

Pneumonia Detection using Deep Learning - CNN

By

Andrew John 18BEC1278

N. T. Srihari 18BEC1180

R. Vignesh 18BEC1064

Vighnesh M 18BEC1223

A project report submitted to

Dr. Sucheta M

SCHOOL OF ELECTRONICS ENGINEERING (SENSE)

in partial fulfilment of the requirements for the course of

ECE3009 - Neural Networks and Fuzzy Control

in

B.Tech. Electronics and Communications Engineering



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

VIT CHENNAI, Vandalur - Kelambakkam Road

Chennai – 600127

December - 2021

BONAFIDE CERTIFICATE

Certified that this project report entitled “**Pneumonia Detection using Deep Learning - CNN**” is a bonafide work of **Andrew John - 18BEC1278, N.T Srihari - 18BEC1180, R.Vignesh - 18BEC1064, and Vighnesh M - 18BEC1223** who carried out the Project work under my supervision and guidance for **ECE3009- Neural Networks and Fuzzy Control**

Dr. Sucheta M

Associate Professor

School of Electronics Engineering (SENSE),

VIT University, Chennai

Chennai – 600 127.

ABSTRACT

Pneumonia accounts for 14% of all deaths of children under 5 years old, killing 740,180 children in 2019. About 1 million adults in the US seek care in a hospital due to pneumonia every year, and 50,000 die from this disease. than the age extremes, it is always a threat. Half of all non-immunocompromised adults hospitalized for severe pneumonia in the US are younger adults (18-57 years of age). According to the World Health Organization (WHO), the most common diagnosis for severe COVID-19 is severe pneumonia.

In China, doctors classified 81% of COVID-19 cases as mild. These mild infections include mild cases of pneumonia. The remaining 19% of cases were more severe. In the United States, 1.5 million people were diagnosed with pneumonia in an emergency department during 2018. Unfortunately, more than 40,000 people died from the disease that year in the United States. Most of the people affected by pneumonia in the United States are adults.

We propose a network intrusion detection model using Convolutional Neural Networks (CNNs). As CNN's perform well with image data we convert the raw traffic vector format into image format. In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. The dataset that we have used has been obtained from Kaggle. The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal).

ACKNOWLEDGEMENT

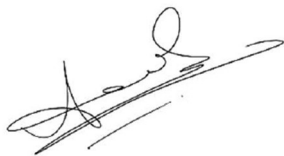
We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Sucheta M.**, Associate Professor, School of Electronics Engineering, for her consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Sivasubramanian. A.**, Dean of School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for her unstinting support.

We express our thanks to our Head of the Department **Dr. Vetrivelan. P** & **Dr. D Thirupusundari** for their support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

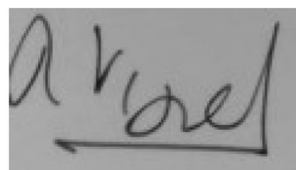
We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity; they provided us in undergoing this course in such a prestigious institution.



ANDREW



SRIHARI



VIGNESH



VIGHNESH

TABLE OF CONTENTS

Ch No.			Topic	Page No.
			Abstract	3
			Acknowledgment	4
1			INTRODUCTION	6
	1.1		<i>Objective</i>	6
	1.2		<i>Motivation</i>	6
	1.3		<i>Features</i>	6
	1.4		<i>Applications</i>	6
2			LITERATURE REVIEW	9
3			DESIGN & IMPLEMENTATION	12
	3.1		<i>About Software, Dataset and Pneumonia</i>	12
		3.1.1	Software Used	12
		3.1.2	Dataset Used	14
		3.1.3	About Pneumonia, Symptoms and Cure	15
	3.2		<i>Convolutional Neural Network</i>	16
		3.2.1	Definition	16
		3.2.2	Architecture	16
		3.2.3	Layers in CNN	17
		3.2.4	Pooling	18
	3.3		<i>Algorithm</i>	18
	3.4		<i>Coding and Implementation</i>	20
4			RESULTS & DISCUSSION	27
5			CONCLUSION	29
	5.1		<i>Conclusion</i>	29
	5.2		<i>Future Scope</i>	29
6			REFERENCES	30
7			BIODATA	31

CHAPTER - I

INTRODUCTION

1.1 Objective

The goal of this project is to develop a predictive model that uses Convolutional Neural Networks (CNN) to determine whether a person has pneumonia or not based on the patient's chest x-ray images. The goal of the project is to achieve high accuracy while minimising loss, ensuring that the model does neither overfit nor underfit.

1.2 Motivation

The goal of this project is to automate the process of determining whether or not a person has pneumonia based on their chest x-ray picture, hence enhancing the speed and accuracy of the diagnosis. Because pneumonia can render a person immunocompromised and immunosuppressed, they are more likely to get infected with COVID - 19, which damages the lungs similarly to pneumonia. The same is true when a person has COVID-19, which makes them more susceptible to pneumonia.

1.3 Features

The features or novelty of the project are,

- I. CNN is performed on two-dimensional / image data,
- II. CNN is used on non-image data by converting its one-dimensional attributes into 2D image data,
- III. CNN is computationally efficient. It uses special convolution and pooling operations and performs parameter sharing. This enables CNN models to run on any device, making them universal
- IV. The model will be trained on a huge dataset (over 100k data points),
- V. CNN parameters can be fine-tuned (number of convolutional/pooling layers, filter size, initial weights etc.) to get optimal results and
- VI. This model will also address the dataset imbalance problem that previous models did not.

1.4 Applications

Simple applications of CNNs which we can see in everyday life are obvious choices, like facial recognition software, image classification, speech recognition programs, etc. These are terms which we, as laymen, are familiar with, and comprise a major part of our everyday life, especially with image-savvy social media networks like Instagram. Some of the key applications of CNN are listed here -

1. *Facial-Recognition:*

Facial recognition is broken down by a convolutional neural network into the following major components -

- Identifying every face in the picture
- Focusing on each face despite external factors, such as light, angle, pose, etc.
- Identifying unique features
- Comparing all the collected data with already existing data in the database to match a face with a name.

