

# **PNEUMONIA DETECTION USING CONVOLUTIONAL NEURAL NETWORK (CNN)**

A project report submitted in the partial fulfilment of the requirements for the award

of

**Bachelor of Technology**

in

**Electronics and Communication Engineering**

by

**M ALEKYA (20A81A04A2)**

**G WORSHIP VIVEK (20A81A0482)**

**A JNAN VICTOR KERI (20A81A0469)**

**S N V KUMAR SWAMY (20A81A04B7)**

under the guidance of

**Sri T Sreenivas M. Tech., (Ph. D)**

**Assoc Professor**



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

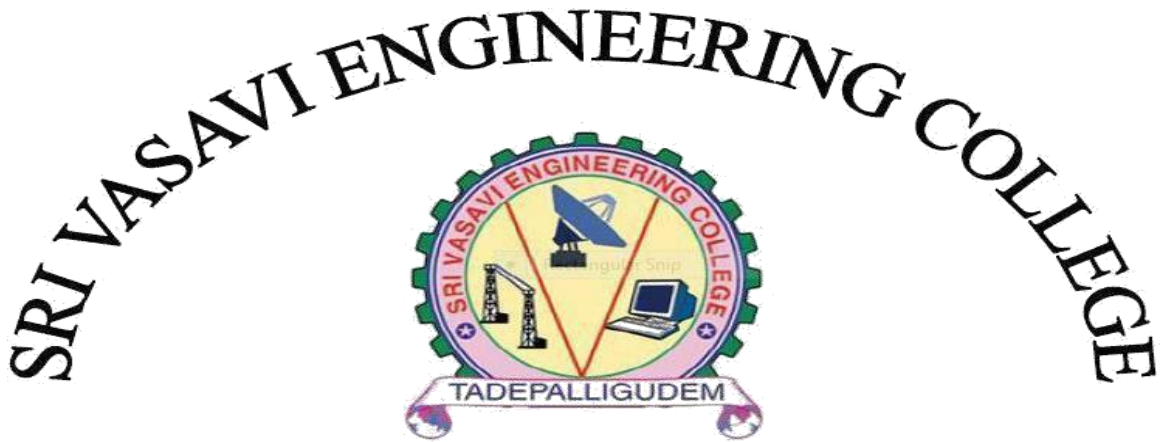
**SRI VASAVI ENGINEERING COLLEGE (AUTONOMOUS)**

(Approved by **AICTE**, Affiliated to JNTUK, Kakinada)

Accredited by **NBA** and **NAAC** with 'A' Grade

Pedatadepalli, **TADEPALLIGUDEM – 534 101**.W.G.Dist. (**A.P**)

**2020-2024**



**DEPARTMENT OF**  
**ELECTRONICS AND COMMUNICATION ENGINEERING**  
**CERTIFICATE**

This is to certify that the project report entitled “**PNEUMONIA DETECTION USING CONVOLUTION NEURAL NETWORK(CNN)**” submitted by **M. ALEKYA (20A81A04A2)**, **G. WORSHIP VIVEK (20A81A0482)**, **A. JNAN VICTOR KERI (20A81A0469)**, **S.N.V. KUMAR SWAMY (20A81A04B7)**, in partial fulfillment of the requirements for award of the Degree of Bachelor of Technology in **ELECTRONICS AND COMMUNICATION ENGINEERING** for the Academic Year 2020 - 2024 of **SRI VASAVI ENGINEERING COLLEGE**, Tadepalligudem affiliated to JNTUK, accredited by NBA and NAAC with ‘A’ grade, is a record of bonafide work carried out by them under my guidance and supervision.

**Project Guide**  
**Sri T. Srinivas**  
**Assoc Professor**

**Head of the Dept**  
**Dr. E. Kusuma Kumari**  
**Professor**

**External Examiner**

## ACKNOWLEDGEMENT

We take this opportunity to express our profound sense of gratitude in all its humbleness to our beloved **Sri T. Sreenivas**, M. Tech, Ph.D., Sr, Asst. Professor in Electronics and Communication Engineering, Sri Vasavi Engineering College, Tadepalligudem. For his excellent guidance, meticulous care and enthusiastic encouragement.

We would like thank to our project coordinator **Dr. T. D.N.S.S. SARESWARA RAO.**, M. Tech, Ph.D., Department of ECE, who has been directly and indirectly part of this journey, for his encouragement to complete our project work, for this excellent guidance, meticulous care and enthusiastic encouragement in motivating us to take up this challenging task and helped us in the completion of the project.

We are indebted to **Dr. E. KUSUMA KUMARI, Ph.D., Professor & Head of the Department** of Electronics and Communication Engineering, for her constant encouragement during execution of the project.

We would like to express our gratitude to **Dr. G.V.N.S.R. RATNAKARA RAO, Ph.D., Principal and Management** of our college for their valuable advice and Supervision up without which this project would not have seen the light of the day. We sincerely wish to thank all the staff members of the department of Electronics and Communication Engineering.

## PROJECT ASSOCIATES

<b>M ALEKYA</b>	<b>(20A81A04A2)</b>
<b>G WORSHIP VIVEK</b>	<b>(20A81A0482)</b>
<b>A JNAN VICTOR KERI</b>	<b>(20A81A0469)</b>
<b>S N V KUMAR SWAMY</b>	<b>(20A81A04B7)</b>

## DECLARATION

We hereby declare that the project entitled “**PNEUMONIA DETECTION USING CONVOLUTIONAL NEURAL NETWORK(CNN)**” is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology in Electronics and Communication Engineering under the esteemed supervision of **Sri T. Sreenivas, M. Tech, Ph.D., Sr, Asst. Professor in Electronics and Communication Engineering.** This is a record of work carried out by us and results embodied in this project report have not been submitted to any other university for the award of any degree.

## PROJECT ASSOCIATES

<b>M ALEKYA</b>	<b>(20A81A04A2)</b>
<b>G WORSHIP VIVEK</b>	<b>(20A81A0482)</b>
<b>A JNAN VICTOR KERI</b>	<b>(20A81A0469)</b>
<b>S N V KUMAR SWAMY</b>	<b>(20A81A04B7)</b>

## ABSTRACT

Pneumonia is a life-threatening respiratory disease, particularly affecting the elderly population. Early and accurate diagnosis is crucial for effective treatment and improved patient outcomes.

This project presents a method for detecting and classifying pneumonia in patients based on their chest X-ray images using EfficientNetB1 model in convolutional neural networks (CNNs). The proposed approach utilizes a series of convolutional and max-pooling layers followed by SeLU activation to process the input images. The processed features are then passed through dense layers with sigmoid activation at the output neuron for classification. To improve model generalization, data augmentation techniques are employed. The proposed model is used to develop a robust and efficient model capable of accurately identifying pneumonia cases from chest X-ray images. The experimental results demonstrate that the EfficientNetB1 CNN model achieves high accuracy in pneumonia detection, making it a promising tool for early diagnosis in clinical settings.

***Keywords:***

Pneumonia, CT scans, Deep Learning, Image Analysis,  
Convolutional Neural Network (CNN)

# CONTENTS

ABSTRACT .....	v
CONTENTS .....	vi
LIST OF FIGURES .....	ix
LIST OF TABLES .....	x
CHAPTER 1 .....	1
INTRODUCTION .....	1
1.1    OVERVIEW OF PNEUMONIA .....	2
1.1.1    CAUSES: .....	2
1.1.2    TRANSMISSION: .....	2
1.1.3    RISK FACTORS: .....	3
1.1.4    SYMPTOMS: .....	3
1.1.5    DIAGNOSIS: .....	3
1.1.6    PREVENTION: .....	4
1.1.7    TREATMENT: .....	4
1.2    IMPORTANCE OF EARLY DETECTION .....	4
1.3    DEEP LEARNING FOR MEDICAL DIAGNOSIS .....	5
CHAPTER 2 .....	7
LITERATURE REVIEW .....	7
2.1    STUDY OF RELATED WORKS .....	7
CHAPTER 3 .....	10

CONVOLUTIONAL NEURAL NETWORK .....	10
3.1 CONVOLUTIONAL NEURAL NETWORK .....	10
3.2 CNN ARCHITECTURE .....	10
3.2.1 CONVOLUTIONAL LAYER .....	11
3.2.2 POOLING LAYER .....	11
3.2.3 FULLY CONNECTED LAYER .....	12
CHAPTER 4.....	15
PROPOSED METHODOLOGY .....	15
4.1 PROPOSED METHOD FOR PNEUMONIA DETECTION .....	15
4.2 WORKING FLOW AND IMPLEMENTATION .....	17
4.3 PRE-TRAINED MODELS .....	21
4.3.1 VGG16.....	22
4.3.2 RESNET.....	23
4.3.3 INCEPTIONV3.....	24
4.4 PROPOSED MODEL .....	25
CHAPTER 5.....	26
SOFTWARE DESCRIPTION .....	26
5.1 PYTHON .....	26
5.1.1 Introduction To Python .....	26
5.1.2 Python Features.....	27
5.1.3 Python Libraries.....	28
5.1.4 PYTHON APPLICATIONS.....	29

5.2	DEEP LEARNING .....	31
5.2.1	INTRODUCTION TO DEEP LEARNING .....	31
5.2.2	DEEP LEARNING LIBRARIES AND FRAMEWORKS.....	31
5.2.3	DEEP LEARNING VS MACHINE LEARNING.....	32
CHAPTER 6.....		33
RESULTS AND ANALYSIS .....		33
6.1	DATASET DESCRIPTION.....	33
6.2	IMPLEMENTATION DETAILS .....	35
6.2.1	Pretrained Models Implementation.....	35
6.2.2	PROPOSED MODEL IMPLEMENTATION.....	38
6.3	PARAMETERS.....	41
6.4	PERFORMANCE OF DIFFERENT MODELS.....	42
6.5	PERFORMANCE OF EXISTING AND PROPOSED MODELS.....	43
6.6	GRAPHS FOR ACCURACY AND LOSS .....	44
CHAPTER 7.....		47
CONCLUSION .....		47
7.1	CONCLUSION .....	47
7.2	FUTURE SCOPE .....	47
CHAPTER 8.....		48
REFERENCES .....		48



## LIST OF FIGURES

Figure 1Difference between normal and Pneumonia lung .....	2
Figure 2Main symptoms of infectious Pneumonia.....	3
Figure 3Architecture of CNN.....	10
Figure 4Pooling Layer.....	12
Figure 5Architecture of CNN layers .....	14
Figure 6Workflow of proposed model .....	17
Figure 7VGG16 Architecture .....	22
Figure 8ResNet Architecture.....	23
Figure 9InceptionV3 Architecture.....	24
Figure 10Architecture of EfficientNetB1 .....	25
Figure 11Images of normal Pneumonia.....	33
Figure 12Test, Train directories .....	34
Figure 13Implementation of VGG16 .....	35
Figure 14Implementation of INCEPTIONV3 .....	36
Figure 15Implementation of ResNet .....	37
Figure 16Implementation of EfficinetNet .....	39
Figure 17 Result of EfficientNetB1 Model .....	40
Figure 18Predictions of EfficientNet.....	40
Figure 19Parameters of proposed.....	41
Figure 20Graphs for Accuracy .....	44
Figure 21Graphs for Loss.....	45

## **LIST OF TABLES**

Table 1Literature Review .....	9
Table 2Parameters of different models.....	42
Table 3Accuracy in different Models.....	43
Table 4Performance of Existing model and Proposed model .....	43

# **CHAPTER 1**

## **INTRODUCTION**

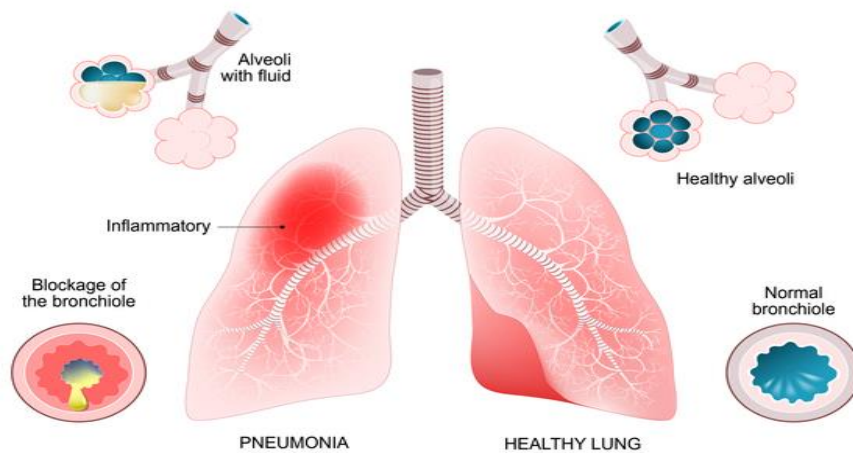
Pneumonia is a respiratory infection that affects the lungs, causing inflammation in the air sacs, or alveoli. It can be caused by various infectious agents, including bacteria, viruses, fungi, or parasites. Pneumonia can range from mild to severe and can be life-threatening, particularly in vulnerable populations such as young children, the elderly, and those with weakened immune systems. Treatment for pneumonia depends on the cause and severity of the infection. Bacterial pneumonia is commonly treated with antibiotics, while viral pneumonia may require supportive care such as rest, hydration, and antiviral medications. In severe cases, hospitalization may be necessary for close monitoring and intravenous antibiotics or oxygen therapy.

