasets fail to detect objects when they see them under different lighting conditions and from new angles. Despite the limits of convolutional neural networks, however, there's no denying that they have caused a revolution in artificial intelligence. Today, CNNs are used in many computer vision applications such as facial recognition, image search, and editing, augmented reality, and more. As advances in convolutional neural networks show, these achievements are remarkable and useful, but still very far from replicating the key components of human intelligence [7].

CNN Models

This sub section describes the use of CNN architectures for modeling pneumonia disease detection.

2.6.1 AlexNet model

AlexNet which was developed by A.Krizhevesky for visual object recognition in 2012 won an ImageNet Challenge. It is the most studied architecture of CNN for classification tasks on images [11]. AlexNet structure has 5 convolutional layers, 2 fully-connected layers with max pooling, and the activation layer that connects to 1000 classes. Its parameters comprise of 61 million values. It has eight layers with learnable parameters [6]. The model consists of five layers with a combination of max pooling followed by 3 fully connected layers and they use Relu activation in each of these layers except the output layer.

ReLU Nonlinearity -- AlexNet uses Rectified Linear Units (ReLU) instead of the tanh function, which was standard at the time. ReLU's advantage is in training time; a CNN using

ReLU was able to reach a 25% error on the CIFAR-10 dataset six times faster than a CNN using tanh function.

Multiple GPUs -- AlexNet allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU. Not only does this mean that a bigger model can be trained, but it also cuts down on the training time.

Overlapping Pooling -- CNNs traditionally "pool" outputs of neighboring groups of neurons with no overlapping. However, when the authors introduced

overlap, they saw a reduction in error by about 0.5% and found that models with overlapping pooling generally find it harder to overfit [12].

2.6.2 GoogLeNet model

GoogleNet architecture's goal is to learn with its parameters reduced. It is inspired based on Inception architecture with a total of 22 convolutional layers using small convolutions and small - batch normalizations. Its parameters comprise of 4 million values[19].

The GoogLeNet architecture proposed by Szegedy is 22-layer deep network consisting of inception modules, instead of uniformly progressive layers. An inception block accommodates a large number of units at each stage by hosting parallel convolution and pooling layers, resulting in an uncontrolled computational complexity because of the increased number of parameters. To control the computational complexity, the GoogLeNet model uses inception blocks with dimension reduction, rather than the naive inception block. The performance of GoogLeNet, in which the inception block was introduced, proves that an optimal sparse architecture built from the available dense building blocks improves the performance of artificial neural networks for computer vision tasks.

2.6.3 LeNet model

LeNet-5 was developed by one of the pioneers of deep learning, Yann LeCun in 1998 in his work titled ‘Gradient-Based Learning Applied to Document Recognition’. LeNet was used in detecting handwritten cheques by banks based on MNIST dataset. Fully connected networks and activation functions were previously known in neural networks. LeNet-5 introduced convolutional and pooling layers. LeNet-5 is believed to be the base for all other.

ConvNets. LeNet architecture comprises of 2 convolutional, pooling layers with a flattened convolutional layer, 2 fully-connected layers, and softmax classifier. LeNet is straightforward and small, making it perfect for teaching the basics of CNNs, it can run on the central processing unit (CPU) even if the system does not have a suitable graphical processing unit (GPU). LeNet's parameter consists of 60 thousand values only [13].

2.6.4 StridedNet model

StridedNet architecture employs stride convolutions for lower volume size, the use of batch normalization, and hyperparameters fine-tuning to improve training accuracy. The utilization of dropout is to help the network generalize and minimize overfitting. The stridedNet parameters are approximately 1 million [20].

2.6.5 ResNet-50 model

ResNet consists of 3 convolutions for a total of 75 layers deep in every convolution block and combines several sizes of convolution kernels with residual blocks during training to lessen training time. Its parameters comprise of 61 million values.

The ResNet-18 model proposed by He et al. It is based on a residual learning framework, which increases the efficiency of deep network training. The residual blocks in the ResNet models facilitate the optimization of the overall network, which in turn improves model accuracy, unlike the original unreferenced mapping in monotonically progressive convolutions. These residuals or "skip connections" perform identity mapping, which neither adds parameters nor increases the computational complexity [17].

receptive field: 3×3 [16]. In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input

channels. The convolution stride is fixed to 1 pixel; the spatial padding of Conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 Conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the Conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.

Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000- way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

All hidden layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalisation (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time[18].

2.7 Activation Function

Activation function Relu is used throughout except for the last layer where Soft max is used for binary classification problem. Activation Function decides whether a neuron should be activated or not. The choice of activation function in the hidden layer will control how well the network model learns the training dataset. The choice of activation function in the output layer will define the type of predictions the model can make. The input is fed to the input layer, the neurons perform a linear transformation on this input using the weights and biases, according to the following equation :

$$x = (\text{weight} * \text{input}) + \text{bias} \quad \text{----------2.1}$$

The rectified linear activation is the default activation when developing multilayer Perceptron and convolutional neural networks. But there are different types of activation function used in model suchs as Sigmoid tanh Activation Function, Leaky ReLU, Softmax Activation Function. Neural network without any activation function would not be able to realize such complex

2.8 Optimizers

Optimizers are algorithms or methods used to minimize an error function(loss function)or to maximize the efficiency of production. Optimizers are mathematical functions which are dependent on model's learnable parameters i.e Weights & Biases [23]. The loss function is the core of deep learning. Simply, it is a method of evaluating how good the model is predicting. Optimizers are methods used to change the attributes of neural network such as weights and learning rate in order to reduce the losses. Categorical crossentropy is a loss function that is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one. Formally, it is designed to quantify the difference between two probability distributions[20]. Gradient descent is a first-order optimization algorithm which is dependent on the first order derivative of a loss function. It calculates which way the weights should be altered so that the function can reach a minima. Through back propagation, the loss is transferred from one layer to another and the model’s parameters also known as weights are modified depending on the losses so that the loss can be minimized [21]. So it has huge drawback of requiring large memory to calculate gradient on the whole dataset. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iteratively based in training data.

2.8.1 Gradient Descent

Gradient descent is an optimization algorithm based on a convex function and tweaks its parameters iteratively to minimize a given function to its local minimum. Gradient Descent iteratively reduces a loss function by moving in the direction opposite to that of steepest ascent. It is dependent on the derivatives of the loss function for finding minima. uses the data of the entire training set to calculate the gradient of the cost function to the parameters which requires large amount of memory and slows down the process [22].

2.8.2 Stochastic Gradient Descent

SCD is a variant of Gradient Descent. It update the model parameters one by one. If the model has 10K dataset SGD will update the model parameters 10k times.Batch gradient descent

performs redundant computations for large datasets, as it recomputes gradients for similar examples before each parameter update.

SGD does away with this redundancy by performing one update at a time. It is therefore usually much faster and can also be used to learn online.SGD performs frequent updates with a high variance that cause the objective function to fluctuate heavily as in Image 1.[21]

2.8.3 Adam(Adaptive Moment Estimation)

Adam optimizer is one of the most popular and famous gradient descent optimization algorithms. It is a method that computes adaptive learning rates for each parameter. It stores both the decaying average of the past gradients, similar to momentum and also the decaying average of the past squared gradients, similar to RMS-Prop and Adadelta. Thus, it combines the advantages of both the methods.[24]

2.9 Model Parameters

Deep learning often deals with large amounts of data. This data is broken down into smaller chunks called batches and fed to the neural networks one-by-one. The batch size is the total number of training examples in a batch. Gradient descent is an iterative process and updating the parameters through back propagation in a single pass is not enough. One epoch is when the entire dataset is passed forward and backward through the neural network once. Epoch is once all images are processed one time individually of forward and backward to the network, then that is one epoch. I like to make sure my definition of epoch is correct.

2.10 Summary

From the study of literature, it is observed that there are different types of architectures in CNN. Each architecture has specific types of layer that increases the complexity and performance of the model.

CHAPTER 3

METHODOLOGY

Deep learning models can interpret medical images like X-ray, MRI scan, CT scan to perform diagnosis.The reuse of a previously learned model on a new problem is known as transfer learning, particularly in deep learning since it can train deep neural networks with a large amount of data. In the proposed method, Chest X-Ray images are collected, and the images in the dataset are pre-processed in various CNN architectures that have been utilized in this study to detect Pneumonia or Normal X-rays.