2. *Analyzing-Documents:*

Convolutional neural networks can also be used for document analysis. This is not just useful for handwriting analysis, but also has a major stake in recognizers. For a machine to be able to scan an individual's writing, and then compare that to the wide database it has, it must execute almost a million commands a minute. It is said with the use of CNNs and newer models and algorithms, the error rate has been brought down to a minimum of 0.4% at a character level, though it's complete testing is yet to be widely seen.

3. *Historic-and-Environmental-Collections:*

CNNs are also used for more complex purposes such as natural history collections. These collections act as key players in documenting major parts of history such as biodiversity, evolution, habitat loss, biological invasion, and climate change.

4. *Understanding Climate:*

CNNs can be used to play a major role in the fight against climate change, especially in understanding the reasons why we see such drastic changes and how we could experiment in curbing the effect. It is said that the data in such natural history collections can also provide greater social and scientific insights, but this would require skilled human resources such as researchers who can physically visit these types of repositories. There is a need for more manpower to carry out deeper experiments in this field.

5. *Grey-Areas:*

Introduction of the grey area into CNNs is posed to provide a much more realistic picture of the real world. Currently, CNNs largely function exactly like a machine, seeing a true and false value for every question. However, as humans, we understand that the real world plays out in a thousand shades of grey. Allowing the machine to understand and process fuzzier logic will help it understand the grey area as humans live in and strive to work against. This will help CNNs get a more holistic view of what humans see.

6. *Advertising:*

CNNs have already brought in a world of difference to advertising with the introduction of programmatic buying and data-driven personalized advertising.

7. *Other Interesting Fields:*

CNNs are poised to be the future with their introduction into driverless cars, robots that can mimic human behavior, aids to human genome mapping projects, predicting earthquakes and natural disasters, and maybe even self-diagnoses of medical problems. So, you wouldn't even have to drive down to a clinic or schedule an appointment with a doctor to ensure your sneezing attack or high fever is just the simple flu and not symptoms of some rare disease. One problem that researchers are working on with CNNs is brain cancer detection. The earlier detection of brain cancer can prove to be a big step in saving more lives affected by this illness.

CHAPTER - II

LITERATURE REVIEW

Luka *et al.* [1] describes the use of machine learning algorithms to process chest X-ray images in order to support the decision making process in determining the correct diagnosis whether the patient is affected with pneumonia or not. The implementation was based on CNN model using Python programming and scientific tools. The activation function used in these experiments was the ReLU activation function. ReLU behaves very well in deep neural networks because it is fast to compute. With the trained model, 334 out of 381 were accurately predicted as images of X-rays with pneumonia, while 187 out of 205 were accurately predicted as X-rays without pneumonia and the model gave an accuracy of 88.90%. Islam and Arefin [2] presents an automated technique of detecting the skin disease using computer algorithms and image processing. The given paper proposes a model that can gather PSL, perform it's analysis. The system uses image processing methods. It can analyse, create databases, find features, process the information and identify the disease. All this is one on features based on the texture of the skin and other morphological features. The input images were fed to see the output. The system was able to almost accurately identify and detect arsenic in the images. Logesh *et al* [3] analyze the detection of heart disease using machine learning algorithms and python programming. Their main objective of the paper is to get better accuracy to detect heart-disease using algorithms in which the target output counts whether a person has heart disease or not. The paper represented in, is Compared with KNN, SVM, Random classifier, decision tree classifier given accurate result for Heart Disease Prediction System- HDPS. The prediction was made with a better accuracy of 98.83% by the decision tree machine learning method. In this, the Support Vector Machine algorithm classifies the data values by using hyper planes and the decision tree is implemented by Gini index method in which the highest gain of the attributes gives a better representation of the decision tree algorithm with accuracy of 82.6%.

Gabruseva *et al* [4] developed a computational approach for pneumonia regions detection based on single-shot detectors, squeeze-and-extinction deep convolutional neural networks, augmentations and multi-task learning. They proposed a simple and effective algorithm for the localization of lung opacities regions. The model was based on single-shot detector RetinaNet with Se-ResNext101 encoders, pre-trained on ImageNet dataset. The results of detection models can change significantly between epochs and depend largely on thresholds. The model was based on RetineNet SSD with Se-ResNext101 encoders pre-trained on ImageNet dataset, heavy augmentations with custom rotation, multi-task learning with global classification, and postprocessing. For the final ensemble, the outputs from the same model for 4 cross-validation folds and several checkpoints were combined before applying NMS algorithms. The postprocessing with re-scaling predictions was applied to compensate for the difference between the train and test sets labelling processes. The training dataset included data for 25684 patients and the test set had data for 1000 patients. Pranav *et al* [5] develop an algorithm that can detect pneumonia from chest X-rays at a level exceeding practicing radiologists. Their algorithm, CheXNet, is a 121-layer convolutional neural network trained on ChestX-ray14, currently the largest publicly available chest X-ray dataset, containing over 100,000 frontal view X-ray images with 14 diseases. They extend CheXNet to detect all 14 diseases in ChestX-ray14 and achieve state of the art results on all 14 diseases.

Their model, ChexNet, is a 121-layer convolutional neural network that inputs a chest X-ray image and outputs the probability of pneumonia along with a heatmap localizing the areas of the image most indicative of pneumonia. Limitations: First, only frontal radiographs were presented to the radiologists and model during diagnosis, but it has been shown that up to 15% of accurate diagnoses require the lateral view, neither the model nor the radiologists were not permitted to use patient history, which has been shown to decrease radiologist diagnostic performance in interpreting chest radiographs.