Pneumonia killed more than 808 000 children under the age of 5 in 2017, accounting for 15% of all deaths of children under 5 years. People at-risk for pneumonia also include adults over the age of 65 and people with preexisting health problems.

The lungs are made up of small sacs called alveoli, which fill with air when a healthy person breathes. When an individual has pneumonia, the alveoli are filled with pus and fluid, which makes breathing painful and limits oxygen intake. These infections are generally spread by direct contact with infected people.

Despite advances in medical care, pneumonia remains a significant global health concern, contributing to substantial morbidity and mortality worldwide. Efforts

to improve prevention, early detection, and treatment are crucial in combating this respiratory illness and reducing its impact on public health.



*Figure 1 Difference between normal and Pneumonia lung*

## **1.1 OVERVIEW OF PNEUMONIA**

### **1.1.1 CAUSES:**

Pneumonia can be caused by a variety of infectious agents, including bacteria, viruses, fungi, and, less commonly, parasites. The most common bacterial cause of pneumonia is *Streptococcus pneumoniae*, while viruses such as influenza, respiratory syncytial virus (RSV), and adenovirus are also frequent culprits. Fungal pneumonia is more common in individuals with weakened immune systems or underlying lung conditions.

### **1.1.2 TRANSMISSION:**

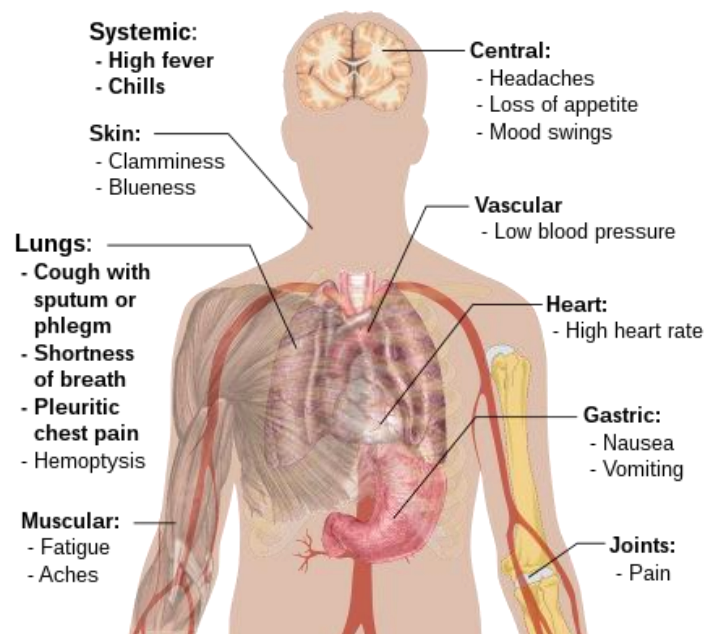
The transmission of pneumonia-causing pathogens typically occurs through respiratory droplets from coughs or sneezes of infected individuals. It can also spread indirectly by touching contaminated surfaces and then touching the face.

### 1.1.3 RISK FACTORS:

Certain factors increase the risk of developing pneumonia, including age (very young or elderly), weakened immune system, chronic lung diseases (such as COPD), smoking, underlying medical conditions (like heart disease or diabetes), recent respiratory infections, and exposure to pollutants or toxins.

### 1.1.4 SYMPTOMS:

Symptoms of pneumonia can vary depending on the cause, the individual's age, and overall health. Common symptoms include cough, fever, chills, chest pain, rapid breathing, fatigue, and difficulty breathing. In older adults and those with weakened immune systems, symptoms may be less specific or severe.



*Figure 2 Main symptoms of infectious Pneumonia*

### 1.1.5 DIAGNOSIS:

Diagnosing pneumonia typically involves a combination of clinical assessment, physical examination, chest X-rays, and sometimes laboratory tests such as sputum analysis or blood cultures to identify the causative agent.

### **1.1.6 PREVENTION:**

Preventive measures include vaccination against common bacterial and viral pathogens, practicing good respiratory hygiene (e.g., washing hands frequently, covering coughs and sneezes), avoiding smoking and exposure to pollutants, and managing underlying health conditions.

### **1.1.7 TREATMENT:**

Treatment for pneumonia depends on the cause and severity of the infection. Bacterial pneumonia is usually treated with antibiotics, while viral pneumonia may require supportive care such as rest, hydration, and antiviral medications. Severe cases may necessitate hospitalization for close monitoring, oxygen therapy, and intravenous antibiotics.

## **1.2 IMPORTANCE OF EARLY DETECTION**

Early detection of pneumonia disease is paramount due to its significant impact on treatment outcomes, public health, and healthcare resource allocation. Timely identification of pneumonia cases enables healthcare providers to promptly initiate appropriate treatment, thereby improving patient outcomes and reducing the risk of complications and mortality associated with the disease. Additionally, early detection facilitates the implementation of infection control measures, helping to contain the spread of infectious agents causing pneumonia and preventing outbreaks. By identifying cases early, healthcare systems can efficiently allocate resources, including hospital beds, medical staff, diagnostic tests, and medications, ensuring that those in need receive timely care while minimizing healthcare costs and system strain. Furthermore, early detection plays a crucial role in preventing long-term health impacts

associated with pneumonia, such as impaired lung function and increased susceptibility to respiratory infections in the future. It also helps prevent secondary infections and reduces the burden on individuals and healthcare systems by facilitating quicker recovery times. Screening high-risk populations, promoting vaccination, and raising awareness about pneumonia symptoms are essential strategies for achieving early detection and mitigating the impact of this respiratory illness.

### **1.3 DEEP LEARNING FOR MEDICAL DIAGNOSIS**

Deep learning is a subset of machine learning that involves algorithms inspired by the structure and function of the human brain's neural networks. In medical diagnosis, deep learning techniques are increasingly being utilized due to their ability to analyze complex datasets, recognize patterns, and make predictions with remarkable accuracy.

Deep learning is employed in pneumonia detection through the analysis of medical images such as chest X-rays and CT scans. Convolutional Neural Networks (CNNs), a type of deep learning architecture, excel in image recognition tasks.

These networks are trained on large datasets of labelled images, learning to identify patterns indicative of pneumonia.

- Image Analysis: CNNs analyze chest X-ray images to identify areas of opacity, which are indicative of lung abnormalities such as pneumonia.

- Feature Extraction: Deep learning models automatically extract relevant features from chest X-ray images, such as the presence of infiltrates, consolidations, and other characteristic signs of pneumonia.

- Classification: Once features are extracted, the model classifies the X-ray as either normal or indicative of pneumonia. This binary classification helps in decision-making by healthcare professionals.
- Training on Labelled Data: CNNs are trained on large datasets of chest X-ray images labelled by radiologists. This training process enables the model to learn the distinguishing features of pneumonia from normal lung tissue.
- Validation and Testing: The trained model is validated and tested on separate datasets to assess its performance in accurately identifying pneumonia cases.
- Performance Evaluation: The performance of the deep learning model is evaluated based on metrics such as sensitivity, specificity, accuracy, and area under the receiver operating characteristic curve (AUC-ROC).
- Integration into Clinical Workflow: Once validated, the deep learning model can be integrated into the clinical workflow to assist radiologists in interpreting chest X-ray images more efficiently and accurately.



## CHAPTER 2

### LITERATURE REVIEW

#### 2.1STUDY OF RELATED WORKS

**Rajpurkar et al. (2017):** proposed a Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning by using CheXNet. This study introduced CheXNet, a CNN model trained on the ChestX-ray14 dataset, achieving performance comparable to radiologists in pneumonia detection. It highlighted the potential of deep learning for automating pneumonia diagnosis from chest X-ray images and discussed the challenges associated with dataset annotation and model interpretation. The highest accuracy achieved is 88.8%. This accuracy value indicates the model's performance in correctly identifying cases of pneumonia versus non-pneumonia from chest X-ray images. It's important to note that accuracy is just one of several performance metrics reported in the paper, alongside other metrics such as sensitivity, specificity, and AUC-ROC, which provide a more comprehensive assessment of the model's performance.

**Deniz Yagmur Urey et al.** presented a research paper involving the early diagnosis of Pneumonia by using deep learning techniques. The authors posit a novel approach of focusing on the biological aspects of this fatal disease and detecting it by X-ray imaging. The classification methods used are Convolutional Neural Networks (CNN) and Residual Neural Networks. The Comparative study helps detecting Pneumonia at an early stage and thus, appropriate treatment can be provided to cure the disease. This research influenced our paper, to conduct a similar assessment, but on a more generalized data, a plethora of images and on a detailed approach of layers of the neural network to make it more efficient and produce higher accuracy.

**Dimpy Varshni et al.** explained the development of an automatic system for detection of pneumonia through various deep learning models. The authors analyzed medical images and developed a Convolutional Neural network for disease classification and scaling of data. The architecture consists of a DenseNet-169 layer architecture for feature extraction. The architecture was combined with a SVM Model for binary classification. The results of the model are analysed with visualization curves.

**Garima Verma et. al.** presented a research paper which analyses and identifies pneumonia, based on X-ray images, using convolutional neural network. The implementation was done using 6 convolutional layers followed by max-pooling layers after each. This provided us with an insight to incorporate lesser number of convolutional layers, for faster computing and classification of the deep learning model. The research helps detect pneumonia, based on Chest X-rays.

**Okeke Stephen et. al.** provides a similar insight on classifying numerous x-rays to detect pneumonia based on convolutional neural networks. The accuracy obtained through their research helps us evaluate our model in comparison, depending on the loss and accuracy of the neural network. Their network is provided with a (200x200x3) dimension input shape, while the images are focused and it uses (64x64x3) dimensions, to decrease the computations.

**Shanay Shah, Heeket Mehta.** proposes a deep learning approach using Convolutional Neural Networks (CNN) for the detection of pneumonia from chest X-ray images. It details the methodology, including data augmentation techniques, CNN architecture with multiple convolutional and max pooling layers, and the use of activation functions

like ReLU and sigmoid. The model is trained and evaluated on a dataset of chest X-ray images, with the goal of accurately classifying images as having pneumonia or being normal. The results demonstrate high accuracy and low loss, suggesting the effectiveness of the proposed CNN model for pneumonia diagnosis from X-ray images.