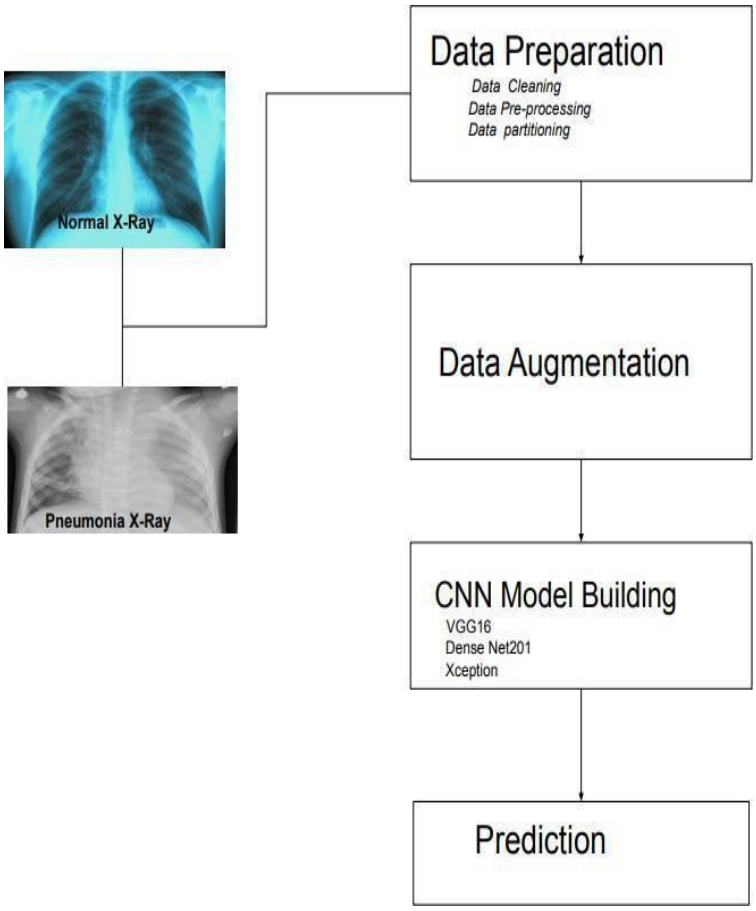

Figure 3.1 Proposed method of Pneumonia detection-Block Diagram

3.1 System Specification

Following are the hardware and software specifications using which this project was developed.

3.1.1 Software Requirements

Environment : VS Code

Programming Language: Python 3.7

Processing Unit : Tensor flow Processing Unit

3.1.2 Hardware Requirements

RAM: 16 GB

Processor: Intel i5 Hexa Core processors

3.2 Dataset Preparation

Dataset used was Mendeley Dataset V2 having 5856 Chest X-ray images from both Pneumonia and Normal class. In the Pneumonia dataset, both the combination of viral and bacterial pneumonia x-ray images are stored. Before proceeding with the proposed method, the pictures were partitioned into training and testing sets.

In this work, the dataset extracted by Mendeley website having 5841 Chest X-ray images of both Pneumonia and Normal class had been used. In the Pneumonia dataset, both the combination of Viral and Bacterial Pneumonia x-ray images are stored. The images were partitioned into training and testing sets in the ratio of 66/33. A total of 1929 pictures were set apart for testing and 3912 images were used for training. All the images in the dataset are in 512*512 pixels in size. The number of images in training and testing in both classes is listed in Table 3.1.

Table 3.1 Dataset Statistics

Data	Normal	Viral Pneumonia	Bacterial Pneumonia
Training Set	1055	1000	1857
Testing Set	520	494	915
Total	1575	1494	2772

The dataset is organized into 3 section train, test, validation and contains sub folders for each image category, Pneumonia and Normal. Web scraping is used to extract the minor pneumonia chest x -ray images from the websites for the better performance of the model. The data is mounted into the Google drive for fast processing of the large volume of data. Once the dataset is uploaded into the Google drive there is no need to spend networkdata every time.

Chest X-ray images were selected from retrospective cohorts of pediatric patients from Guangzhou Women and Children's Medical Center, Guangzhou. All chest X-ray imaging was performed as part of patient routine clinical care. For the analysis of chest x-ray images, all chest radiographs were initially screened for quality control by removing all low quality or unreadable scans. The diagnoses for the images were then graded by two expert physicians before being cleared for training the AI system. In order to account for any grading errors, the evaluation set was also checked by a third expert.

3.3 Data Augmentation

Data augmentation techniques are used to increase the input images and also resize the images of dataset from 512*512 converted into 224*224 pixels to reduce the heavy computation and for faster processing.The resized images are then transformed from RGB to gray scale images.

data generator class is designed to provide real-time data augmentation and it requires lower memory usage. It generates augmented images on the fly while the model is still in the training stage. Image data generator class ensures that the model receives new variations of the images at each epoch, but it only returns the transformed images and does not add it to the original corpus .

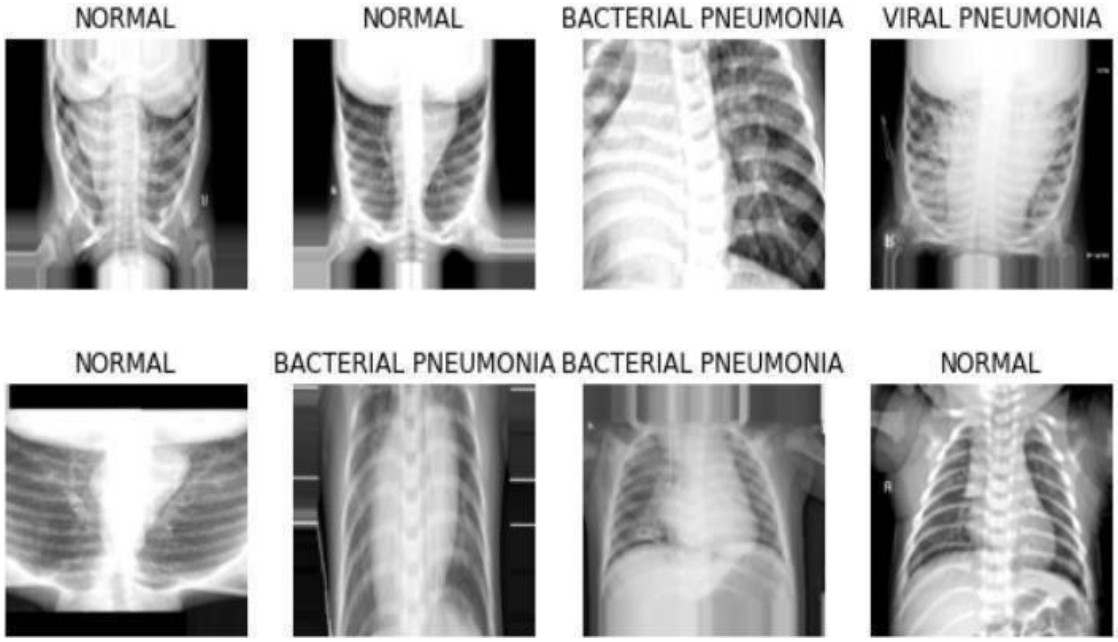

Figure 3.2 Visualization of data augmentation techniques of Chest X ray

3.4 CNN Overview

A convolutional neural network is a network architecture for deep learning which learns directly from data, eliminating the need for manual feature extraction. CNNs are particularly useful for finding patterns in images to recognize objects, faces, and scenes. They can also be quite effective for classifying non-image data such as audio, time series, and signal data. CNN can have tens or hundreds of layers that each learn to detect different features of an image. Filters are applied to each training image at different resolutions, and the output of each one.

3.5 Transfer Learning

Transfer learning is the reuse of a pre-trained model on a new problem. It's currently very popular in deep learning because it can train deep neural networks with comparatively little data. This is very useful in the data science field since most real-world problems typically do not have millions of labeled data points to train such complex models. Transfer learning (TL) is a research problem that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.