Turbe *et al* [6] have used deep learning to classify images of rapid human immunodeficiency virus (HIV) tests acquired in rural South Africa. Using newly developed image capture protocols with the Samsung SM-P585 tablet, 60 fieldworkers routinely collected images of HIV lateral flow tests. From a library of 11,374 images, deep learning algorithms were trained to classify tests as positive or negative. A pilot field study of the algorithms deployed as a mobile application demonstrated high levels of sensitivity (97.8%) and specificity (100%) compared with traditional visual interpretation by humans—experienced nurses and newly trained community health worker staff—and reduced the number of false positives and false negatives. Gonclaves *et al* [7] have performed a systematic review related to applications of deep learning in gastric tissue disease analysis by digital histology, endoscopy and radiology images. This review highlighted the high potential and shortcomings in deep learning research studies applied to gastric cancer, ulcer, gastritis and non-malignant diseases. Our results demonstrate the effectiveness of gastric tissue analysis by deep learning applications. Moreover, we also identified gaps of evaluation metrics, and image collection availability, therefore, impacting experimental reproducibility. Asnaoui *et al* [8] presented a comparison of recent deep convolutional neural network (CNN) architectures for automatic binary classification of pneumonia images based on fine-tuned versions of (VGG16, VGG19, DenseNet201, Inception_ResNet_V2, Inception_V3, Resnet50, MobileNet_V2 and Xception) and a retraining of a baseline CNN. They conclude that the fine-tuned version of Resnet50 shows highly satisfactory performance with rate of increase in training and testing accuracy (more than 96% of accuracy).

Sharma *et al* [9] presents a novel approach for detecting the presence of pneumonia clouds in chest X-rays (CXR) by using only Image processing techniques. For this, we have worked on 40 analog chest CXRs pertaining to Normal and Pneumonia infected patients. To detect pneumonia clouds we have used Otsu thresholding which will segregate the healthy part of the lung from the pneumonia infected cloudy regions. We are proposing to compute the ratio of area of healthy lung region to total lung region to establish a result. Li Feng *et al* [10] took standard chest radiographs in 36 patients with 46 focal airspace opacities due to pneumonia (10 patients had bilateral opacities) and 20 patients without focal opacities were included in an observer study. A bone suppression image processing system was applied to the 56 radiographs to create corresponding bone suppression images. In the observer study, eight observers, including six attending radiologists and two radiology residents, indicated their confidence level regarding the presence of a focal opacity compatible with pneumonia for each lung, first by use of standard images, then with the addition of bone suppression images. Receiver operating characteristic (ROC) analysis was used to evaluate the observers' performance. The mean value of the area under the ROC curve (AUC) for eight observers was significantly improved from 0.844 with use of standard images alone to 0.880 with standard plus bone suppression images ($P < 0.001$) based on 46 positive lungs and 66 negative lungs.

Tilve *et al* [11] focuses on surveying and comparing the detection of lung disease using different computer-aided techniques and suggests a revised model for detecting pneumonia, which will then be implemented as part of their future research. They used image pre-processing techniques to convert raw X-ray images into standard formats for analysis and detection, machine learning techniques such as CNN, RESNET, CheXNet, DENSENET, ANN and KNN, which is an important phase in accurate pneumonia detection. There are several approaches used to detect lung diseases using computer-aided diagnosis but techniques using machine learning algorithms have proved to be more reliable.

CHAPTER - III

DESIGN & IMPLEMENTATIONS

3.1 About Software, Dataset and Pneumonia

3.1.1 The Software used:

Google Colab and *Jupyter Notebook* are the python coding platforms used.

Some of the important python libraries used are TensorFlow, pandas, NumPy, Random, OS, Matplotlib, Seaborn, OpenCV, and Keras.

- **TensorFlow:** TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. Tensorflow is a symbolic math library based on dataflow and differentiable programming.
- **pandas:** pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
- **NumPy:** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- **Random:** Python Random module is an in-built module of Python which is used to generate random numbers. These are pseudo-random numbers meaning these are not truly random. This module can be used to perform random actions such as generating random numbers, print random a value for a list or string, etc
- **OS:** The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality.
- **Matplotlib:** Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.
- **Seaborn:** Seaborn is an **open-source Python library built on top of matplotlib**. It is used for data visualization and exploratory data analysis. Seaborn works easily with dataframes and the Pandas library. The graphs created can also be customized easily.
- **OpenCV:** OpenCV-Python is a library of Python bindings designed to solve computer vision problems. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

- **Keras:** Keras runs on top of open source machine libraries like TensorFlow. Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow.

3.1.2 The Dataset Used:

Chest X-Ray Images (Pneumonia) Dataset:

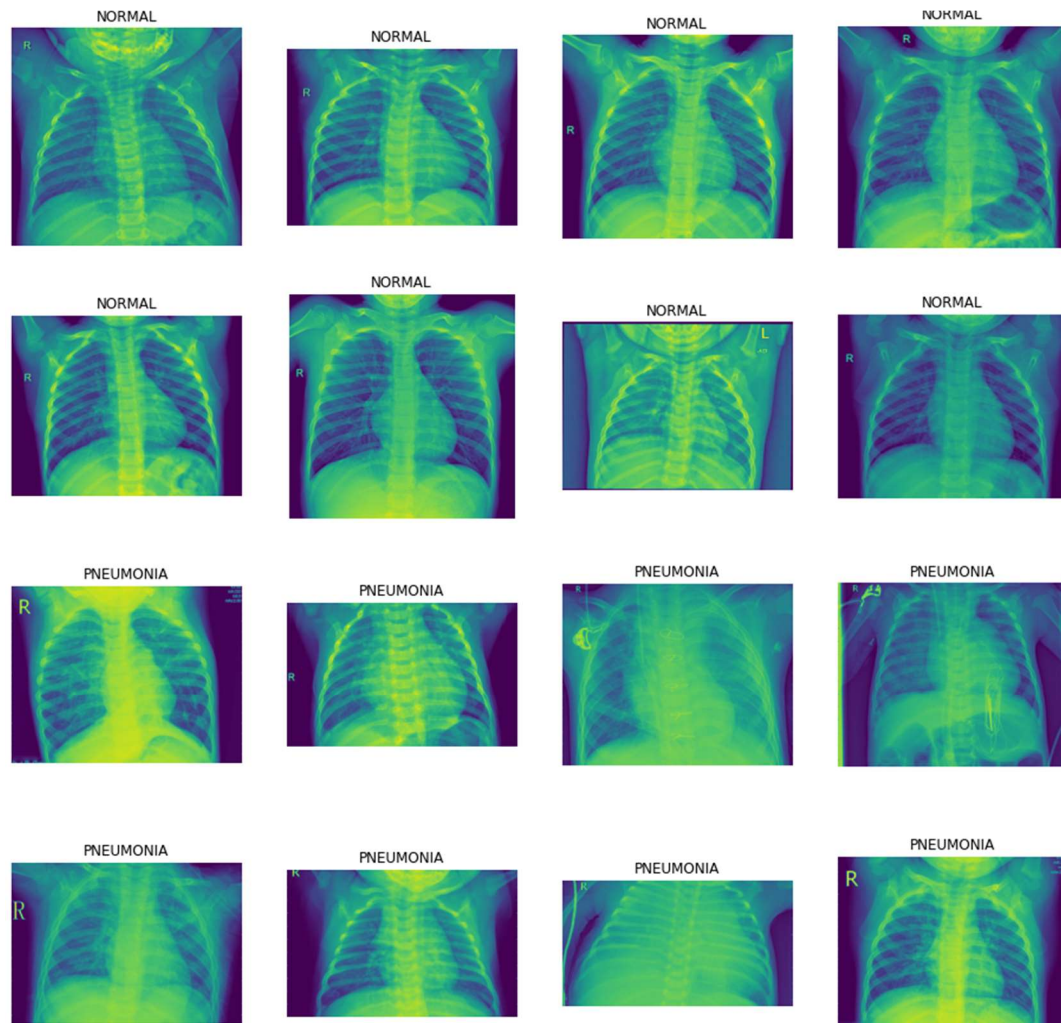


Fig-1: *The dataset used for Pneumonia Detection*

The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal). Chest X-ray images (anterior-posterior) were selected from retrospective cohorts of pediatric patients of one to five years old from Guangzhou Women and Children's Medical Center, Guangzhou. All chest X-ray imaging was performed as part of patients' 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.

3.1.3 About Pneumonia, Symptoms and Cure:

Pneumonia is a lung infection that causes the air sacs in one or both lungs to become inflamed. Pneumonia is caused by several organisms, including bacteria, viruses, and fungi, filling the air sacs with fluid or pus (purulent material), resulting in a cough with phlegm or pus, fever, chills, and trouble breathing. The severity of pneumonia can range from minor to life-threatening. Infants and young children, persons over the age of 65, and people with health issues or compromised immune systems are the most vulnerable.

Symptoms:

Pneumonia symptoms range from moderate to severe, depending on the type of germ that caused the illness, as well as your age and overall health. Mild signs and symptoms are sometimes mistaken for those of a cold or the flu, but they stay much longer.

The following are some of the signs and symptoms of pneumonia:

- Chest pain when you breathe or cough
- Confusion or changes in mental awareness (in adults age 65 and older)
- Cough, which may produce phlegm
- Fatigue
- Fever, sweating and shaking chills
- Lower than normal body temperature (in adults older than age 65 and people with weak immune systems)
- Nausea, vomiting or diarrhoea
- Shortness of breath

The illness may be undetected in new-borns and infants. Alternatively, they may vomit, have a fever and cough, seem restless or exhausted and depleted of energy, or have breathing and feeding difficulties.

When should you see a doctor?

If you have trouble breathing, chest discomfort, a persistent fever of 102 degrees Fahrenheit (39 degrees Celsius), or a chronic cough, especially if you're coughing up pus, see your doctor.

People in these high-risk categories should consult a doctor more frequently:

- Adults over 65 years old
- Signs and symptoms in children under the age of two
- People who have a compromised immune system or have an underlying health issue
- People who are undergoing chemotherapy or using immune-suppressing medications

Pneumonia can swiftly become a life-threatening illness in some elderly persons and people with heart failure or persistent lung difficulties.

3.2 Convolutional Neural Network:

3.2.1 Definition:

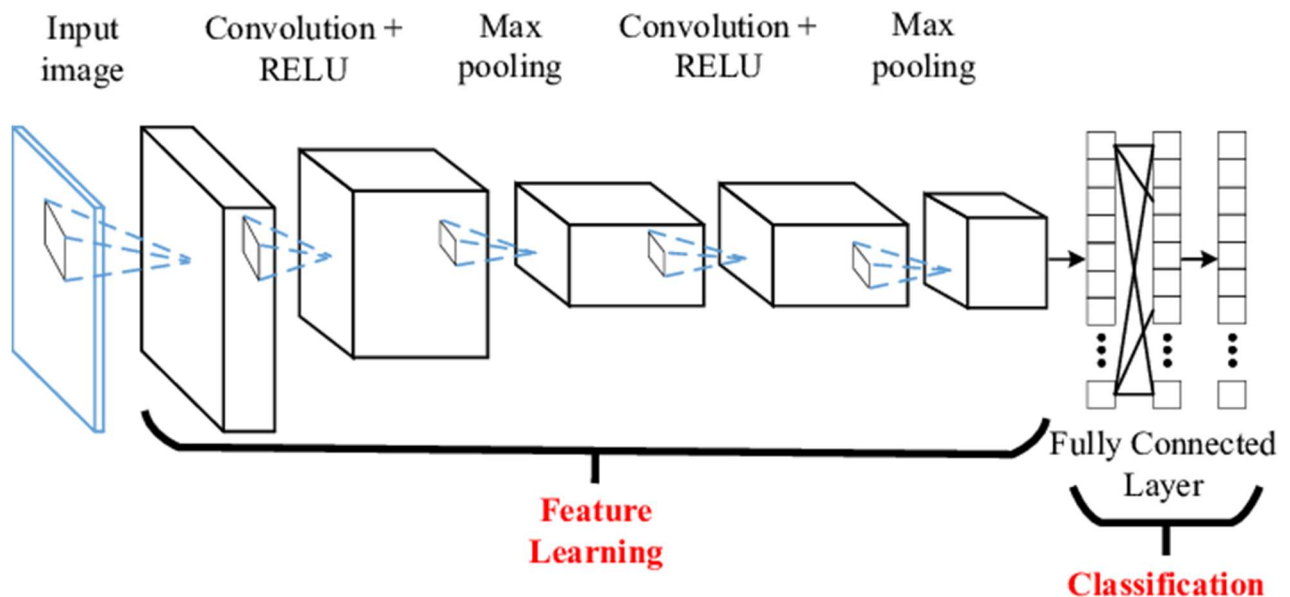


Fig-2: Architecture of Convolutional Neural Network

Convolutional neural networks (CNNs) is a classical DL algorithm, and it has been applied in many fields. It can not only select features but also classify the traffic data. CNN can learn better features automatically than traditional feature selection algorithms. The more traffic data, the more useful features the CNN can learn, the better classification the CNN performs. However, machine learning algorithms are easy to over-fit in a massive data environment. Hence, CNN is suitable for the massive network environment. Besides, compared with other DL algorithms, the greatest advantage of CNN is that it shares the same convolutional kernels, which would reduce the number of parameters and calculation amount of training once greatly, it can more quickly identify attack type of traffic data.