Author Name	Year	Model Name	Merits	Demerits	Accuracy
Rajpurkar et al.	2017	CheXNet : Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning	High Performance, Interpretability, Transfer Learning	Limited to Frontal-View X-rays, Single Disease Focus, Interpretability Limitations	82%
Deniz Yagmur Urey et al.	2018	ResNet: The early diagnosis of Pneumonia by using deep learning techniques	Use of advanced deep learning models, Architectural improvements	Potential overfitting, Data quality and variability	90.4%
Dimpy Varshni et al.	2019	SVM Model: Automatic system for detection of pneumonia through various deep learning models	Automated Detection, Data Scaling, Visualization	Computational Complexity, Limited Information	87%
Okeke Stephen et. al.	2019	Detect pneumonia based on convolutional neural networks	Feature Learning, Dimensionality Reduction	Limited Information, Data Dependency	90.5%
Shanay Shah, Heeket Mehta	2020	Sequential Model: A deep learning	High Accuracy and Low Loss, Feature Learning,	Data Dependency, Computational	92.47%

*Table 1 Literature Review*

## CHAPTER 3

# CONVOLUTIONAL NEURAL NETWORK

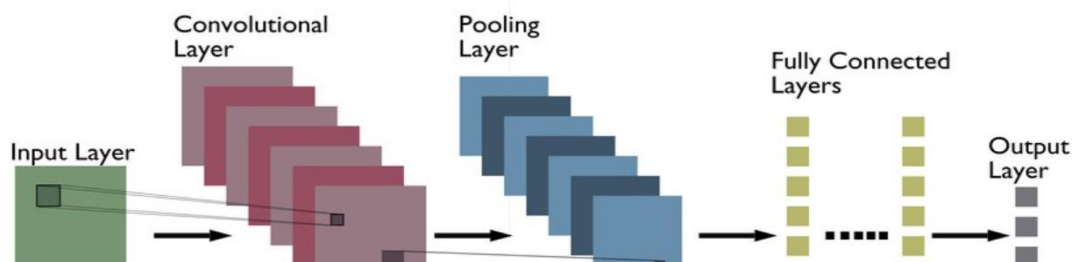
### 3.1 CONVOLUTIONAL NEURAL NETWORK

A Convolutional Neural Network (CNN) is a class of deep neural networks primarily designed to process and analyse visual data such as images. CNNs have revolutionized various fields such as computer vision, image recognition, and medical imaging due to their ability to automatically and accurately learn hierarchical patterns and features directly from raw pixel data.

### 3.2 CNN ARCHITECTURE

Convolutional neural networks are known for their superiority over other artificial neural networks, given their ability to process visual, textual, and audio data. The CNN architecture comprises three main layers: convolutional layers, pooling layers, and a fully connected (FC) layer.

There can be multiple convolutional and pooling layers. The more layers in the network, the greater the complexity and (theoretically) the accuracy of the machine learning model. Each additional layer that processes the input data increases the model's ability to recognize objects and patterns in the data.



*Figure 3 Architecture of CNN*

### 3.2.1 CONVOLUTIONAL LAYER

This layer is the first layer that is used to extract the various features from the input images. In this layer, we use a filter or Kernel method to extract features from the input image. A kernel is a small matrix, with its height and width smaller than the image to be convolved.

The result of this process is a feature map that highlights the presence of the detected features in the image.

$$\frac{W - F + 2P}{S} + 1$$

CNNs often include multiple stacked convolutional layers. Through this layered architecture, the CNN progressively interprets the visual information contained in the raw image data. In the earlier layers, the CNN identifies basic features such as edges, textures or colours. Deeper layers receive input from the feature maps of previous layers, enabling them to detect more complex patterns, objects and scenes.

### 3.2.2 POOLING LAYER

Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.

The pooling layer summarises the features present in a region of the feature map generated by a convolution layer. So, further operations are performed on summarised features instead of precisely positioned features generated by the convolution layer. This makes the model more robust to variations in the position of the features in the input image.

$$\frac{W - F}{S} + 1$$

## TYPES OF POOLING LAYERS

### 1.Max Pooling

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

### 2.Average Pooling

Average pooling computes the average of the elements present in the region of feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.

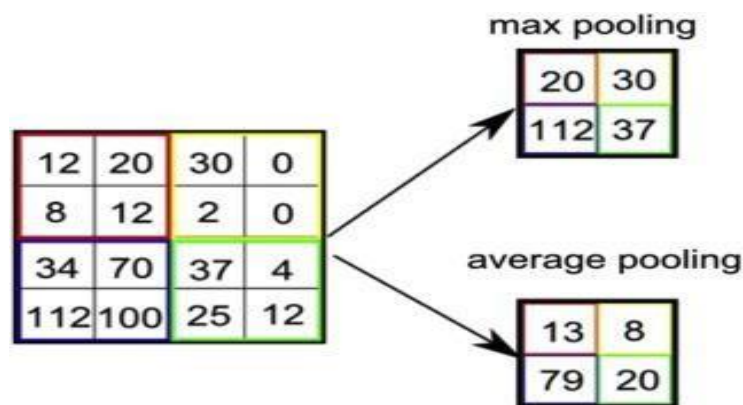


Figure 4 Pooling Layer

### 3.2.3 FULLY CONNECTED LAYER

A Fully Connected (FC) layer, also known as a dense layer, is a type of layer used in artificial neural networks where each neuron or node from the previous layer is connected to each neuron of the current layer. It's called "fully connected" because of this complete linkage. FC layers are typically found towards the end of a neural network architecture and are responsible for producing final output predictions.

The FC layer performs classification tasks using the features that the previous layers and filters extracted. Instead of ReLu functions, the FC layer typically uses a SoftMax function that classifies inputs more appropriately and produces a probability score between 0 and 1.

### **Activation Function**

An Activation Function decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction. There are several Commonly used activation functions such as the ReLu, SoftMax, tanH, and the Sigmoid functions. Each of these functions has a specific usage.

**-Sigmoid** - For a binary classification in the CNN model.

**-tanH** - The tanh function is very similar to the sigmoid function. The only difference is that it is symmetric around the origin. The range of values, in this case, is from -1 to 1.

**-SoftMax**- It is used in multinomial logistic regression and is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes.

**-ReLU**- the main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time.

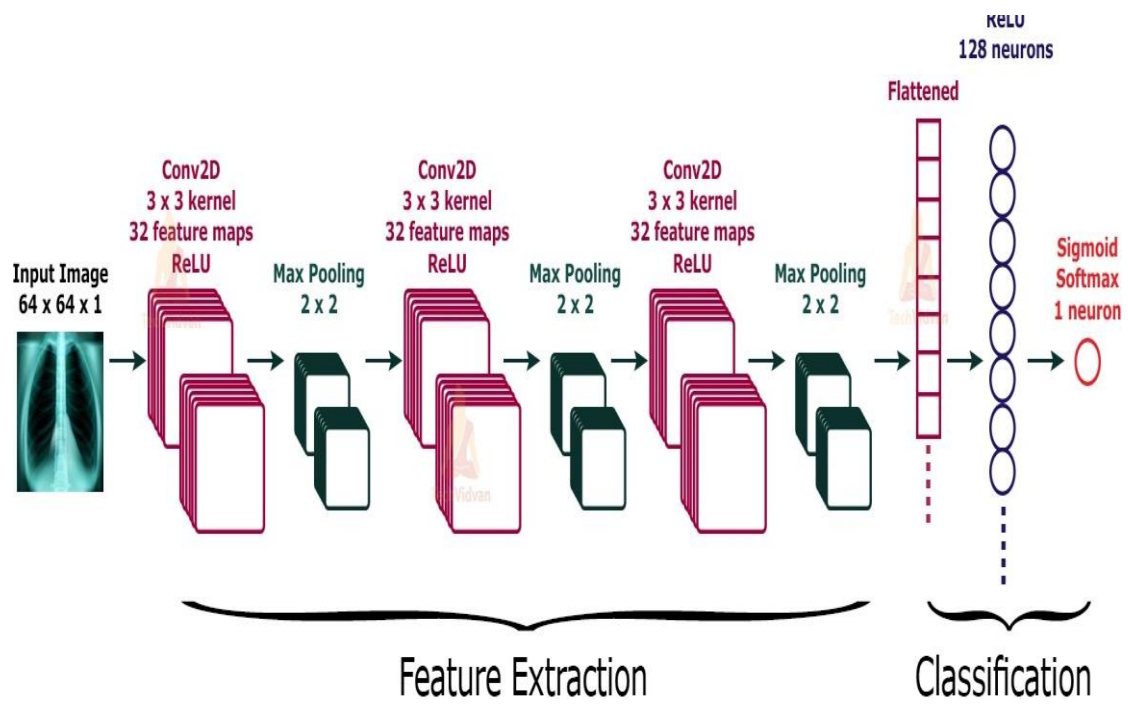


Figure 5 Architecture of CNN layers



## **CHAPTER 4**

### **PROPOSED METHODOLOGY**

#### **4.1 PROPOSED METHOD FOR PNEUMONIA DETECTION**

In our proposed project for pneumonia detection using Convolutional Neural Networks (CNN), we employed the EfficientNetB1 model architecture, a fundamental building block in deep learning frameworks such as Keras. The Sequential model allowed us to construct our CNN architecture layer by layer in a sequential fashion, facilitating the intuitive design and implementation of the neural network.

Beginning with the input layer, which received grayscale Chest X-ray images of dimensions (256, 256, 3), we sequentially added convolutional layers, batch normalization layers, max pooling layers, dropout layers, and fully connected layers. Each layer played a crucial role in feature extraction, dimensionality reduction, regularization, and high-level reasoning, ultimately leading to accurate pneumonia detection.

The EfficientNetB1 model provided a clear and organized structure for our CNN architecture, enabling us to easily experiment with different layer configurations, hyperparameters, and optimization techniques. By leveraging the Sequential model, we were able to efficiently develop and train our pneumonia detection model, achieving high accuracy and robust performance on unseen data.

## PROPOSED MODEL

```
base_model=keras.applications.EfficientNetB1(weights='imagenet',
include_top=False,input_shape=(256, 256, 3))

x = base_model.output

x = keras.layers.Conv2D(filters=256, kernel_size=(5, 5), strides=(1, 1),
activation='relu', padding="same")(x)

x = keras.layers.Conv2D(filters=128, kernel_size=(5, 5), strides=(1, 1),
activation='relu', padding="same")(x)

x = keras.layers.Conv2D(filters=256, kernel_size=(5, 5), strides=(1, 1),
activation='relu', padding="same")(x)

x = keras.layers.GlobalAveragePooling2D()(x)

x = keras.layers.Dense(128, activation='selu')(x)

output = keras.layers.Dense(3, activation="softmax")(x)

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

iters = int(train_df.shape[0]/32)*25

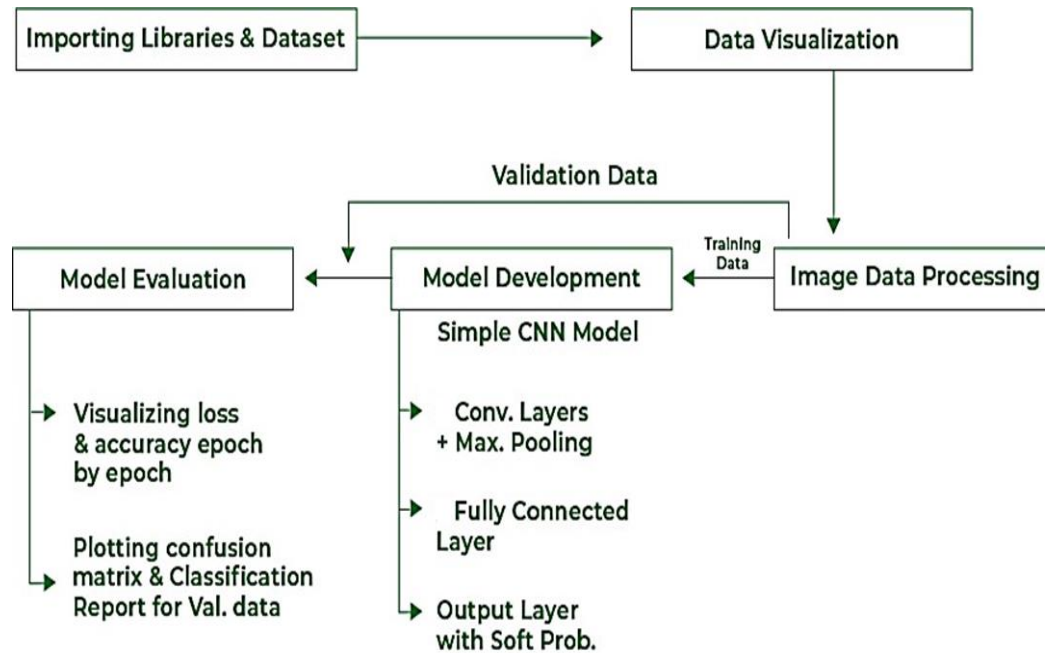
one_cycl = OneCycleLR(iters,max_lr=0.00085)

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

history2=model.fit(train,validation_data=val,epochs=25,batch_size=32,callbacks=[one_cycl])
```

## 4.2 WORKING FLOW AND IMPLEMENTATION

### IMPLEMENTATION



*Figure 6 Workflow of proposed model*

**Working Flow of the proposed model involves –**

**1.Importing Libraries & Dataset:** Import the necessary libraries as Pandas, NumPy, Matplotlib, Seaborn, Keras and from these libraries import required functions. For the dataset requirement from the Kaggle online platform download pneumonia chest images dataset.