3.6 Visual Geometry Group16 Architecture

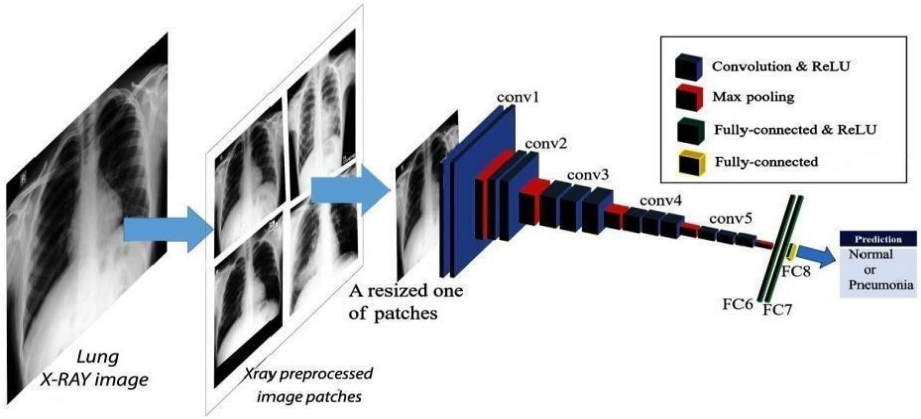

Figure 3.3 VGG16 Architecture Diagram

VGG16 with weights trained on the ImageNet database is used for extracting features from the images. ImageNet database consists of more than 14 million images categorized into 1000 classes. Since pre-trained models like VGG16 have already learned to extract features from the images, these models have shown magnificent performance when applied on datasets of similar domains. The network consisted of 16-layers with 3 convolutional layers, which are followed by 5 dense layers and the output layer. For the dimensionality reduction, a series of max-pooling layers have been introduced in the architecture VGG16 is having a large number of hyper-parameters. It has convolution layers of 3x3 filter size with a stride of 1 and always uses same padding and max pool layer of 2x2 filter size with stride of 2.

Convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 fully connected layers followed by a soft max

function for output. The 16 in VGG16 refers to the 16 layers that have weights. This network is a pretty largenetwork and it has about 138 million parameters. Following each convolutional layer, a batch normalization layer is used to normalize the data that comes from the convolutional layer, allowing for less complex processing after the convolutional layer. A pooling layer is utilized with each convolutional layer. The purpose of the pooling layers is to make the feature map smaller. There are several types of pooling layers, with max pooling being the most effective and widely utilized. This pooling layer takes the maximum feature portion from the images and makes a smaller representation. They are often applied after the convolutional layer in order to decrease the complexity of the calculation.

3.6.1 Input Layer

The input layer in CNN should contain image data. Image data is represented by a three-dimensional matrix and it has to be reshaped into a single column. The

pre-processed grey images of the dataset have a size of 224 * 224 pixelsand the images are fed into the model layers.

3.6.2 Convolution Layer

The main task of these layers is to find out the similarity or connection of features from the previous layer. These layers then make a feature map using the edges of the features. In this proposed method, six convolutional 2D layers are used along with some filters to get the feature maps. These feature maps represent features or edges from the images using the filters to assemble them. Every filter is trained to obtain the edges or features from the Chest X-ray images from the dataset. The output feature map is obtainedusing the following equation

3.6.3 Filter layer

Filters detect patterns such as edges in an image by detecting the changes in intensity values of the image. In terms of an image, a high-frequency image is the one where the intensity of the pixels changes by a large amount, whereas a low-frequency image is the one where the intensity is almost uniform. Usually, an image has both high and low frequency components. The high-frequency components correspond to the edges of an object because at the edges the rate of change of intensity of pixel values is high.

3.6.4 Rectified linear activation unit

After each convolution operation, RELU a non-linear activation function is applied. This operation is applied to each pixel and replaces all the negative pixel values in the feature map with zero. Usually, the image is highly non-linear, which

means it has varied pixel values. In this scenario it is very difficult for an algorithm to make correct predictions. RELU activation function is applied in these cases to decrease the non-linearity and make the job easier. Therefore this layer helps in the detection of features, decreasing the non-linearity of the image, converting negative pixels to zero which also allows detecting the variations of features.

3.6.5 Dense Layers

Dense Layer is simple layer of neurons in which each neuron receives input from all the neurons of previous layer, thus called as dense. Dense Layer is used to classify image based on output from convolutional layers. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

3.6.6 Output Layer

For classification purposes, a soft max activation function is employed in the output layer of the algorithm. It is referred to as the ReLu activation function. It is mainly used for finding the probability prediction using the values from 0 to

1. The classifier gives output either as "0" (Non- pneumonia) or "1" (Pneumonia)

3.6.7 Model Parameters

The total parameters of the VGG16 architecture are 134,268,738 and all of them are learnable parameters. All the Max-Pooling layers lack parameters, and the batch normalization layers do not have any parameters that can be learned .

3.7 DenseNet201

DenseNet201, which is a short form of the Dense Convolutional Network, needs less numbers of parameters than a conventional CNN DenseNet has four different

variants, DenseNet121, DenseNet169, DenseNet201,The feature maps of all the preceding layers are used as inputs. The input of a layer inside DenseNet201 is the concatenation of feature maps from previous layers. DenseNet201 require fewer parameters than an equivalent traditional CNN, as there is no need to learn redundant feature maps. DenseNet201 architecture is shown in figure

3.4. Dense nets layers are very narrow (e.g. 12 filters), and they just add a small set of new feature- maps. Another problem with very deep networks is the problems to train. DenseNet201 solves this issue since each layer has direct access to the gradients from the loss function and the original input image

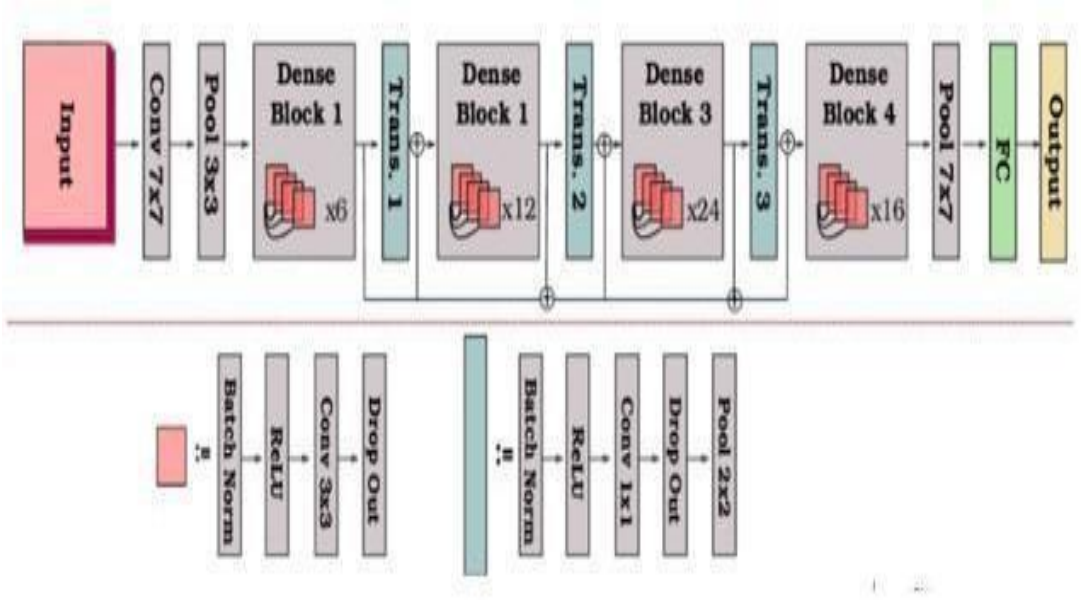

Figure 3.4 DenseNet201 Architecture Diagram [Source[2]]

3.7.1Transition Layers

Since each dense block will increase the number of channels, adding too many layers will lead to an excessively complex model. A transition layer is used to control the complexity of the model. It reduces the number of channels by using a 1×1 convolutional layer. It uses the transition layer to halve the height and width and halve the number of channels, further reducing the complexity of the model.

3.8 XceptionNet

Xception by Google, stands for Extreme version of Inception, is reviewed with a modified depthwise separable convolution, it is even better than Inception- v3.The Xception module has 3 main parts, the Entry flow, the Middle flow and the Exit flow as shown in Fig 3.5. The entry flow has two blocks of convolutional layers followed by a ReLU activation. There are also various Separable convolutional layers. There are also Max Pooling layers. When the strides are different than one, the strides are also mentioned.