3.2.2 Architecture:

There are two main parts to a CNN architecture:

- In a process known as Feature Extraction, a convolution tool isolates and identifies the distinct characteristics of a picture for analysis.
- A fully connected layer that uses the output of the convolution process to forecast the image's class using the information acquired in earlier stages.

Convolution Layers:

Convolutional layers, pooling layers, and fully connected (FC) layers are the three types of layers that make up the CNN. A CNN architecture will be constructed when these layers are layered. There are two additional significant factors, the dropout layer, and the activation function, which are explained below, in addition to these three layers.

3.2.3 Layers in CNN

1. *Convolutional Layer*

- This is the initial layer that extracts the different characteristics from the input photos. The convolution mathematical operation is done between the input picture and a filter of a certain size $M \times M$ in this layer. The dot product between the filter and the sections of the input picture with regard to the size of the filter is taken by sliding the filter across the input image ($M \times M$).
- The Feature map is the result, and it contains information about the picture such as its corners and edges. This feature map is then supplied to further layers, which learn a variety of different features from the input picture.

2. *Pooling layer*

- A Pooling Layer is usually applied after a Convolutional Layer. This layer's major goal is to lower the size of the convolved feature map in order to reduce computational expenses. This is accomplished by reducing the connections between layers and operating independently on each feature map. There are numerous sorts of Pooling procedures, depending on the mechanism used.
- The biggest element is obtained from the feature map in Max Pooling. The average of the components in a predetermined sized Image segment is calculated using Average Pooling. Sum Pooling calculates the total sum of the components in the designated section. The Pooling Layer is typically used to connect the Convolutional Layer and the FC Layer.

3. *Fully Connected layer*

- The weights and biases, as well as the neurons, make up the Fully Connected (FC) layer, which is used to link the neurons between two layers. The last several layers of a CNN Architecture are generally positioned before the output layer.
- The preceding layers' input images are flattened and supplied to the FC layer in this step. After that, the flattened vector is sent via a few additional FC levels, where the mathematical functional operations are normally performed. The categorization procedure gets started at this point.

4. *Dropout*

- When all of the characteristics are linked to the FC layer, the training dataset is prone to overfitting. Overfitting happens when a model performs so well on training data that it has a detrimental influence on its performance when applied to fresh data.
- To address this issue, a dropout layer is employed, in which a few neurons are removed from the neural network during the training process, resulting in a smaller model. After passing a dropout of 0.3, 30% of the nodes in the neural network are dropped out at random.

5. Activation Functions

- Finally, the activation function is one of the most crucial elements in the CNN model. They're utilised to learn and approximate any form of network variable-to-variable association that's both continuous and complicated. In basic terms, it determines which model information should fire in the forward direction and which should not at the network's end.
- It gives the network non-linearity. The ReLU, Softmax, tanH, and Sigmoid functions are some of the most often utilised activation functions. Each of these functions has a distinct use. For a binary classification CNN model, sigmoid and softmax functions are favoured, whereas softmax is typically employed for multi-class classification.

3.2.4 Pooling:

A pooling layer is a new layer added after the convolutional layer. The addition of a pooling layer after the convolutional layer is a common pattern used for ordering layers within a convolutional neural network that may be repeated one or more times in a given model.

Two common functions used in the pooling operation are:

- Average Pooling: Calculate the average value for each patch on the feature map.
- Maximum Pooling (or Max Pooling): Calculate the maximum value for each patch of the feature map.

In this model we have done maximum pooling. This is because our data set consists of chest x-ray images which are dark. So by Max pooling we select the brighter pixels from the image. It is useful when the background of the image is dark and we are interested in only the lighter pixels of the image.

3.3 Algorithm:

Step 1: Data Preprocessing

- The pneumonia dataset contains 5863 chest x-ray images which are of two categories (Pneumonia/ Normal).
- Two labels are – (i) Pneumonia, (ii) Normal
- Before training the images we need to pre-process the dataset. The major step in data preprocessing is **data augmentation**.
- In order to avoid overfitting, we need to artificially expand our dataset. We can make your existing dataset even larger. The idea is to alter the training data with small transformations to reproduce the variations. Approaches that alter the training data in ways that change the array representation while keeping the label the same are known as data augmentation techniques. Some popular augmentations people use are grayscales, horizontal flips, vertical flips, random crops, color jitters, translations, rotations, and much more. By applying just a couple of these transformations to our

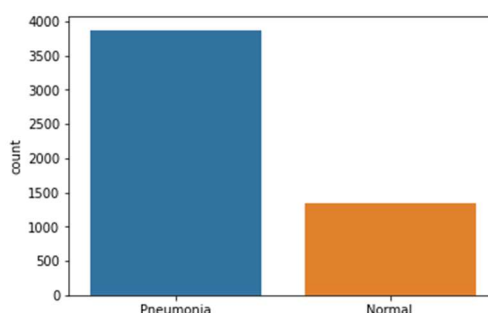
training data, we can easily double or triple the number of training examples and create a very robust model.

Step 2: Loading the Dataset

- The Chest X-ray data we are using from [Cell](#) divides the data into train, val, and test files. There are only 16 files in the validation folder, and we would prefer to have a less extreme division between the training and the validation set.
- We will append the validation files and create a new split that resembles the standard 80:20 division instead

Step 3: Data Visualization

- Since the project is working on a classification problem, if our data labels are not likewise distributed- the results will be highly biased too. This is not recommended.