**2.Data Visualization:** This block preprocesses the data by mapping numerical labels to categorical labels (“Lung\_Opacity”, “Normal”, and “Viral Pneumonia”), and then it visualizes the distribution of these labels using a count plot to gain insights into the dataset's class distribution. This step is essential for understanding the balance of

classes in the dataset and identifying any potential class imbalance issues that may need to be addressed during model training.

**3.Image Data Processing:** Preprocessing the training data images is a crucial step in preparing them for training a Convolutional Neural Network (CNN). The preprocessing steps ensure that the input data is in a suitable format and condition for the model to learn effectively. It follows some steps such as

Resizing Images- Resizing images to a fixed size ensures that all images fed into the model have the same dimensions, making it easier for the model to learn patterns and features consistently across the dataset.

Normalization-Normalization involves scaling the pixel values of the images to a specific range. Commonly, pixel values are scaled to the range  $[0, 1]$  or  $[-1, 1]$ . Normalization helps stabilize and accelerate the training process by ensuring that the input data is within a consistent numerical range.

Data Augmentation: Data augmentation techniques are applied to increase the diversity and robustness of the training dataset. Augmentations such as random rotations, flips, shifts, zooms, and brightness adjustments introduce variations in the training data, helping the model generalize better to unseen examples.

Corresponding to each preprocessed image, there are labels indicating the class or category of the image. For binary classification tasks like pneumonia detection, the labels may be binary (e.g., 0 for "Pneumonia" and 1 for "Normal").

The dataset is typically split into training and validation data using a predefined ratio, such as 80% for training and 20% for validation. This split ensures that the model is trained on a sufficient amount of data while also having a separate subset for validation.

**Training data:** The training data subset is used to train the CNN model. It consists of a large portion of the preprocessed images and their corresponding labels. During

training, the model learns from the patterns and features present in the training data to make accurate predictions. The training data should be representative of the overall dataset and contain a diverse range of examples to facilitate effective learning.

**Validation data:** The validation data subset is used to evaluate the performance of the CNN model during training. It consists of a smaller portion of the preprocessed images and their corresponding labels, separate from the training data. The validation data helps monitor the model's performance on unseen data and detect overfitting or underfitting during training. Unlike the training data, the validation data should not be used for training the model; its sole purpose is to provide an unbiased evaluation of the model's performance.

During the training process, the model is trained using the training data, and its performance is evaluated on the validation data at regular intervals (e.g., after each epoch). This evaluation helps track the model's progress and allows for adjustments to the model architecture, hyperparameters, and optimization strategy as needed to improve performance.

**4. Model Development:** To develop a CNN model with the specified layers, we create a sequential model using Keras, a popular deep learning library.

**Convolutional Layers (5):** Convolutional layers apply learnable filters to the input data, extracting features through convolution operations. In this model, we'll add 5 convolutional layers, each followed by a ReLU activation function for introducing non-linearity.

**Batch Normalization Layers (4):** Batch normalization layers normalize the activations of the previous layer, helping stabilize and accelerate the training process. In this model, we'll add 4 batch normalization layers after the convolutional layers.

**Max Pooling Layers (5):** Max pooling layers downsample the feature maps generated by the convolutional layers, reducing spatial dimensions and computational complexity. In this model, we'll add 5 max pooling layers to perform spatial downsampling.

**Dropout Layers (4):** Dropout layers randomly deactivate a fraction of neurons during training, preventing overfitting by promoting robustness and generalization in the network. In this model, we'll add 4 dropout layers to regularize the network.

**Flatten Layer (1):** The flatten layer transforms the multi-dimensional feature maps into a one-dimensional vector, preparing the data for input into the fully connected layers.

**Dense (Fully Connected) Layers (2):** Dense layers are fully connected layers that aggregate features and perform high-level reasoning and decision-making tasks. In this model, we'll add 2 dense layers, followed by a final output layer with a sigmoid activation function for binary classification.

## **5. Model Evaluation:**

### **a. Visualize Loss and Accuracy Epoch by Epoch:**

During the training process, the model computes both the loss and accuracy metrics on the training and validation datasets after each epoch (complete pass through the entire training dataset).

A line plot can be created where the x-axis represents the number of epochs, and the y-axis represents the value of the loss or accuracy metric.

### **b. Plot the Confusion Matrix for the Validation Data:**

The confusion matrix is a table that visualizes the performance of a classification model.

It shows the counts of true positive, true negative, false positive, and false negative predictions made by the model on the validation dataset.

Each row of the matrix represents the actual class labels, while each column represents the predicted class labels.

Plotting the confusion matrix provides a comprehensive view of the model's performance, including its ability to correctly classify different classes and any potential misclassifications.

### **C. Generate a Classification Report for the Validation Data:**

The classification report provides a detailed summary of the model's performance on the validation dataset, including metrics such as precision, recall, F1-score, and support for each class.

Additionally, the report may include the accuracy and other metrics to assess the overall performance of the model across all classes.

Generating a classification report helps in evaluating the model's performance on individual classes and identifying areas for improvement.

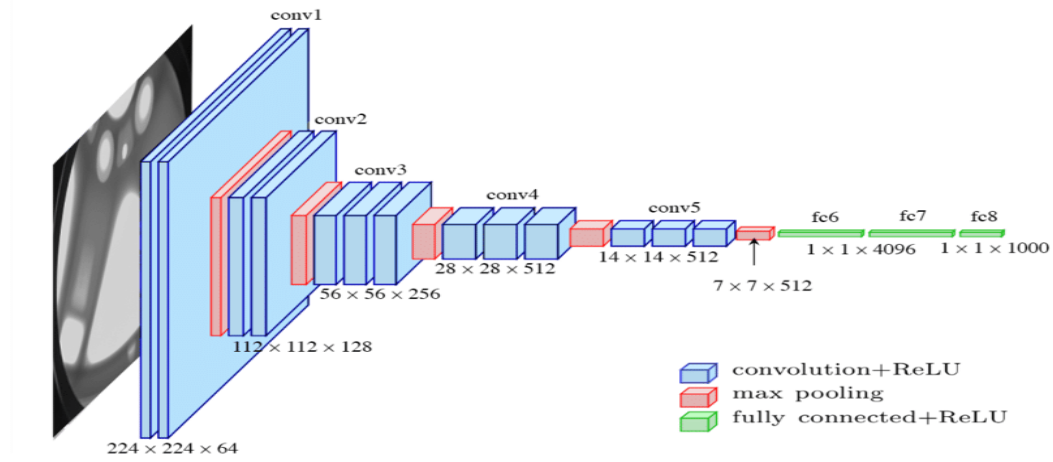
## **4.3 PRE-TRAINED MODELS**

A pre-trained model refers to a neural network architecture that has been previously trained on a large-scale dataset, typically for a general task such as image classification or natural language processing. These models learn meaningful representations of features from the data, capturing patterns and relationships that are transferable across related tasks. By leveraging pre-trained models, practitioners can benefit from the knowledge encoded in the learned parameters, enabling faster

convergence and improved performance on specific tasks, even with limited training data.

Fine-tuning pre-trained models involves adapting them to new datasets or tasks by updating their parameters through additional training, resulting in state-of-the-art performance with reduced computational resources and training time.

#### 4.3.1 VGG16

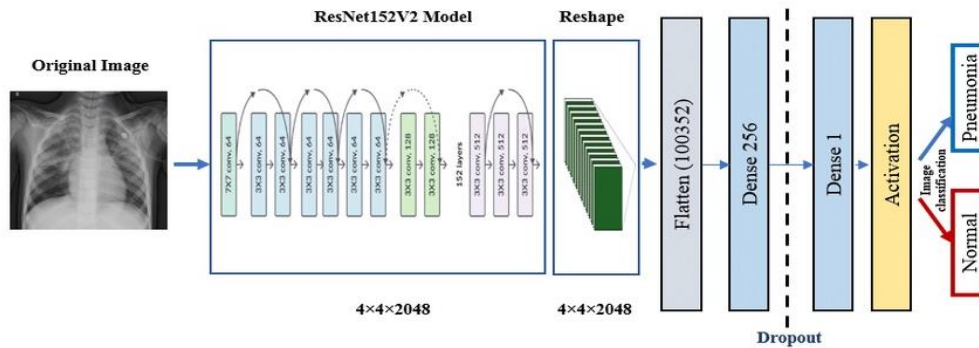


*Figure 7VGG16 Architecture*

VGG16 is a convolutional neural network (CNN) architecture consisting of 16 layers, developed by the Visual Geometry Group (VGG) at the University of Oxford. It gained popularity for its simplicity and effectiveness in image classification tasks. VGG16 is trained on large-scale datasets like ImageNet, enabling it to learn rich feature representations from images. It consists of convolutional layers with small  $3 \times 3$  filters, followed by max-pooling layers for down sampling. VGG16 has been widely used as a base model for various computer vision applications, including object recognition and feature extraction, thanks to its strong performance and generalization ability.



### 4.3.2 RESNET



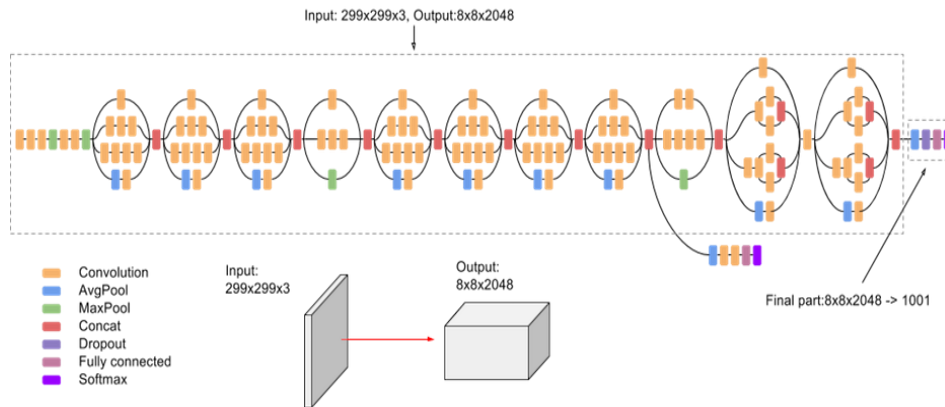
*Figure 8 ResNet Architecture*

ResNet, short for Residual Network, is a powerful convolutional neural network (CNN) architecture that revolutionized deep learning in computer vision tasks. Introduced by Kaiming He et al. in 2015, ResNet addresses the problem of vanishing gradients and degradation in deep networks by introducing skip connections or shortcuts that allow the network to skip certain layers.

ResNet architectures come in several variants, including ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, which differ in the number of layers and computational complexity. These architectures have been widely adopted and adapted for various computer vision tasks, such as image classification, object detection, and semantic segmentation, achieving state-of-the-art performance on benchmark datasets.

ResNet has significantly advanced the field of deep learning by enabling the training of deeper and more accurate neural networks, contributing to breakthroughs in image recognition and other computer vision applications.

### 4.3.3 INCEPTIONV3



*Figure 9 InceptionV3 Architecture*

Inception V3 is a CNN-based Deep transfer learning approach, was released by Google Net in 2014. This model has 42 layers and is less error-prone than its predecessors. It has  $7 \times 7$  convolutions and uses an auxiliary classifier to spread label information throughout the network. It employs RMPS optimizers and includes label smoothing, a normalizing component added to the loss formula to prevent outliers in a class from causing over fitting in the model that produces the best results.

It is designed for image classification and object detection tasks. InceptionV3 builds upon the original Inception architecture, introducing several improvements to enhance performance and efficiency.

Key features of InceptionV3 include the use of inception modules, which employ multiple parallel convolutional operations of different sizes within the same layer. This allows the network to capture features at various scales and complexities, leading to improved representation learning.