3.9 Separable Layers

Separable convolutions consist of first performing a depth-wise spatial convolution (which acts on each input channel separately) followed by a pointwise convolution which mixes the resulting output channels

Traditional Convolutional layer = 3x3x3x64 = 1,728

Separable Convolutional layer =(3x3x1x3) + (1x1x3x64)=27+192 = 219

3.9.1 Model Parameter

The total parameters of the DenseNet201 architecture are 20,865,578 In this 4,098 of them are learnable parameters and 20,861,480 non-learnable parameters.

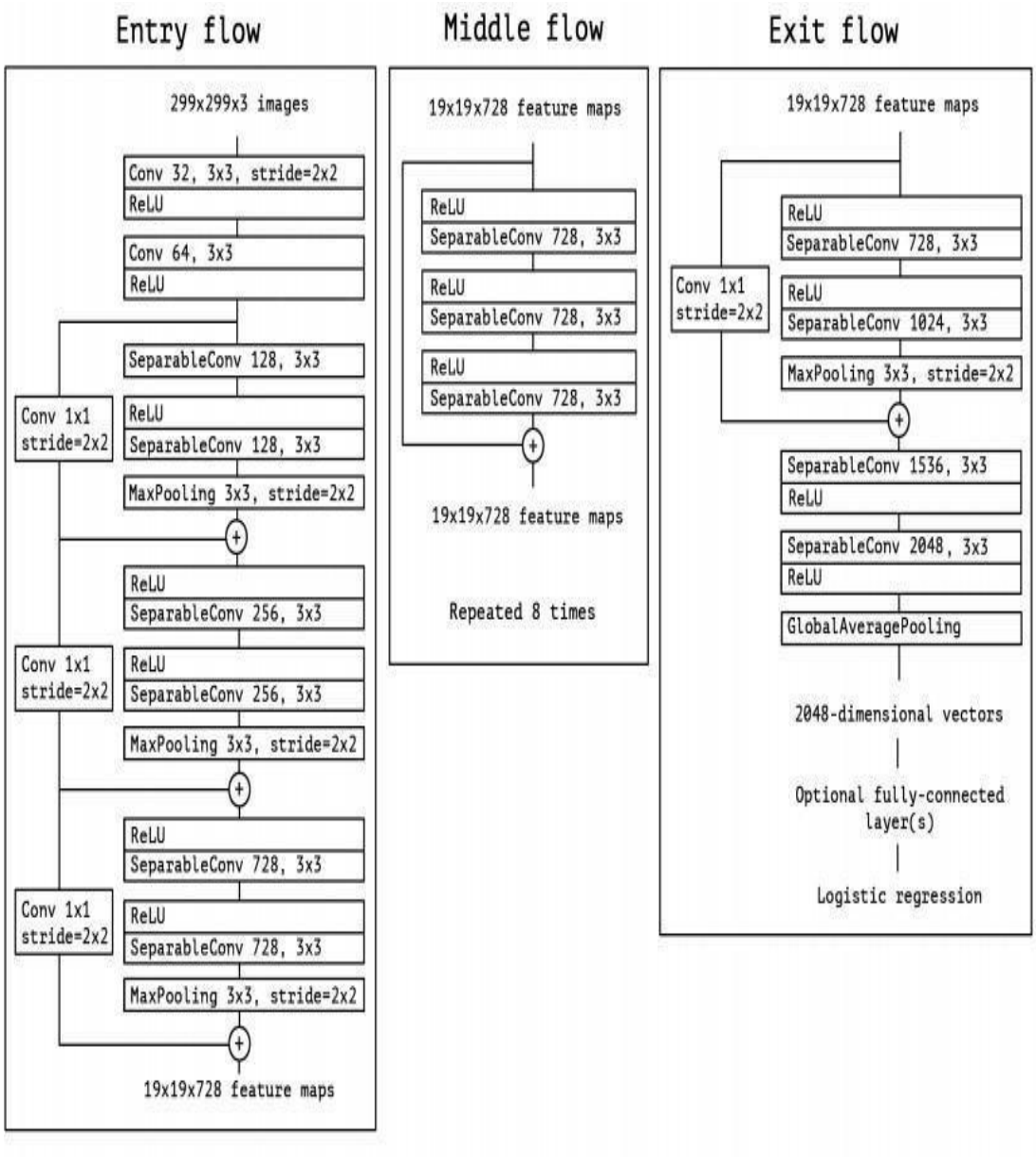

Figure 3.10 XceptionNet Architecture

3.11 Feature Extraction

Feature extraction is a fundamental component of the proposed pneumonia detection system, as it enables the model to learn and represent critical patterns from both visual and clinical data sources. In this work, a dual-branch architecture is employed to extract features independently from chest X-ray images and blood test parameters.

For image-based feature extraction, pre-trained convolutional neural network (CNN) models such as VGG16, DenseNet201, and XceptionNet are utilized. These models are chosen for their proven performance in medical imaging tasks. The CNNs process the chest X-ray images through a series of convolutional, pooling, and activation layers to learn spatial hierarchies and identify significant patterns such as lung opacity, inflammation, and structural irregularities. The resulting deep features represent the high-level characteristics of the input images that are relevant for pneumonia detection.

In parallel, clinical data including white blood cell (WBC) count, C-reactive protein (CRP) levels, neutrophil percentage, and lymphocyte percentage are passed through a fully connected feed-forward neural network. This branch extracts meaningful relationships and trends from the blood parameters, which are often indicative of infection or inflammation.

After independent feature extraction from both modalities, the outputs from the image and clinical branches are combined using a fusion layer. This fusion layer employs a weighted averaging mechanism, which allows the model to dynamically balance and integrate information from both sources during training. This strategy ensures that the model leverages the most informative aspects of each modality, enhancing the overall prediction accuracy and robustness of the system.This comprehensive feature extraction methodology allows the system to gain a deeper understanding of the patient's condition

3.11.1 Fusion Layer Integration

Fusion layer integration is a critical component of the proposed multimodal pneumonia detection system, designed to combine the strengths of both visual and clinical data for improved diagnostic accuracy. In this work, two separate feature extraction pipelines are employed: one for processing chest X-ray images using a pre-trained Convolutional Neural Network (CNN) model, such as DenseNet201, and another for analyzing patient blood test parameters through a fully connected neural network. These parallel branches extract deep features from their respective data sources, representing spatial lung patterns and physiological indicators related to pneumonia. Once the features from both modalities are extracted, they are passed into a fusion layer where they are combined using a weighted averaging mechanism. The model is trained to learn and adjust these weights, determining the relative contribution of image- based and clinical features in the final prediction. This dynamic fusion allows the system to adapt to varying input quality or relevance, giving more emphasis to the more informative modality in each case. For example, when X-ray features are not clearly distinguishable, the model can rely more heavily on clinical indicators such as white blood cell count (WBC), C-reactive protein (CRP), neutrophils, and lymphocytes. The fused feature vector is then passed through additional fully connected layers followed by a softmax activation function to classify the input as Normal, Bacterial Pneumonia, or Viral Pneumonia. This fusion strategy effectively simulates the clinical decision-making process, where healthcare professionals integrate radiological findings with laboratory results to reach a conclusive diagnosis. By leveraging this approach, the model enhances its ability to make accurate predictions, especially in complex borderline.

3.12 Training

For training the CNN model, optimizer and early stopping are used. The parameters such as softmax Activation function, Adam optimizer, 5 epochs, 100 steps per epoch, Batch size of 32 and categorical cross entropy loss function were used in training the model. Early stopping is an optimization technique used to reduce over fitting without compromising on model accuracy. The main idea behind early stopping is to stop training before a model starts to over fit.

This proposed model uses an Adam optimizer, and this optimization is a stochastic gradient descent method that is based on adaptive estimation of the first- order and second-order moments. An epoch is a term used in deep learning and indicates the number of passes of the entire training dataset, the CNN algorithm has completed. The proposed system is trained up to 5 epochs for various models such as VGG16, DenseNet201, and XceptionNet. The best accuracy is found after 5 epochs. There is no variation of bias got after 5 epochs.