- Clearly, the data seems to be imbalanced in this case. Now in order to balance the distribution of the training examples, we would be using data augmentation procedures.

Step 4: Image Normalization

- Now in the images that we have, there might be instances where the range of illumination is too huge for processing. Like any other model, if a normalization factor is introduced, the model's ability to interpret the data becomes much better.
- Likewise, a Convolution Neural Network will be much more efficient when the data is normalized.

Step 5: Data Augmentation

- Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model.

Step 6: Train the Model

- For our metrics, we want to include precision and recall as they will provide us with a more informed picture of how good our model is. Accuracy tells us what fractions are the labels are correct. Since our data is not balanced, accuracy might give a skewed sense of a good model (i.e. a model that always predicts PNEUMONIA will be 74% accurate but is not a good model).
- Precision is the number of true positives (TP) over the sum of TP and false positives (FP). It shows what fraction of labeled positives are actually correct.

3.4 Coding and Implementation:

1. IMPORTING LIBRARIES

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
import os
import random
import tensorflow as tf
from tensorflow import keras
import keras.backend as K
from matplotlib import cm
import matplotlib.image as mpimg
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score, confusion_matrix
```

2. EDA

```
inputdir = 'L:\My Drive\FALL SEM 2021 - 22\F -
NN\PROJECT\FINAL FROM KAGGLE\chest_xray\chest_xray'
path_train = inputdir + '/train'
path_test = inputdir + '/test'
path_val = inputdir + '/val'
train_normal = os.path.join(path_train + '/NORMAL')
train_pneumonia = os.path.join(path_train + '/PNEUMONIA')
test_normal = os.path.join(path_test + '/NORMAL')
```

```

test_pneumonia = os.path.join(path_test + '/PNEUMONIA')
val_normal = os.path.join(path_val + '/NORMAL')
val_pneumonia = os.path.join(path_val + '/PNEUMONIA')
print("Total number of normal images in training set:
",len(os.listdir(train_normal)))
print("Total number of pneumonic images in training set:
",len(os.listdir(train_pneumonia)))
print("Total number of normal images in test set:
",len(os.listdir(test_normal)))
print("Total number of pneumonic images in test set:
",len(os.listdir(test_pneumonia)))
print("Total number of normal images in val set: ",
len(os.listdir(val_normal)))
print("Total number of pneumonic images in val set: ",
len(os.listdir(val_pneumonia)))
# Parameters for our graph; we'll output images in a 4x4
configuration
nrows = 4
ncols = 4
# Set up matplotlib fig, and size it to fit 4x4 pics
fig = plt.gcf()
fig.set_size_inches(ncols * 4, nrows * 4)
next_normal_pix = [os.path.join(train_normal, fname) for fname
in os.listdir(train_normal)[0:8]]
next_pneumonia_pix = [os.path.join(train_pneumonia, fname) for
fname in os.listdir(train_pneumonia)[0:8]]
for i, img_path in
enumerate(next_normal_pix+next_pneumonia_pix):
    # Set up subplot; subplot indices start at 1
    sp = plt.subplot(nrows, ncols, i + 1)
    sp.axis('Off') # Don't show axes (or gridlines)
    img = mpimg.imread(img_path)
    if i<=7 :
        sp.title.set_text('NORMAL')
    else :
        sp.title.set_text('PNEUMONIA')
    plt.imshow(img)

```

```
plt.show()
```

3. DATA PREPROCESSING

```
def data_processing(img_size, batch_size):

    train_datagen = ImageDataGenerator(rescale = 1./255,
                                       zoom_range = 0.3,
                                       vertical_flip = True,
                                       width_shift_range =
0.1,
                                       height_shift_range =
0.1)

    val_datagen = ImageDataGenerator(rescale=1./255)

    train_set = train_datagen.flow_from_directory(
        path_train,
        target_size=(img_size, img_size),
        batch_size=batch_size,
        class_mode='binary',
        shuffle=True)

    val_set = val_datagen.flow_from_directory(
        path_val,
        target_size=(img_size, img_size),
        batch_size=batch_size,
        class_mode='binary',
        shuffle=True)

    test_data = []
    test_labels = []
    for cond in ['/NORMAL/', '/PNEUMONIA/']:
        for img in (os.listdir(path_test + cond)):
            img = plt.imread(path_test + cond + img)
            img = cv2.resize(img, (img_size, img_size))
            img = np.dstack([img, img, img])
```

```

        img = img.astype('float32') / 255
        if cond == '/NORMAL/':
            label = 0
        elif cond == '/PNEUMONIA/':
            label = 1
        test_data.append(img)
        test_labels.append(label)
    test_data = np.array(test_data)
    test_labels = np.array(test_labels)

    return train_set, val_set, test_data, test_labels

img_size = 150
batch_size = 32
train_set, val_set, test_data, test_labels =
data_processing(img_size, batch_size)

```

4. BUILDING THE MODEL

```

from keras.models import Sequential
from tensorflow.keras.layers import
Conv2D,MaxPool2D,SeparableConv2D,BatchNormalization,Dropout,Fl
atten,Dense

model = Sequential()

model.add(Conv2D(filters = 16, kernel_size = (3, 3),
input_shape = train_set.image_shape, activation = 'relu',
padding = 'same'))
model.add(Conv2D(filters = 16, kernel_size = (3, 3),
activation = 'relu', padding = 'same'))
model.add(MaxPool2D(pool_size = (2, 2)))

model.add(SeparableConv2D(filters = 32, kernel_size = (3, 3),
activation = 'relu', padding = 'same'))

```