## 4.4 PROPOSED MODEL

### EFFICIENTNETB1

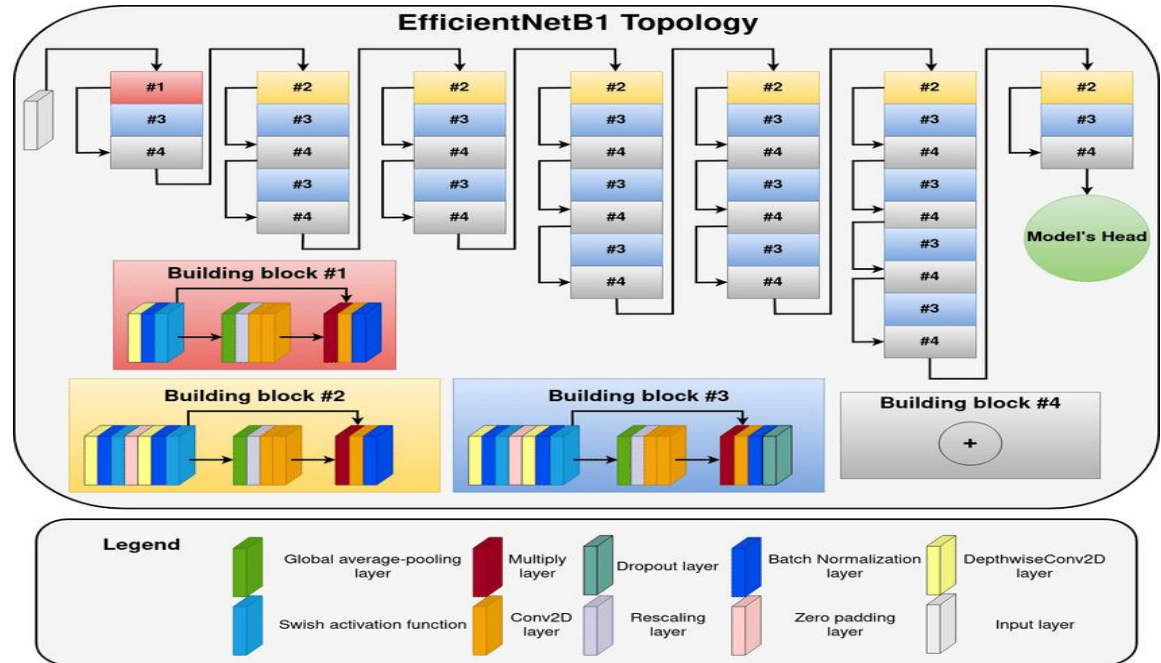


Figure 10 Architecture of EfficientNetB1

EfficientNet, first introduced in Tan and Le, 2019 is among the most efficient models (i.e. requiring least FLOPS for inference) that reaches State-of-the-Art accuracy on both ImageNet and common image classification transfer learning tasks. The smallest base model is similar to MnasNet, which reached near-SOTA with a significantly smaller model. By introducing a heuristic way to scale the model, EfficientNet provides a family of models (B0 to B7) that represents a good combination of efficiency and accuracy on a variety of scales. Such a scaling heuristics (compound-scaling, details see Tan and Le, 2019) allows the efficiency-oriented base model (B0) to surpass models at scale, while avoiding extensive grid-search of hyperparameters. A summary of the latest updates on the model is available at [here](https://arxiv.org/abs/1908.07973), where various augmentation schemes and semi-supervised learning approaches are applied to further improve the imagenet performance of the models.

## **CHAPTER 5**

### **SOFTWARE DESCRIPTION**

#### **5.1PYTHON**

##### **5.1.1 Introduction To Python**

Python is a high-level programming language known for its simplicity and readability. Created by Guido van Rossum and first released in 1991, Python emphasizes code readability and ease of use. It's an interpreted language, meaning code is executed line by line without needing to be compiled beforehand. Python's syntax is clean and straightforward, making it ideal for beginners and experienced programmers alike. It has a vast standard library and active community support, making it suitable for a wide range of applications, including web development, data analysis, machine learning, and automation.

Python is versatile and can be used for various purposes, including web development, data analysis, artificial intelligence, machine learning, automation, and scientific computing. It has a large and active community of developers who contribute to its ecosystem by creating libraries, frameworks, and tools for different domains.

Overall, Python is a powerful and popular programming language that is widely used in both academia and industry due to its simplicity, readability, and extensive capabilities.

### 5.1.2 Python Features

**1.Simple and Easy to Learn:** Python is designed to be easy to read and write, which makes it a great language for beginners. Its syntax is clean and straightforward, which reduces the cost of program maintenance and development.

**2.Interpreted Language:** Python is an interpreted language, meaning that it doesn't need to be compiled before execution. This makes the development process faster since you can immediately see the results of your code.

**3.High-Level Language:** Python abstracts many complex programming tasks, making it accessible to programmers with varying levels of experience. It provides built-in data structures and high-level constructs such as dictionaries, lists, and loops, which simplify coding.

**4.Dynamically Typed:** Python is dynamically typed, meaning you don't need to declare variable types explicitly. This allows for faster development, but it can also lead to potential errors if types are misused.

**5.Extensive Standard Library:** Python comes with a vast standard library that provides modules and packages for a wide range of tasks, from working with files to networking, web development, and more. This library reduces the need to write code from scratch and promotes code reuse.

**6.Portability:** Python is platform-independent, which means you can write code on one platform (e.g., Windows) and run it on another (e.g., Linux) without modification.

**Community and Ecosystem:** Python has a large and active community of developers who contribute to its ecosystem. There are thousands of third-party libraries and frameworks available for various purposes, making Python suitable for almost any programming task.

**7.Versatility:** Python is used in various domains, including web development, data analysis, machine learning, artificial intelligence, automation, scientific computing, and more. Its versatility makes it a popular choice for both hobbyists and professionals.

**8.Object-Oriented:** Python supports object-oriented programming (OOP) concepts such as classes and inheritance, allowing for the creation of reusable and modular code.

**9.Extensible and Embeddable:** Python can be extended with C/C++ code, allowing for performance-critical tasks to be implemented in these languages. It can also be embedded within other applications to provide scripting capabilities.

**10.Cross-platform:** Python is platform-independent, meaning that code written on one platform (e.g., Windows) can be easily executed on another platform (e.g., Linux or macOS) without modification.

### **5.1.3 Python Libraries**

**NumPy:** A fundamental package for scientific computing with Python, providing support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays.

**Pandas:** A powerful library for data manipulation and analysis. It provides data structures like DataFrame and Series, along with tools for reading and writing data from various file formats, such as CSV, Excel, SQL databases, and more.

**Matplotlib:** A plotting library for creating static, interactive, and animated visualizations in Python. It provides a MATLAB-like interface and supports a wide variety of plots and charts.

**Scikit-learn:** A simple and efficient library for machine learning tasks such as classification, regression, clustering, and dimensionality reduction. It integrates well with other scientific Python libraries like NumPy and SciPy.

**Seaborn:** Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn is built on top of matplotlib and closely integrated with pandas' data structures, making it easy to visualize data from DataFrames.

**Keras:** Keras is a powerful and versatile tool for building and training neural networks in Python. Whether you're a beginner or an experienced deep learning practitioner, Keras makes it easy to create and experiment with deep learning models for a wide range of applications.

#### 5.1.4 PYTHON APPLICATIONS

**Web Development:** Python is widely used for building web applications and websites. Frameworks like Django, Flask, and Pyramid provide developers with tools and patterns for quickly developing robust web applications.

**Data Science and Analytics:** Python is one of the leading languages for data analysis and machine learning. Libraries like NumPy, Pandas, Matplotlib, and scikit-learn are commonly used for data manipulation, visualization, statistical analysis, and machine learning tasks.

**Artificial Intelligence and Machine Learning:** Python is the preferred language for many AI and machine learning researchers and practitioners. Frameworks like TensorFlow, PyTorch, Keras, and scikit-learn provide tools and APIs for building and training neural networks, deep learning models, and other machine learning algorithms.

**Scientific Computing:** Python is widely used in scientific computing for tasks such as numerical simulations, mathematical modeling, and data visualization. Libraries like NumPy, SciPy, and matplotlib are essential tools for scientists and researchers in fields like physics, biology, chemistry, and engineering.

**Automation and Scripting:** Python's simplicity and readability make it an excellent choice for writing scripts and automating repetitive tasks. It's commonly used for tasks like system administration, file manipulation, data processing, and more.

**Desktop GUI Applications:** Python can be used to develop desktop applications with graphical user interfaces (GUIs). Libraries like Tkinter, PyQt, and wxPython provide tools for creating cross-platform GUI applications with ease.

**Game Development:** Python is used in game development for tasks like scripting, game logic, and rapid prototyping. Libraries like Pygame provide tools and APIs for creating 2D games, while frameworks like Panda3D and Godot support 3D game development.

**Web Scraping and Automation:** Python's libraries like BeautifulSoup and Scrapy are commonly used for web scraping, extracting data from websites, and automating web interactions.

**Internet of Things (IoT):** Python is used in IoT applications for tasks like sensor data processing, device control, and data analysis. Frameworks like Raspberry Pi and MicroPython provide support for developing IoT applications using Python.

**Education:** Python is widely used in education as a first programming language due to its simplicity and readability. It's used in introductory computer science courses, coding boot camps, and online tutorials to teach programming concepts to beginners.



## 5.2 DEEP LEARNING

### 5.2.1 INTRODUCTION TO DEEP LEARNING

Deep learning is a subfield of machine learning that focuses on algorithms inspired by the structure and function of the brain's neural networks. It's a type of artificial intelligence (AI) that enables computers to learn from large amounts of data and make decisions without explicit programming. It is inspired by the structure and function of the human brain's neural networks.

Deep learning represents a powerful approach to solving complex problems in AI and has revolutionized many fields with its ability to learn from data and extract meaningful patterns and representations. As research in deep learning continues to advance, we can expect to see even more impressive applications and breakthroughs in the future.

### 5.2.2 DEEP LEARNING LIBRARIES AND FRAMEWORKS

**TensorFlow:** Developed by Google Brain, TensorFlow is one of the most widely used deep learning frameworks. It offers a flexible and comprehensive ecosystem for building and training neural networks, with support for both research and production deployments.

TensorFlow provides high-level APIs like Keras for easier model development, as well as TensorFlow.js for running models in the browser and TensorFlow Lite for mobile and embedded devices.

**PyTorch:** Developed by Facebook's AI Research lab (FAIR), PyTorch is known for its simplicity and flexibility. It offers dynamic computational graphs, making it easier to debug models and experiment with different architectures. PyTorch provides a user-friendly interface and supports imperative programming, which allows for more

intuitive model development. It also offers torch vision for computer vision tasks and torch text for natural language processing.

**Keras:** Keras is a high-level neural networks API written in Python and capable of running on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK). It provides a user-friendly interface for building and training neural networks, enabling rapid prototyping and experimentation. Keras is now integrated into TensorFlow as its official high-level API.

### **5.2.3 DEEP LEARNING VS MACHINE LEARNING**

Deep learning is preferred over traditional machine learning in pneumonia detection because it can automatically learn complex patterns directly from medical images like chest X-rays. Unlike traditional methods, deep learning models, especially convolutional neural networks (CNNs), do not require manual feature extraction. They can analyse large datasets of labelled images to identify subtle patterns indicative of pneumonia, leading to higher accuracy and robustness in detecting the disease. Additionally, deep learning models are adaptable to variations in imaging techniques and patient demographics, making them suitable for diverse clinical settings. Overall, deep learning streamlines the detection process, improves accuracy, and enhances patient care in pneumonia diagnosis.

## CHAPTER 6

### RESULTS AND ANALYSIS

#### 6.1 DATASET DESCRIPTION

The dataset used in this project was sourced from Kaggle and is named the "Lung Disease". It consists of images of chest X-rays categorized into two main classes: "Normal" and "Pneumonia." The dataset is organized into three directories: "Lung\_Opacity", "Normal" and "Viral Pneumonia," each containing a subset for training and testing purposes.