Batch size plays a major role in the CNN-based model. In this proposed method, the batch size is 32. The best accuracy was found using a batch size of 32. The batch mainly divides the training dataset and train model batch-wise respectively. Over fitting was avoided by using early stopping and regularizers in the first dense layer.

Thus, initially Dataset was splitted in the ratio of 67% and 33%. Augmentation technique is used to enhance the Chest X-Ray images to avoid overfitting. Three models such as VGG16, DenseNet201, XceptionNet Models were used to train and test the dataset where finally, the model performance evaluvatedusing graph.

CHAPTER 4

RESULTS

The chest- X ray images are divided into Training set 3912 images, and testing set 1929. the images are resized to 224*244 size normalized before feeding them into the CNN networks. The CNN models used are VGG16, DenseNet201, XceptionNet. In figure 4.1 - 4.3, accuracy and loss graph of VGG16, DenseNet201, XceptionNet Models are shown.

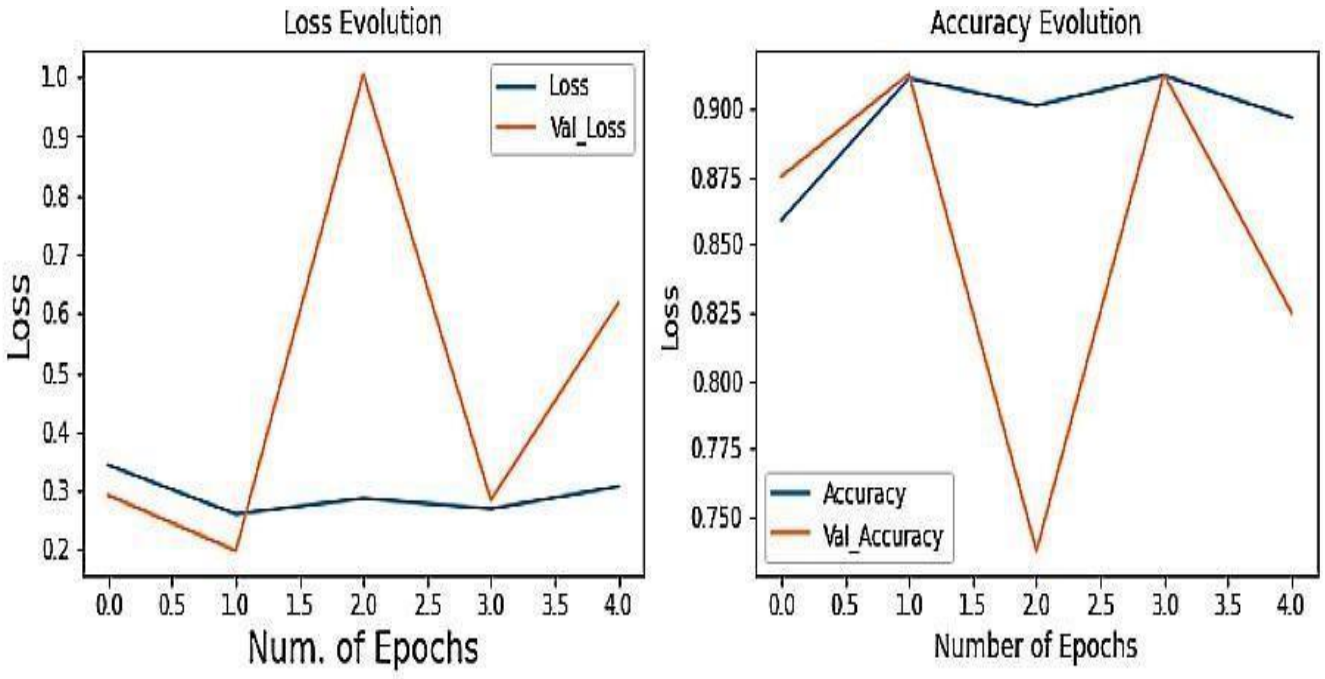

Figure 4.1 Accuracy and loss graph of VGG16

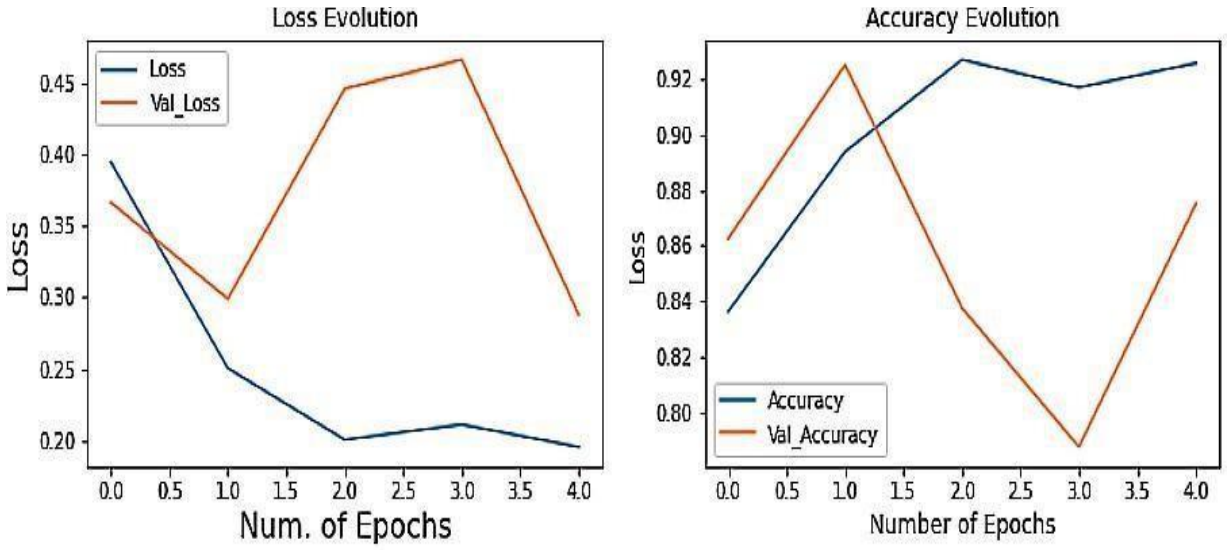

Figure 4.2 Accuracy and loss graph of DenseNet201

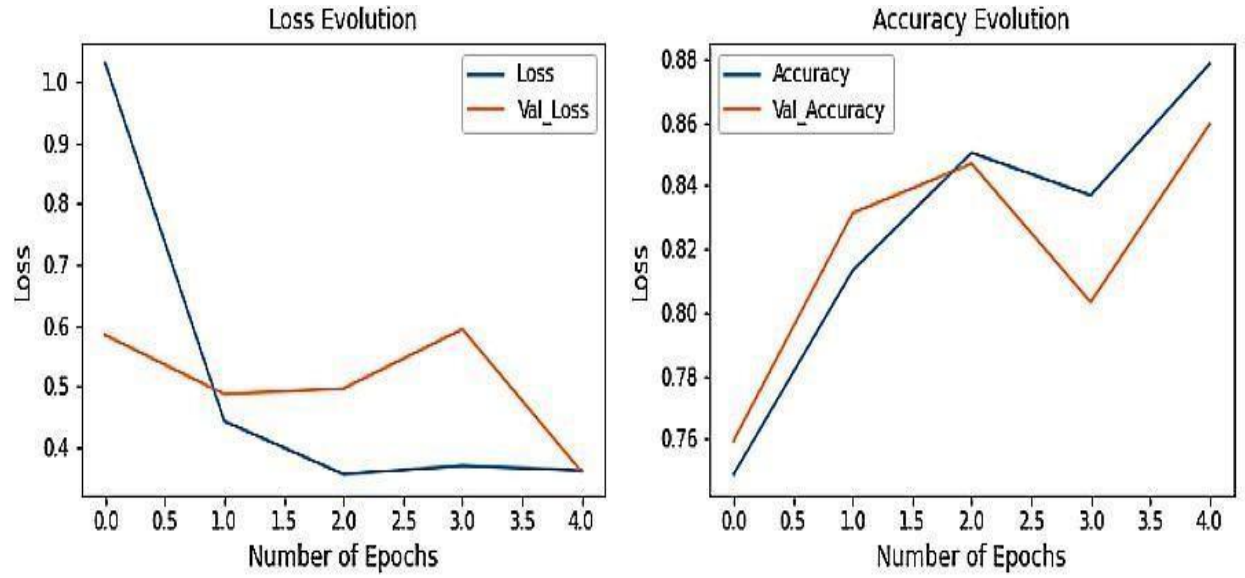

Figure 4.3 The accuracy and loss graph of XecptionNet

TP (True Positive)refers to Pneumonia Image being classified as Pneumonia. TN (True Negative) refers to Normal Image b
FP (False Positive)refers to Normal Image as Pneumonia. FN (False Negative)refers to Pneumonia Image Normal.