```
model.add(SeparableConv2D(filters = 32, kernel_size = (3, 3),
activation = 'relu', padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size = (2, 2)))

model.add(SeparableConv2D(filters = 64, kernel_size = (3, 3),
activation = 'relu', padding = 'same'))
model.add(SeparableConv2D(filters = 64, kernel_size = (3, 3),
activation = 'relu', padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size = (2, 2)))

model.add(SeparableConv2D(filters = 128, kernel_size = (3, 3),
activation = 'relu', padding = 'same'))
model.add(SeparableConv2D(filters = 128, kernel_size = (3, 3),
activation = 'relu', padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size = (2, 2)))
model.add(Dropout(rate = 0.2))

model.add(SeparableConv2D(filters = 256, kernel_size = (3, 3),
activation = 'relu', padding = 'same'))
model.add(SeparableConv2D(filters = 256, kernel_size = (3, 3),
activation = 'relu', padding = 'same'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size = (2, 2)))
model.add(Dropout(rate = 0.2))

model.add(Flatten())
model.add(Dense(units = 512, activation = 'relu'))
model.add(Dropout(rate = 0.7))
model.add(Dense(units = 128, activation = 'relu'))
model.add(Dropout(rate = 0.5))
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dropout(rate = 0.3))
model.add(Dense(units = 1, activation = 'sigmoid'))
```



```
model.summary()
```

5. COMPILING THE MODEL AND CALLBACKS

```
model.compile(optimizer = tf.keras.optimizers.Adam(lr = 0.001,
decay = 1e-5), loss = 'binary_crossentropy', metrics =
['accuracy'])
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
checkpoint = ModelCheckpoint(filepath =
'trained_pneumonia_model.h5', save_best_only = True,
save_weights_only = False)
lr_reduce = ReduceLROnPlateau(monitor = 'val_loss', factor =
0.3, patience = 2, verbose = 2, mode = 'max')
```

6. TRAINING THE MODEL

```
count_normal, count_pneumonia = len(os.listdir(train_normal)),
len(os.listdir(train_pneumonia))

weight_0 = (1 / count_normal) * (count_normal +
count_pneumonia) / 2.0
weight_1 = (1 / count_pneumonia) * (count_pneumonia +
count_normal) / 2.0

class_weights = {0 : weight_0, 1 : weight_1}
model_training = model.fit_generator(train_set,
steps_per_epoch =
train_set.samples // batch_size,
epochs = 100,
validation_data = val_set,
callbacks = [checkpoint,
lr_reduce],
class_weight = class_weights)
```

7. MODEL EVALUATION

```

from matplotlib import pyplot as plt
fig, ax = plt.subplots(1, 2, figsize = (8, 4))
ax = ax.ravel()

for i, metric in enumerate(['accuracy', 'loss']):
    ax[i].plot(model_training.history[metric])
    ax[i].plot(model_training.history['val_' + metric])
    ax[i].set_title('Model {}'.format(metric))
    ax[i].set_xlabel('Number of epochs')
    ax[i].set_ylabel(metric)
    ax[i].legend(['train', 'val'])
predictions = model.predict(test_data)
accuracy = accuracy_score(test_labels,
np.round(predictions))*100
cm = confusion_matrix(test_labels, np.round(predictions))

cmd = ConfusionMatrixDisplay(cm, display_labels=['Normal',
'Pneumonia'])
cmd.plot(cmap = plt.cm.Blues)
plt.title('Confusion Matrix')

tn, fp, fn, tp = cm.ravel()
print('Test Metrics:')
precision = tp / (tp + fp) * 100
recall = tp / (tp + fn) * 100
print('Accuracy: {}'.format(accuracy))
print('Precision: {}'.format(precision))
print('Recall: {}'.format(recall))
print('F1-score:
{}'.format(2*precision*recall/(precision+recall)))