##### Directory Structure:

**Lung\_Opacity:** Contains 1125 files categorized as Lung\_Opacity chest X-rays.

**Normal:** Contains 1250 files categorized as normal chest X-rays.

**Pneumonia:** Contains 1100 files categorized as pneumonia chest X-rays.

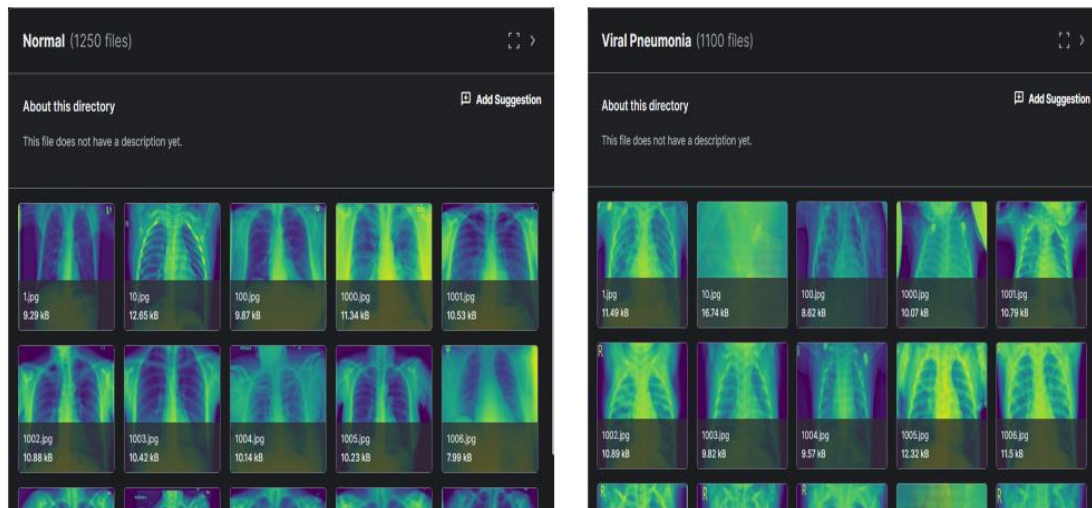
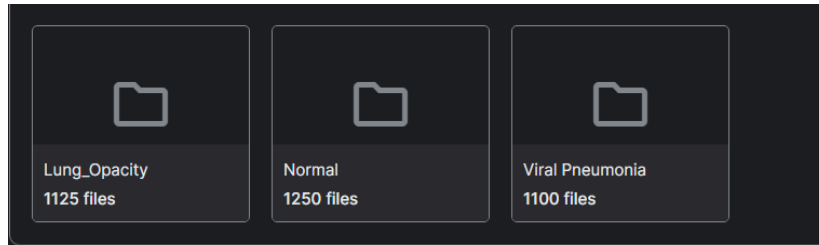


Figure 11 Images of normal Pneumonia

Each of these directories further contains subsets for training and testing.

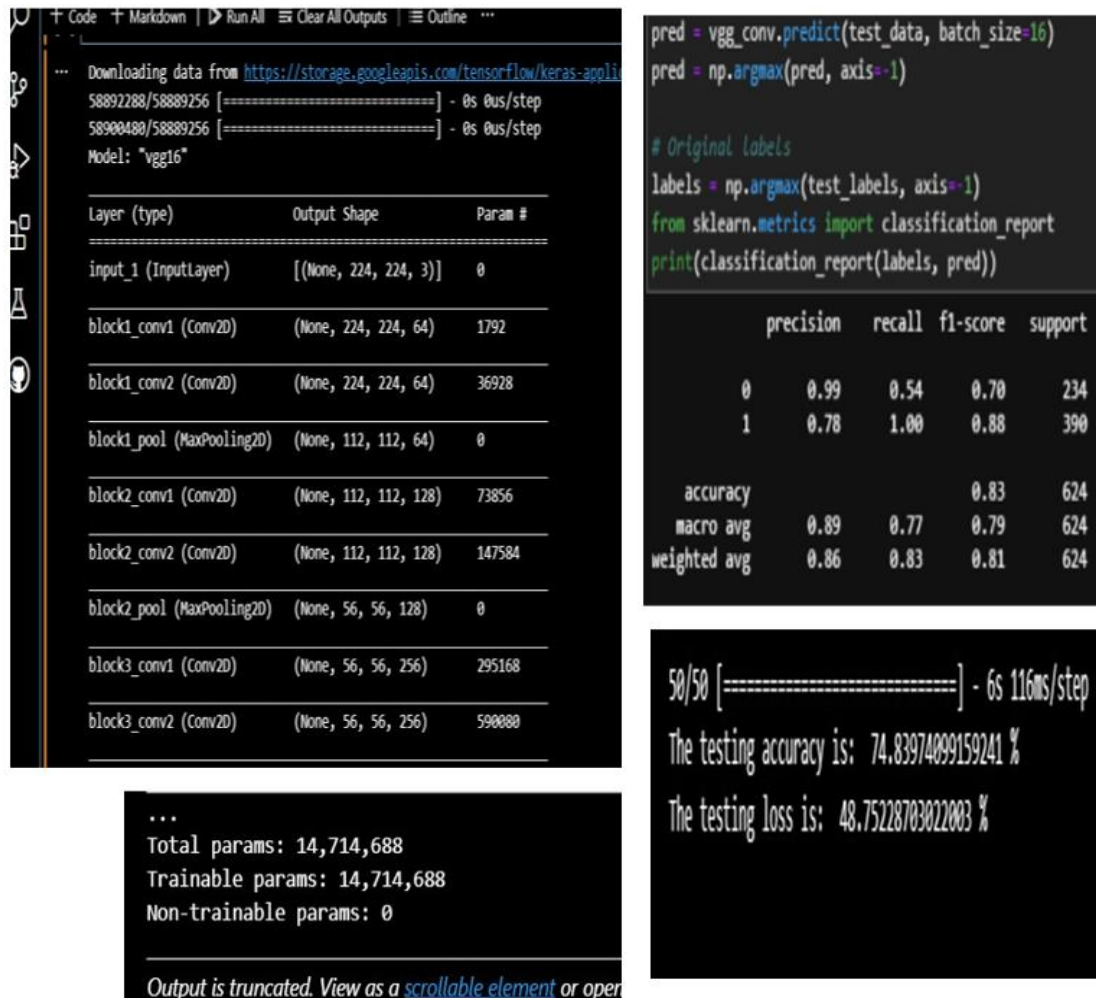


*Figure 12 Test, Train directories*

**Dataset Size:** The training subset comprises a total of 5318 images, including both normal and pneumonia chest X-rays. The testing subset includes 764 images reserved for evaluating the performance of the trained model.

## 6.2 IMPLEMENTATION DETAILS

### 6.2.1 Pretrained Models Implementation



... Downloading data from [https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16\\_data.zip](https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_data.zip) - 0s 0us/step  
58892288/58889256 [=====] - 0s 0us/step  
58900480/58889256 [=====] - 0s 0us/step  
Model: "vgg16"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080

...  
Total params: 14,714,688  
Trainable params: 14,714,688  
Non-trainable params: 0

Output is truncated. View as a [scrollable element](#) or open

```
pred = vgg_conv.predict(test_data, batch_size=16)
pred = np.argmax(pred, axis=-1)

# Original labels
labels = np.argmax(test_labels, axis=-1)
from sklearn.metrics import classification_report
print(classification_report(labels, pred))
```

	precision	recall	f1-score	support
0	0.99	0.54	0.70	234
1	0.78	1.00	0.88	390
accuracy			0.83	624
macro avg	0.89	0.77	0.79	624
weighted avg	0.86	0.83	0.81	624

50/50 [=====] - 6s 116ms/step  
The testing accuracy is: 74.83974099159241 %  
The testing loss is: 48.75228703022003 %

Figure 13 Implementation of VGG16

## INCEPTIONV3

```

# Networks
from keras.preprocessing import image
from keras.applications.inception_v3 import InceptionV3
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.imagenet_utils import preprocess_input

# Layers
from keras.layers import Dense, Activation, Flatten, Dropout, GlobalAveragePooling2D, BatchNormalization
from keras import backend as K
from keras.layers import Conv2D, SeparableConv2D, MaxPooling2D, LeakyReLU, Activation

print("\n%s: %.2f%%" % (InceptionV3_model.metrics_names[0], scores[0]*100))
print("\n%s: %.2f%%" % (InceptionV3_model.metrics_names[1], scores[1]*100))
print("\n%s: %.2f%%" % (InceptionV3_model.metrics_names[2], scores[2]*100))
print("\n%s: %.2f%%" % (InceptionV3_model.metrics_names[3], scores[3]*100))
print("\n%s: %.2f%%" % (InceptionV3_model.metrics_names[4], scores[4]*100))

loss: 48.20%

accuracy: 79.69%

auc: 84.95%

recall: 72.53%

precision: 78.12%

```

conv2d (Conv2D)	(None, 99, 99, 32)	864	input_1[0][0]
batch_normalization (BatchNormal	(None, 99, 99, 32)	96	conv2d[0][0]
activation (Activation)	(None, 99, 99, 32)	0	batch_normalization[0][0]
conv2d_1 (Conv2D)	(None, 97, 97, 32)	9216	activation[0][0]
batch_normalization_1 (BatchNor	(None, 97, 97, 32)	96	conv2d_1[0][0]
activation_1 (Activation)	(None, 97, 97, 32)	0	batch_normalization_1[0][0]
conv2d_2 (Conv2D)	(None, 97, 97, 64)	18432	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, 97, 97, 64)	192	conv2d_2[0][0]
activation_2 (Activation)	(None, 97, 97, 64)	0	batch_normalization_2[0][0]
max_pooling2d (MaxPooling2D)	(None, 48, 48, 64)	0	activation_2[0][0]
...			
Trainable params: 262,786			
Non-trainable params: 21,803,040			

```

+ Markdown | ▶ Run All | ≡ Clear All Outputs | ≡ Outline ...

base_model = InceptionV3(weights = InceptionV3_weights_path ,
| | | | | include_top = False,
| | | | | input_shape = (IMAGE_SIZE , IMAGE_SIZE , 3))

x = base_model.output
x = Dropout(0.5)(x)
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = BatchNormalization()(x)
predictions = Dense(2, activation='sigmoid')(x)

for layer in base_model.layers:
| layer.trainable = False

Metrics = ['accuracy', tf.keras.metrics.AUC(),tf.keras.metrics.Reca