Table 4.1 provides the value of test accuracy in VGG16, DenseNet201 Models. These are trained with 5 epochs and batch size of 32.

Table 4.1 Performance Evaluation

Model Name	Test Accuracy %
VGG16	84.70
XceptionNet	83.37
DenseNet201	91.49

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

This study describes a CNN-based model aiming to diagnose pneumonia on chest X-ray images. Data augmentation technique was used to enhance the chest X - ray images in the dataset. The three CNN architecture such as VGG16, DenseNet201, XceptionNet were used to extract the features from original images or previous feature maps, which contained only six layers combining ReLU activation function, drop operation, and max pooling layers. These convolutional neural networks models were successfully implemented by employing various methods of parameter tuning like adding dropout, changing learning rates, changing the batch size, number of epochs, adding more complex fully connected layers and changing various stochastic gradient optimizers.

5.2 Future Work

In the future, it is proposed to use a Raspberry Pi Board with a camera to develop a Pneumonia Classification expert system using Edge Machine Learning system. The web application will display the results of the Pneumonia Classification on the web page. This system will serve AI-based pneumonia classifier using Edge Impulse model.

APPENDIX A : SAMPLE CODE

APP.PY

```
from flask import Flask, render_template, request, send_file

from tensorflow.keras.models import load_model

from tensorflow.keras.utils import load_img

from tensorflow.keras.preprocessing.image import img_to_array

import numpy as np

import joblib

import os

import time


app = Flask(__name__)

app.config['UPLOAD_FOLDER'] = 'static'


# Load models

xray_model = load_model('models/pneu_cnn_model.h5')

fusion_model = load_model('models/fusion_model.h5')

blood_model = joblib.load('models/pneumonia_model.pkl')

scaler = joblib.load('models/pneumonia_scaler.pkl')

label_encoder = joblib.load('models/pneumonia_label_encoder.pkl')


@app.route('/', methods=['GET', 'POST'])
```

```
def index():
    blood_test_prediction = None
    xray_prediction = None
    fusion_prediction = None
    image_path = None
    blood_test_results = None

    fields = {
        'wbc_count': 'White Blood Cell (WBC) Count (cells/mL)',
        'lymphocyte': 'Lymphocytes (cells/mL)',
        'platelets': 'Platelets (cells/mL)',
        'hemoglobin': 'Hemoglobin (g/dL)',
        'neutrophils': 'Neutrophils (%)',
        'crp': 'C-Reactive Protein (CRP) (mg/L)',
        'procalcitonin': 'Procalcitonin (PCT) ng/mL'
    }

    if request.method == 'POST':
        submit_type = request.form.get('submit_type')
        try:
            blood_values = [float(request.form.get(key)) for key in fields.keys()]
        except:
```

```
        blood_values = None

    # Blood test only
    if submit_type == 'blood_test' and blood_values:
        features_scaled = scaler.transform([blood_values])
        prediction = blood_model.predict(features_scaled)
        blood_test_prediction = label_encoder.inverse_transform(prediction)[0]
        blood_test_results = dict(zip(fields.values(), blood_values))

    # X-ray only
    elif submit_type == 'xray_upload' and 'xray_image' in request.files:
        xray_file = request.files['xray_image']
        if xray_file and xray_file.filename != '':
            image_path = os.path.join(app.config['UPLOAD_FOLDER'],
xray_file.filename)
            xray_file.save(image_path)

            img = load_img(image_path, target_size=(500, 500),
color_mode='grayscale')
            img_array = img_to_array(img) / 255.0
            img_array = np.expand_dims(img_array, axis=0)
            pred = xray_model.predict(img_array)[0][0]
```

```
            xray_prediction = 'Positive' if pred >= 0.5 else 'Negative'


        # Fusion prediction
        elif submit_type == 'fusion_predict' and 'xray_image' in request.files and blood_values:
            xray_file = request.files['xray_image']
            if xray_file and xray_file.filename != '':
                image_path = os.path.join(app.config['UPLOAD_FOLDER'], xray_file.filename)
                xray_file.save(image_path)


                img = load_img(image_path, target_size=(500, 500), color_mode='grayscale')
                img_array = img_to_array(img) / 255.0
                img_array = np.expand_dims(img_array, axis=0)
                blood_input = np.array(blood_values).reshape(1, 7)
                fusion_pred = fusion_model.predict([img_array, blood_input])[0][0]
                fusion_prediction = 'Positive' if fusion_pred >= 0.5 else 'Negative'


    return render_template('index.html',
                           blood_test_prediction=blood_test_prediction,
                           xray_prediction=xray_prediction,
                           fusion_prediction=fusion_prediction,
```

```
                imagePath=image_path,
                fields=fields,
                blood_test_results=blood_test_results)

@app.route('/download_report/<result_type>/<prediction>', methods=['GET'])
def download_report(result_type, prediction):
    # Generate report content
    report_content = f"Pneumonia Detection Report\n\nMethod:
{result_type}\nPrediction: {prediction}\nDate: {time.ctime()}"
    report_filename = f"Pneumonia_Report_{result_type}_{str(int(time.time()))}.txt"

    report_path = os.path.join('static', report_filename)
    with open(report_path, 'w') as report_file:
        report_file.write(report_content)

    return send_file(report_path, as_attachment=True)

if __name__ == '__main__':
    app.run(debug=True)
```

BLOOD_TEST.PY

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
import joblib

df = pd.read_csv(r"C:\Users\HP\Downloads\pneumonia_blood_data_large.csv")

df.dropna(inplace=True)

label_encoder = LabelEncoder()
df['target'] = label_encoder.fit_transform(df['target'])

X = df.drop('target', axis=1)
y = df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

classification_rep = classification_report(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

joblib.dump(model, 'models/pneumonia_model.pkl')
joblib.dump(scaler, 'models/pneumonia_scaler.pkl')
joblib.dump(label_encoder, 'models/pneumonia_label_encoder.pkl')
```

SHAP.PY

```
import os
import cv2
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array