from sklearn.metrics import classification_report

predictions = np.round(model.predict(test_data))
print(classification_report(test_labels, predictions,
target_names = ['Normal (Class 0)', 'Pneumonia (Class 1)']))

```

CHAPTER - IV

RESULTS & DISCUSSION

4.1 Accuracy: It is the ratio of the number of correct predictions to the total number of input samples. A higher accuracy implies a better performance in a model. It is calculated by:

$$\frac{TP+TN}{TP+FP+TN+FN}$$

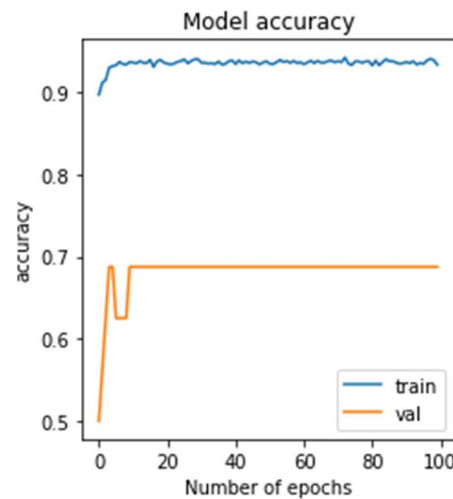


Fig-3: Accuracy of the Model vs number of Epochs

4.2 Loss: Sparse Categorical Cross Entropy calculates the compares each of the predicted probabilities to actual class output. A lower loss implies a better model.

$$J(\mathbf{w}) = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

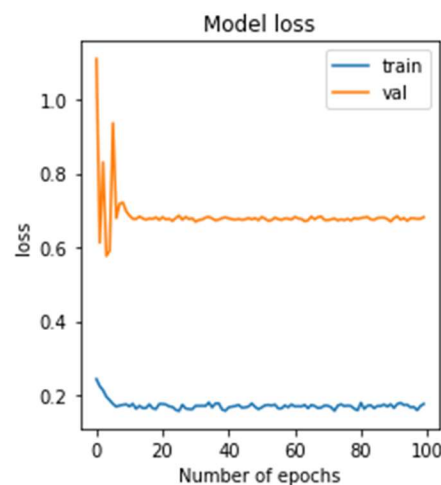
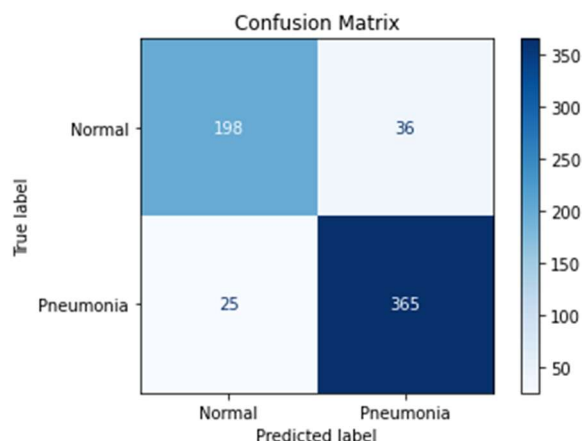


Fig-4: Loss of the Model vs number of Epochs

4.3 Confusion Matrix:

Table-1: Confusion Matrix

	Normal	Pneumonia
Normal	198	36
Pneumonia	25	365



4.4 Classification Report:

Table-2: Classification Report

	Precision	Recall	F1 - score	Support
Normal	0.89	0.85	0.87	234
Pneumonia	0.91	0.94	0.92	390

Training Accuracy: 93.4% Validation Accuracy: 68.75%

Detection Rate and False Alarm Rates:

- **Detection Rate (DR)** is defined as the proportion of the whole sample where the events were detected correctly. The higher the DR, the better the model performance. It can be calculated as:

$$\frac{TP}{TP+FN}$$

- **False Alarm Rate (FAR)** is defined as the expectancy of the false-positive ratio. For any Machine/Deep Learning model, a lower false alarm rate is preferred. It is calculated as:

$$\frac{FP}{FP+TN}$$

CHAPTER - V

CONCLUSION

5.1 Conclusion

The goal of this challenge was to assess whether a person has pneumonia or not. A model was created to find the same and we were able to achieve an accuracy of 93 percent. When a patient has pneumonia and is misdiagnosed as healthy, the model is incorrect, however, in the model we constructed, the chance of a patient being misdiagnosed with pneumonia is less than 10%.

We can generalize the data and improve the accuracy of diagnosis by gathering additional data from various patients in different hospitals in different regions of the world.

Furthermore, this model may be used as a patient screening test since, as seen in the graph, the model virtually always detects a person with pneumonia.

5.2 Future Scope

- Pneumonia Detection can also help in the detection of COVID since both of them actively work on the same bases, akin the lungs get affected. So this model can be extended in order to help ease COVID detection as well.

REFERENCES

- [1] Račić, Luka & Popovic, Tomo & Cakic, Stevan & Šandi, Stevan. (2021). Pneumonia Detection Using Deep Learning Based on Convolutional Neural Network. 10.1109/IT51528.2021.9390137.
- [2] M. A. Islam and M. S. Arefin, "A framework for detecting arsenic disease," 2016 3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT), 2016, pp. 1-5, doi: 10.1109/CEEICT.2016.7873149.
- [3] Chithambaram T, Logesh Kannan N, Gowsalya M et al. Heart Disease Detection Using Machine Learning, 27 October 2020, PREPRINT (Version 1) available at Research Square [<https://doi.org/10.21203/rs.3.rs-97004/v1>]
- [4] Gabruseva, T., Poplavskiy, D., & Kalinin, A. (2020). Deep Learning for Automatic Pneumonia Detection. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).
- [5] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, & Andrew Y. Ng. (2017). CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning.
- [6] Turbé, V., Herbst, C., Mngomezulu, T. et al. Deep learning of HIV field-based rapid tests. *Nat Med* 27, 1165–1170 (2021). <https://doi.org/10.1038/s41591-021-01384-9>
- [7] Gonçalves WGE, Dos Santos MHP, Lobato FMF, Ribeiro-Dos-Santos , de Araújo GS. Deep learning in gastric tissue diseases: a systematic review. *BMJ Open Gastroenterol.* 2020 Mar 26;7(1):e000371. doi: 10.1136/bmjgast-2019-000371. PMID: 32337060; PMCID: PMC7170401.
- [8] El Asnaoui K., Chawki Y., Idri A. (2021) Automated Methods for Detection and Classification Pneumonia Based on X-Ray Images Using Deep Learning. In: Maleh Y., Baddi Y., Alazab M., Tawalbeh L., Romdhani I. (eds) *Artificial Intelligence and Blockchain for Future Cybersecurity Applications. Studies in Big Data*, vol 90. Springer, Cham. https://doi.org/10.1007/978-3-030-74575-2_14
- [9] Sharma, Abhishek; Raju, Daniel; Ranjan, Sutapa (2017). [IEEE 2017 Nirma University International Conference on Engineering (NUiCONE) - Ahmedabad, India (2017.11.23-2017.11.25)] 2017 Nirma University International Conference on Engineering (NUiCONE) - Detection of pneumonia clouds in chest X-ray using image processing approach. , (), 1–4. doi:10.1109/NUiCONE.2017.8325607
- [10] Li, Feng; Engelmann, Roger; Pesce, Lorenzo; Armato, Samuel G.; MacMahon, Heber (2012). Improved detection of focal pneumonia by chest radiography with bone suppression imaging. *European Radiology*, 22(12), 2729–2735. doi:10.1007/s00330-012-2550-y
- [11] A. Tilve, S. Nayak, S. Vernekar, D. Turi, P. R. Shetgaonkar and S. Aswale, "Pneumonia Detection Using Deep Learning Approaches," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020, pp. 1-8, doi: 10.1109/ic-ETITE47903.2020.

BIODATA



Name : Andrew John

Mobile Number : 90037 35112

E-mail : andrewjohn.j2018@vitstudent.ac.in

Address : E405, Radiance Mandarin 200 Feet Thoraipakkam - Pallavaram Radial Road Thoraipakkam, Chennai-97.



Name : N. T. Srihari

Mobile Number : 98405 22487

E-mail : nt.srihari2018@vitstudent.ac.in

Address : F115, Unity Enclave, Mambakkam, Chennai - 600127.



Name : R. Vignesh

Mobile Number : 88709 82952

E-mail : r.vignesh2018@vitstudent.ac.in

Address : 3D7 Jains Inseli Park, OMR Road, Padur, Chennai 603103.



Name : Vighnesh M.

Mobile Number : 81481 66514

E-mail : vighnesh.m2018@vitstudent.ac.in

Address : 4A/25, Andavar Street, Choolaimedu, Chennai-94.