```

Figure 14 Implementation of INCEPTIONV3

## RESNET

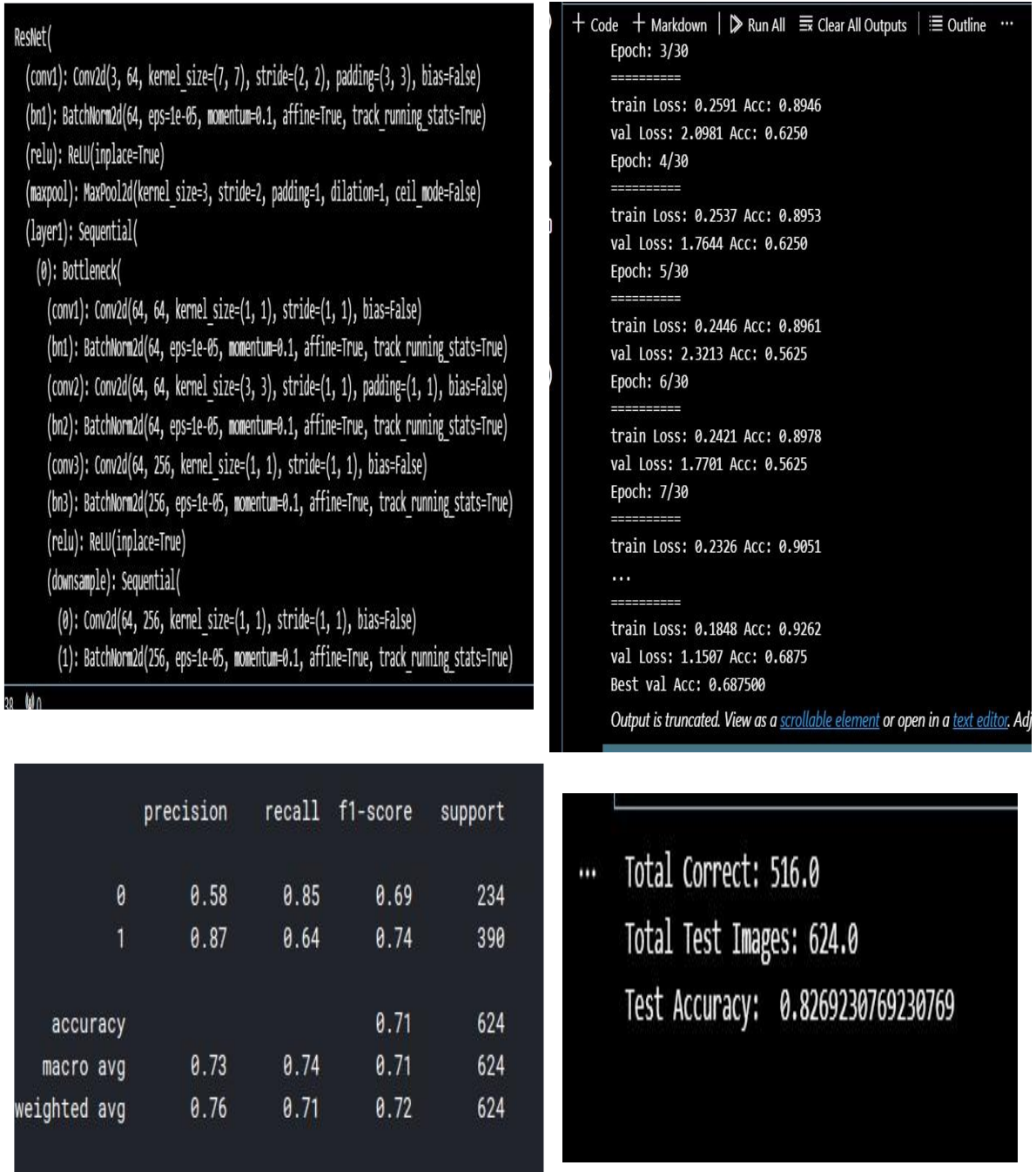


Figure 15 Implementation of ResNet

## 6.2.2 PROPOSED MODEL IMPLEMENTATION

### EFFICIENTNET

```
def enhance_image(image):  
    image = cv2.addWeighted(image, 1.5, image, -0.5, 0)  
  
    kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])  
    image = cv2.filter2D(image, -1, kernel)  
  
    hue = image[:, :, 0]  
    saturation = image[:, :, 1]  
    value = image[:, :, 2]  
    value = np.clip(value * 1.25, 0, 255)  
  
    image[:, :, 2] = value  
  
    return image
```

```
def show_images(image_gen):  
    test_dict = test.class_indices  
    classes = list(test_dict.keys())  
    images, labels = next(image_gen)  
    plt.figure(figsize=(20,20))  
    length = len(labels)  
    if length < 25:  
        r = length  
    else:  
        r = 25  
    for i in range(r):  
        plt.subplot(5,5,i+1)  
        image = (images[i]+1)/2  
        plt.imshow(image)  
        index = np.argmax(labels[i])  
        class_name = classes[index]  
        plt.title(class_name, color="green", fontsize=16)  
        plt.axis('off')  
    plt.show()  
  
show_images(train)
```



```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
from sklearn.model_selection import train_test_split
import tensorflow as tf
import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import cv2
from sklearn.metrics import (cohen_kappa_score, accuracy_score,
                             ConfusionMatrixDisplay, classification_report)
from mpl_toolkits.axes_grid1 import ImageGrid

```

```

print(df['labels'].value_counts())
df['labels'].value_counts().plot(kind='pie', autopct='%1.1f%%', cmap = 'coolwarm')

```

```

labels
Normal          1250
Lung Opacity     1125
Viral Pneumonia  1100
Name: count, dtype: int64
<Axes: ylabel='count'>

```

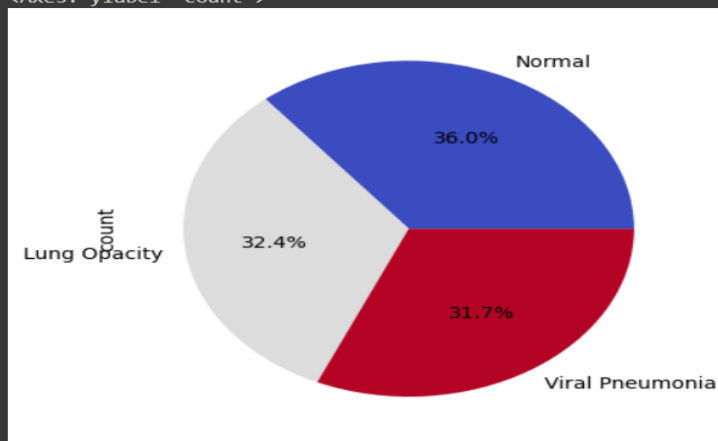


Figure 16 Implementation of EfficinetNet

## SIMULATION RESULT

```
history2 = model.fit(train,validation_data=val,epochs=25,batch_size=32,callbacks=[one_cycl])

Epoch 1/25
70/70 [=====] - 1236s 17s/step - loss: 0.4027 - accuracy: 0.8361 - val_loss: 1.8826 - val_accuracy: 0.3171
Epoch 2/25
70/70 [=====] - 1220s 17s/step - loss: 0.1755 - accuracy: 0.9409 - val_loss: 1.5607 - val_accuracy: 0.3171
Epoch 3/25
70/70 [=====] - 1200s 17s/step - loss: 0.1236 - accuracy: 0.9567 - val_loss: 3.2026 - val_accuracy: 0.3171
Epoch 4/25
70/70 [=====] - 1226s 18s/step - loss: 0.0935 - accuracy: 0.9639 - val_loss: 8.9361 - val_accuracy: 0.3223
Epoch 5/25
70/70 [=====] - 1237s 18s/step - loss: 0.1504 - accuracy: 0.9535 - val_loss: 2.2629 - val_accuracy: 0.4271
Epoch 6/25
70/70 [=====] - 1241s 18s/step - loss: 0.1244 - accuracy: 0.9549 - val_loss: 4.7022 - val_accuracy: 0.3171
Epoch 7/25
70/70 [=====] - 1252s 18s/step - loss: 0.1701 - accuracy: 0.9418 - val_loss: 6.2632 - val_accuracy: 0.3171
Epoch 8/25
70/70 [=====] - 1249s 18s/step - loss: 0.0936 - accuracy: 0.9779 - val_loss: 1.7688 - val_accuracy: 0.3887
Epoch 9/25
70/70 [=====] - 1239s 18s/step - loss: 0.0917 - accuracy: 0.9684 - val_loss: 4.4571 - val_accuracy: 0.3171
Epoch 10/25
70/70 [=====] - 1252s 18s/step - loss: 0.1649 - accuracy: 0.9481 - val_loss: 2.7737 - val_accuracy: 0.3274
Epoch 11/25
70/70 [=====] - 1249s 18s/step - loss: 0.0688 - accuracy: 0.9770 - val_loss: 0.6447 - val_accuracy: 0.8082
Epoch 12/25
70/70 [=====] - 1275s 18s/step - loss: 0.0839 - accuracy: 0.9765 - val_loss: 3.7992 - val_accuracy: 0.3171
Epoch 13/25
70/70 [=====] - 1266s 18s/step - loss: 0.0428 - accuracy: 0.9860 - val_loss: 2.1871 - val_accuracy: 0.3299
Epoch 14/25
70/70 [=====] - 1259s 18s/step - loss: 0.0403 - accuracy: 0.9874 - val_loss: 3.2905 - val_accuracy: 0.6343
Epoch 15/25
70/70 [=====] - 1238s 18s/step - loss: 0.0147 - accuracy: 0.9959 - val_loss: 15.8366 - val_accuracy: 0.3171
Epoch 16/25
70/70 [=====] - 1237s 18s/step - loss: 0.0370 - accuracy: 0.9914 - val_loss: 0.9927 - val_accuracy: 0.7417
```

Figure 17 Result of EfficientNetB1 Model

## Pneumonia

```
[38] image_path = '/content/Lung X-Ray Image/Lung X-Ray Image/Viral Pneumonia/1005.jpg'
is_viral_pneumonia = predict_viral_pneumonia(image_path)
print(is_viral_pneumonia)

1/1 [=====] - 0s 327ms/step
1
```

## Normal

```
image_path = '/content/Lung X-Ray Image/Lung X-Ray Image/Normal/1002.jpg'
is_viral_pneumonia = predict_viral_pneumonia(image_path)
print(is_viral_pneumonia)

1/1 [=====] - 10s 10s/step
0
```

Figure 18 Predictions of EfficientNet

```

Accuracy of test data ||||| 0.9781357882623706
+++++
Kappa Score of test data ||||| 0.9671374490969814
+++++

```

	precision	recall	f1-score	support
Lung Opacity	0.99	0.96	0.97	281
Normal	0.96	0.98	0.97	313
Viral Pneumonia	0.99	1.00	0.99	275
accuracy			0.98	869
macro avg	0.98	0.98	0.98	869
weighted avg	0.98	0.98	0.98	869

```

+++++

```

Figure 19 Parameters of proposed

## 6.3 PARAMETERS

		True Class	
		Positive	Negative
Predicated Class	Positive	TP	FP
	Negative	FN	TN

	positive	negative
positive	274 ( TP )	1 ( FP )
negative	6 ( FN )	307 ( TN )

**Precision:** It measures the percentage of predictions made by the model that are correct.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{precision} = \frac{274}{274+1} = 0.996363$$

**Recall:** It measures the percentage of relevant data points that were correctly identified by the model.

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Recall} = \frac{307}{307+6} = 0.980830$$

**F1 Score:** It is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model.

$$F1=2 ((\text{precision} \times \text{recall})/(\text{precision} + \text{recall}))$$

$$F1=2((0.99 \times 1.00) / (0.99+1.00)) = 0.9963636$$

**Accuracy:** It is a Metrix that computes how many times a model made a correct prediction across the entire dataset.

$$\text{Accuracy} = (\text{Correct predictions})/(\text{All predictions})$$

$$\text{Accuracy} = 274/ 275=0.996363$$

	precision	recall	f1-score	support
Lung Opacity	0.99	0.96	0.97	281
Normal	0.96	0.98	0.97	313
Viral Pneumonia	0.99	1.00	0.99	275
accuracy			0.98	869
macro avg	0.98	0.98	0.98	869
weighted avg	0.98	0.98	0.98	869

## 6.4 PERFORMANCE OF DIFFERENT MODELS

**PRECISION, RECALL, F1 SCORE:**

MODEL	PRECISION	RECALL	F1 SCORE
<b>VGG16</b>	0.78	0.54	0.70
<b>RESNET</b>	0.58	0.85	0.69
<b>INCEPTIONV3</b>	0.78	0.72	0.54
<b>PROPOSED MODEL</b>	0.99	0.98	0.98

*Table 2Parameters of different models*

### ACCURACY IN DIFFERENT MODELS:

	VGG16	INCEPTIONV3	RESNET	PROPOSED MODEL
<b>PARAMETERS</b>	14714688	22070826	23837441	1246401
<b>ACCURACY</b>	74.83	79.69	82.69	98
<b>LOSS</b>	48.75	48.20	28.87	41.73

*Table 3 Accuracy in different Models*

## 6.5 PERFORMANCE OF EXISTING AND PROPOSED MODELS

### PERFORMANCE

#### Performance of Existing Model :

Epoch Stage	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1	0.7893	0.4567	0.8526	0.3088
5	0.9385	0.1649	0.8125	0.4679
10	0.9515	0.1246	0.9022	0.2762
15	0.9519	0.1265	0.9231	0.2079
20	0.9561	0.1139	0.9103	0.2806
25	0.9651	0.0962	0.9247	0.2290

#### Performance of Proposed Model:

Epoch Stage	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1	0.8361	0.4027	0.3171	1.8826
5	0.9535	0.1755	0.4271	1.7688
10	0.9481	0.1236	0.3274	2.1871
15	0.9959	0.0147	0.3171	0.9927
20	0.9991	0.0030	0.9105	0.4504
25	1.0000	1.0674e-04	0.9463	0.3739

*Table 4 Performance of Existing model and Proposed model*

The project's base paper underscores the effectiveness of a deep learning model in accurately classifying pneumonia from Chest X-rays. Despite its success, the reliance on 25 epochs for optimal accuracy poses limitations in terms of computational complexity. To address this, the current project also have training epochs to 25 while maintaining comparable accuracy levels. This optimization enhances efficiency and scalability, making the model more suitable for real-world deployment, where timely diagnosis is critical. By improving computational efficiency without compromising accuracy, the project contributes to early and effective pneumonia diagnosis, ultimately improving patient outcomes in the medical field. 25 Epochs were used to train the model with the batch sizes being 32 and 1 for training and testing, respectively.