def ensure_model_ready(model, sample_input):
    if not model.built:
        model.build(sample_input.shape)
    if not hasattr(model, 'optimizer') or model.optimizer is None:
        model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

```
    return model

def get_last_conv_layer_name(model):
    for layer in reversed(model.layers):
        if isinstance(layer, tf.keras.layers.Conv2D):
            return layer.name
    raise ValueError("No Conv2D layer found in the model.")

def get_gradcam_heatmap(model, image_array, layer_name):
    if not isinstance(image_array, tf.Tensor):
        image_array = tf.convert_to_tensor(image_array)
    last_conv_layer = model.get_layer(layer_name)
    grad_model = tf.keras.Model(inputs=model.input, outputs=[last_conv_layer.output,
model.output])
    with tf.GradientTape() as tape:
        tape.watch(image_array)
        conv_output, predictions = grad_model(image_array)
        if predictions.shape[-1] == 1:
            pred_index = 0
            class_channel = predictions[:, pred_index]
        else:
            pred_index = tf.argmax(predictions[0])
            class_channel = predictions[:, pred_index]
    grads = tape.gradient(class_channel, conv_output)
    pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))
    conv_output = conv_output[0]
    for i in range(pooled_grads.shape[-1]):
        conv_output[:, :, i] *= pooled_grads[i]
```

```
    heatmap = tf.reduce_mean(conv_output, axis=-1)
    heatmap = tf.maximum(heatmap, 0) / tf.reduce_max(heatmap)
    return heatmap.numpy()

def overlay_heatmap_on_image(image_path, heatmap, output_path, alpha=0.5):
    original_img = cv2.imread(image_path)
    if original_img is None:
        raise FileNotFoundError(f"Image not found at: {image_path}")
    heatmap = cv2.resize(heatmap, (original_img.shape[1], original_img.shape[0]))
    heatmap = np.uint8(255 * heatmap)
    heatmap_color = cv2.applyColorMap(heatmap, cv2.COLORMAP_JET)
    overlay = cv2.addWeighted(original_img, 1 - alpha, heatmap_color, alpha, 0)
    os.makedirs(output_path, exist_ok=True)
    output_file = os.path.join(output_path, 'gradcam_result.jpg')
    success = cv2.imwrite(output_file, overlay)
    if not success:
        raise Exception("Failed to save the heatmap image.")
    return output_file

if __name__== "__main__":
    model_path = "models/pneu_cnn_model.h5"
    image_path = r"C:\Users\HP\Downloads\FinalYearProject-master\FinalYearProject-master\static\smpl2.jpeg"
    output_path = r"C:\Users\HP\Downloads\FinalYearProject-master\FinalYearProject-master\static"

    try:
        model = tf.keras.models.load_model(model_path)
```

```
        img = tf.keras.preprocessing.image.load_img(image_path, target_size=(500, 500), color_mode='grayscale')
        img_array = tf.keras.preprocessing.image.img_to_array(img)
        img_array = img_array / 255.0
        img_array = np.expand_dims(img_array, axis=0)
        last_conv = get_last_conv_layer_name(model)
        heatmap = get_gradcam_heatmap(model, img_array, layer_name=last_conv)
        output_file = overlay_heatmap_on_image(image_path, heatmap, output_path)
    except Exception as e:
        import traceback
        traceback.print_exc()
        raise
```

APPENDIX B : SNAPSHOTS

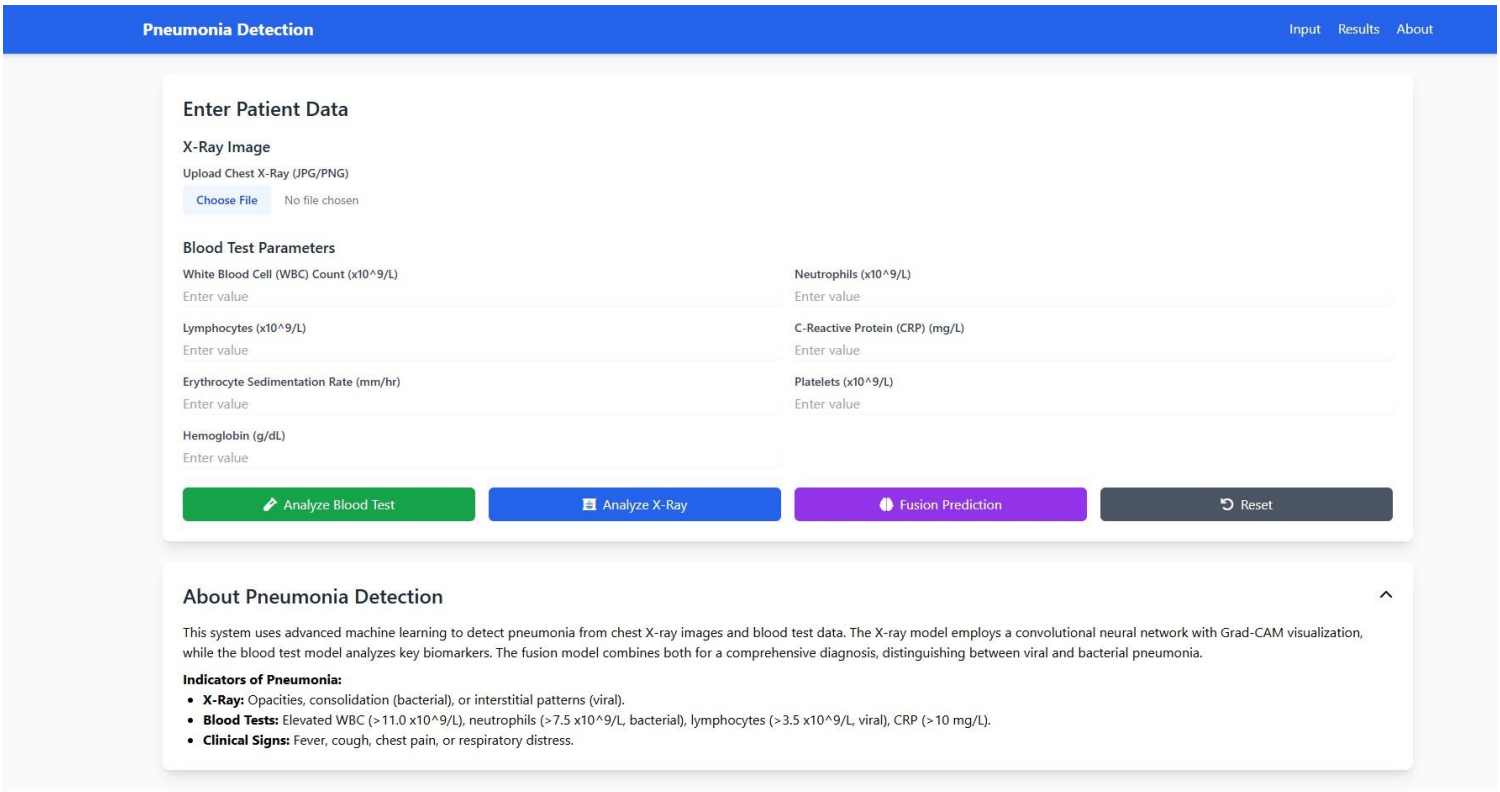

FIGURE B.1: WEB APPLICATION INTERFACE

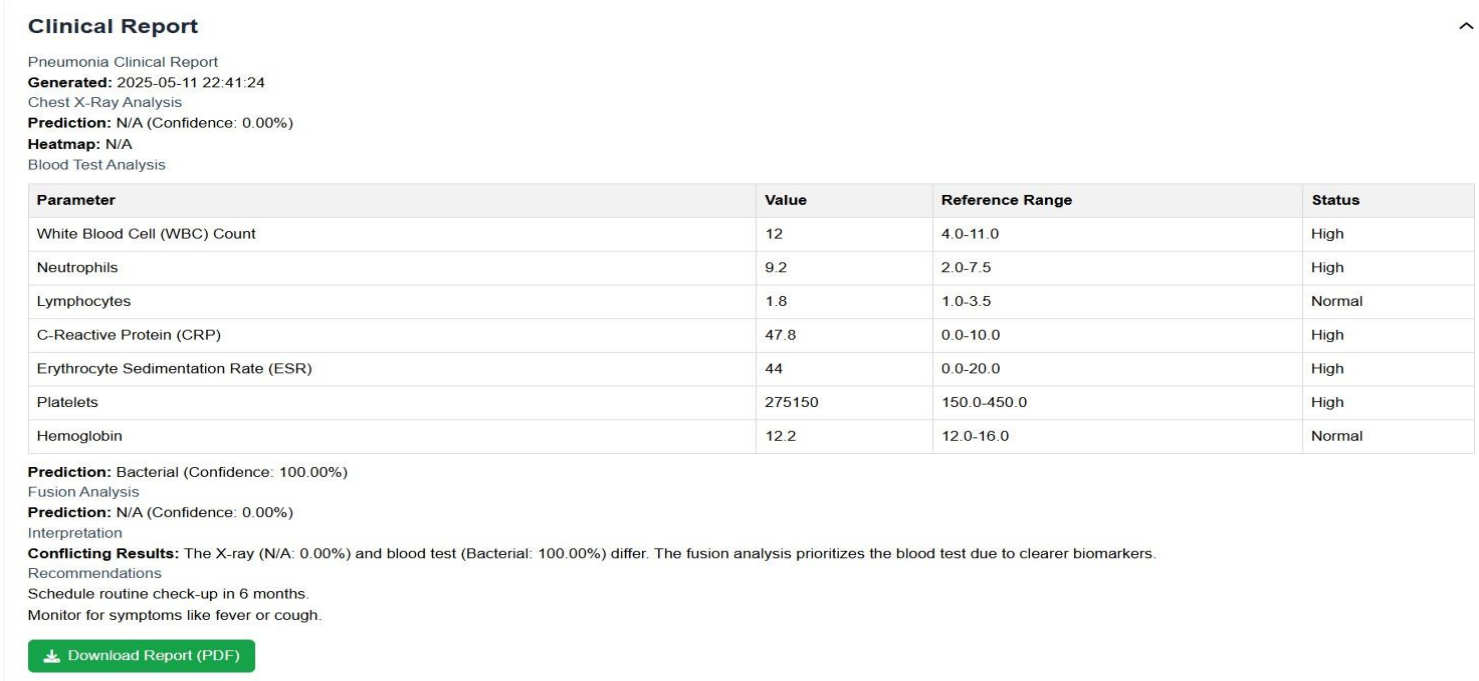

FIGURE B.2 BLOOD TEST REPORT

FIGURE B.3: X-RAY PREDICTION

Report for Model File: output/models/model.h5

	precision	recall	f1-score	support
Normal	0.72	0.87	0.79	234
Pneumonia	0.91	0.80	0.85	390
accuracy			0.83	624
macro avg	0.82	0.84	0.82	624
weighted avg	0.84	0.83	0.83	624

Figure B.4 Report of Trained Model

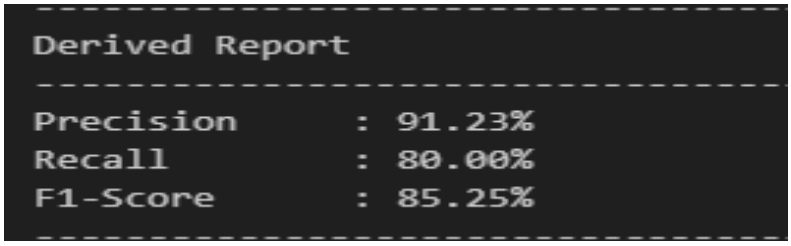

Figure B.5 Test Accuracy of Model

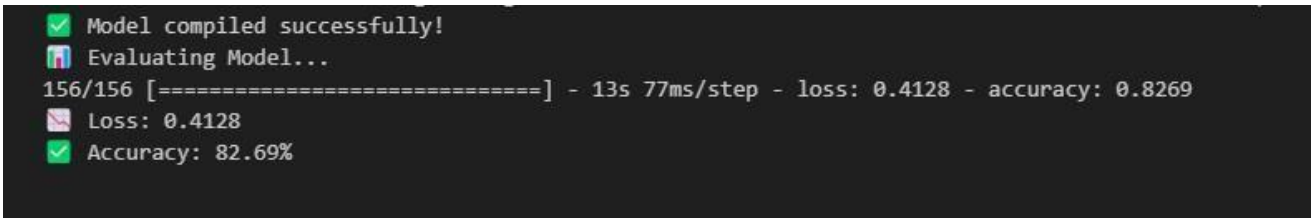

Figure B.6 Test Accuracy for our Model

s

REFERENCES

[1] Alaa M. A. Barhoom and Prof. Dr. Samy S. Abu Naser, (2022), “Diagnosis of Pneumonia Using Deep Learning International Journal of Academic Engineering Research”, Vol. 6 (Issue 2), pp. 48-68.

[2] Albawi, S.; Mohammed, T.A.; Al-Zawi, S, (2017), “Understanding of a convolutional neural network”, 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 21–23 August 2017, pp. 1–6.

[3] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson, (2014), “CNN features off-the-shelf: an astounding baseline for recognition”, in proceedings of IEEE Conference on Computer Vision and Pattern Recognition workshops, pp. 806 - 813.

[4] M. Anthopoulos, S. Christodoulidis, L. Ebner, A. Christe, and S.Mougiakakou, (2016), "Lung Pattern Classification for Interstitial Lung Diseases Using a Deep Convolutional Neural Network," in IEEE Transactions on Medical Imaging, vol. 35, (Issue no. 5), pp. 1207-1216.

[5] D. Demner-Fushman, M.D. Kohli, M.B. Rosenman, S.E. Shooshan, L. Rodriguez, S. Antani,G.R. Thoma, and C.J. McDonald, (2016), “Preparing a collection of radiology examinations for distribution and retrieval”, J Am Med Inform Assoc, Vol 23(2), pp.304 - 10.

[6] Dev Kumar Das, Madhumala Ghosh, Mallika Pal, Asok K Maiti, Chandan Chakraborty, (2013), “Machine learning approach for automated screening of malaria parasite using light microscopic images”, Micron 45, pp. 97-106.

[7] X. Glorot and Y. Bengio, (2010), "Understanding the difficulty of training deep feedforward neural networks", Proceedings of the thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249-256.

[8] Kai-Lung Hua, Che-Hao Hsu, Shintami Chusnul Hidayati, WenHuang Cheng, and Yu-Jen Chen, (2015), “Computer-aided classification of lung nodules on computed tomography images via deep learning technique”, OncoTargets and therapy, vol 8, pp. 506 - 513.

[9] Y. LeCun, Y. Bengio, G. Hinton, (2015), “Deep learning”, Nature, vol. 521, pp. 436 - 444.

[10] Li. Deng and D. Yu, (2014), “Deep Learning”, Signal Processing, vol. 7, pp. 197 - 387.

[11] Mahdieh Poostchi, Kamolrat Silamut, Richard Maude, Stefan Jaeger, George Thoma, (2018), “Image analysis and machine learning for detecting malaria”, Translational Research, Vol 194, pp. 36-55.

[12] S. Militante, (2019),"Fruit Grading of Garcinia Binucao (Batuan) using Image Processing.", International Journal of Recent Technology and Engineering (IJRTE), vol 8.2, pp. 1829-1832.

[13] S. V. Militante, B. D. Gerardo, and N. V. Dionisio, (2019), "Plant Leaf Detection and Disease Recognition using Deep Learning,", IEEE Eurasia Conference on IoT, Communication and Engineering (ECICE), Yunlin, Taiwan, pp. 579-582.

[14] S. V. Militante, and B. G. Sibbaluca,(2020), “Pneumonia Detection Using Convolutional Neural Networks”, International Journal of Scientific & Technology Research, Volume 9(Issue 04), pp. 1332-1337.

[15] P. Moeskops et al., (2016), “Automatic segmentation of MR brain images with a convolutional neural network,”, IEEE transactions on medical imaging, vol. 35 (Issue no. 5), pp. 1252-1261.

[16] Nicholas E Ross, Charles J Pritchard, David M Rubin, and Adriano G Duse, (2006), “Automated image processing method for the diagnosis and classification of malaria on thin blood smears”, Medical and Biological Engineering and Computing Vol 44, 5, pp. 427 - 436.

[17] Nijhawan R, Rishi M, Tiwari A, Dua R. A, (2019), “Novel Deep Learning Framework Approach for Natural Calamities Detection”, In Information and Communication Technology for Competitive Strategies Springer, Singapore, pp. 561-569.

[18] E. Sayed, et. al, (2014), “Computer-aided Diagnosis of Human Brain Tumor through MRI: A Survey and a new algorithm”, Expert System with Applications ,pp. 41-45.

[19] Simonyan, K.; Zisserman, A, (2015), “Very Deep Convolutional Networks for Large-Scale Image Recognition”, 3rd International Conference on Learning Representations, ICLR 2015, San Diego CA, pp. 777-780

[20] G. van Tulder and M. de Bruijne, (2016), “Combining Generative and Discriminative Representation Learning for Lung CT Analysis With Convolutional Restricted Boltzmann Machines,” IEEE transactions on medical imaging, vol. 35 (Issue no. 5), pp. 1262-1272.

[21] Ji wan, Dayong wang, Steven Chu Hong HOI, Pengcheng WU, Jianke ZU, Yongdong Zhang, Jintao Li, (2014), “Deep learning for contentbased image retrieval”, 22nd ACM International Conference on Multimedia, pp. 157-166.

[22] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, (2017), "ChestX-Ray8: Hospital-Scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases,”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, pp. 3462-3471.

[23] Z. Yan et al., (2016), “Multi-Instance Deep Learning: Discover Discriminative Local Anatomies for Bodypart Recognition,” IEEE transactions on medical imaging, vol. 35 (Issue no. 5), pp. 1332-1343.