## 6.6 GRAPHS FOR ACCURACY AND LOSS

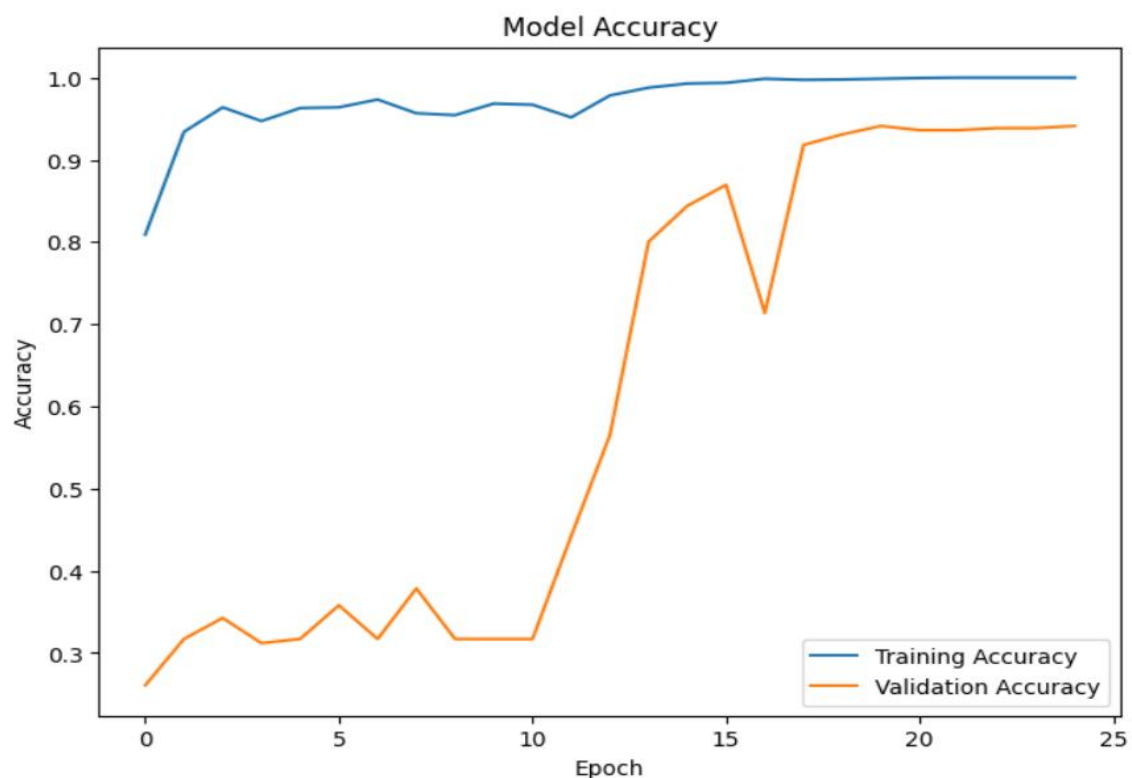


Figure 20 Graphs for Accuracy

**Training Accuracy:** This refers to the accuracy of a machine learning model on the dataset that it was trained on. It's calculated by comparing the predictions made by the model to the actual labels in the training data. Essentially, it measures how well the model fits the training data.

Training Accuracy obtained by using our model is around 1.00

**Validation Accuracy:** This refers to the accuracy of a machine learning model on a separate dataset called the validation set. The validation set is used to evaluate the performance of the model during training and to tune hyperparameters. The validation accuracy gives an indication of how well the model generalizes to new, unseen data. It helps in assessing whether the model is overfitting or underfitting. Validation Accuracy obtained by using our model is around 0.94.

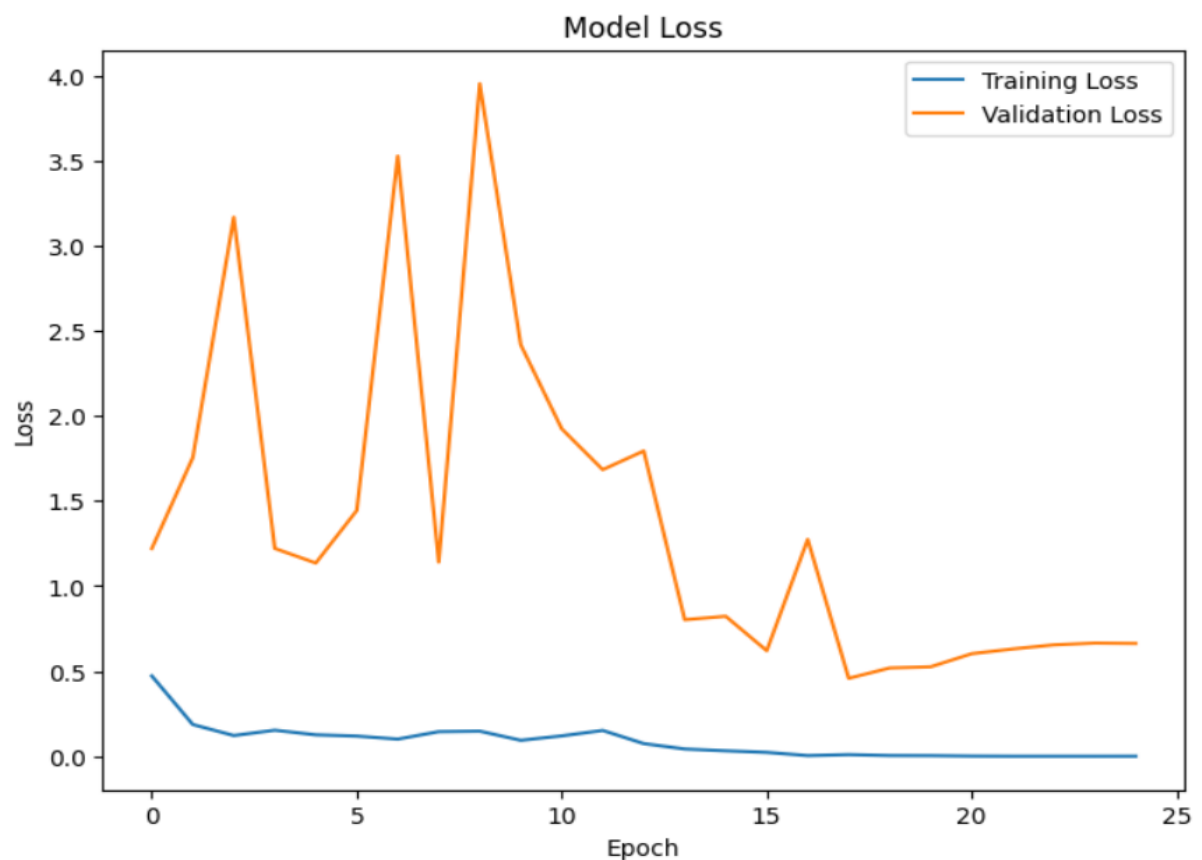


Figure 21 Graphs for Loss

**Training Loss:** The training loss is a metric used to assess how a deep learning model fits the training data. That is to say, it assesses the error of the model on the training set.

Note that, the training set is a portion of a dataset used to initially train the model.

Training Loss obtained by using our model is around 1.00.

**Validation Loss:** The validation set is a portion of the dataset set aside to validate the performance of the model. The validation loss is similar to the training loss and is calculated from a sum of the errors for each example in the validation set.

Validation Loss obtained by using our model is around 0.66.



## **CHAPTER 7**

### **CONCLUSION**

#### **7.1CONCLUSION**

Convolutional Neural Networks (CNNs) for Pneumonia detection is to develop a robust and accurate system for automatically identifying and classifying lung infection from medical images such as X-rays, CT scans, or MRI scans.

In conclusion, achieving a 98% accuracy by epoch 25, contributes to early and effective pneumonia diagnosis, marks a significant advancement in our project's efficiency compared to previous endeavours where the same benchmark was reached by epoch 25 with 92% accuracy.

#### **7.2FUTURE SCOPE**

We will try this approach on more datasets in the future and compare these results to that of human experts. It is envisaged that in the future, models using transfer learning technologies that will undergo training on this dataset would show better outcomes than this CNN model.

## CHAPTER 8

### REFERENCES

- [1] H. Wang and Y. Xia, “Chest Net: A Deep Neural Network for Classification of Thoracic Diseases on Chest Radiography,” pp. 1–8, 2018, [Online]. Available: <http://arxiv.org/abs/1807.03058>
- [2] S. Shah, H. Mehta, and P. Sonawane, Pneumonia detection using convolutional neural networks, no. May. Springer Singapore, 2020. doi: 10.1109/ICSSIT48917.2020.9214289.
- [3] O. P. Yadav, “Chronic Disease Detection Using Deep Learning,” no. May 2021, 2022.
- [4] R. Kundu, R. Das, Z. W. Geem, G. T. Han, and R. Sarkar, “Pneumonia detection in chest X-ray images using an ensemble of deep learning models,” PLoS One, vol. 16, no. 9 September, 2021, doi: 10.1371/journal.pone.0256630.
- [5] K. M. Kuo, P. C. Talley, C. H. Huang, and L. C. Cheng, “Predicting hospital-acquired pneumonia among schizophrenic patients: A machine learning approach,” BMC Med. Inform. Decis. Mak., vol. 19, no. 1, pp. 1–8, 2019, doi: 10.1186/s12911-019-0792-1.
- [6] H. Yue et al., “Machine learning-based CT radiomics method for predicting hospital stay in patients with pneumonia associated with SARS-CoV-2 infection: a multicentre study,” Ann. Transl. Med., vol. 8, no. 14, pp. 859–859, 2020, doi: 10.21037/atm-20-3026.
- [7] K. T. Islam, S. Wijewickrema, A. Collins, and S. O’Leary, “A deep transfer learning framework for pneumonia detection from chest X-ray images,” VISIGRAPP 2020 -

Proc. 15th Int. Jt. Conf. Compute. Vision, Imaging Compute. Graph. Theory Appl., vol. 5, no. Visigrapp 2020, pp. 286–293, 2020, doi: 10.5220/0008927002860293.

[8] D. Varshni, K. Thakral, L. Agarwal, R. Nijhawan, and A. Mittal, “Pneumonia Detection Using CNN based Feature Extraction,” Proc. 2019 3rd IEEE Int. Conf. Electr. Comput. Commun. Technol. ICECCT 2019, pp. 1–7, 2019, doi: 10.1109/ICECCT.2019.8869364.

[9] M. B. Khan, M. T. Islam, and M. Ahmad, “A CNN-based CADx Model for Pneumonia Detection from Chest Radiographs with Web Application,” 2021 Int. Conf. Sci. Contemp. Technol. ICSCT 2021, no. December, 2021, doi: 10.1109/ICSCT53883.2021.9642603.

[10] L. H. Son, A. Kumar, S. R. Sangwan, A. Arora, A. Nayyar, and M. Abdel-Basset, “Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network,” IEEE Access, vol. 7, pp. 23319–23328, 2019, doi: 10.1109/ACCESS.2019.2899260.

[11] F. Saeed, A. Paul, P. Karthigaikumar, and A. Nayyar, “Convolutional neural network based early fire detection,” Multimed. Tools Appl., vol. 79, no. 13–14, pp. 9083–9099, 2020, doi:10.1007/s11042-019-07785-w.

[12] K. Eckle and J. Schmidt-Hieber, “A comparison of deep networks with ReLU activation function and linear spline-type methods,” Neural Networks, vol. 110, pp. 232–242, 2019, doi:10.1016/j.neunet.2018.11.005.

[13] T. Rajasenbagam, S. Jeyanthi, and J. A. Pandian, “Detection of pneumonia infection in lungs from chest X-ray images using deep convolutional neural network and content-based image retrieval techniques,” J. Ambient Intell. Humaniz. Comput., no. 2016, 2021, doi: 10.1007/s12652-021-03075-2