Multiple Chest Disease Detection from X-ray images Applying Deep Neural Network

A thesis

Submitted in partial fulfillment of the requirements for the Degree of

Bachelor of Science in Computer Science and Engineering

Submitted by

Junaed Mohammed Uddin	**160104109**
Syed Muztaba Ali	**160104115**
Sabbir Hossain	**160104124**
Jeba Maliha	**160104143**

Supervised by

Mr. H M Zabir Haque

Assistant Professor

Department of Computer Science and Engineering

Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

Ahsanullah University of Science and Technology

Dhaka, Bangladesh

December, 2020

CANDIDATES' DECLARATION

We, hereby, declare that the thesis presented in this report is the outcome of the investigation performed by us under the supervision of Mr. H M Zabir Haque, Assistant Professor, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. The work was spread over two final year courses, CSE4100: Project and Thesis-I and CSE4250: Project and Thesis-II, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Junaed Mohammed Uddin
160104109

Syed Muztaba Ali
160104115

Sabbir Hossain
160104124

Jeba Maliha
160104143

CERTIFICATION

This thesis titled, **"Multiple Chest Disease Detection from X-ray images Applying Deep Neural Network"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in December, 2020.

Group Members:

Junaed Mohammed Uddin	**160104109**
Syed Muztaba Ali	**160104115**
Sabbir Hossain	**160104124**
Jeba Maliha	**160104143**

Mr. H M Zabir Haque
Assistant Professor & Supervisor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Prof. Dr. Kazi A Kalpoma
Professor & Head
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

ACKNOWLEDGEMENT

First and foremost, We are grateful to the Almighty Allah for the good health and well being that were necessary to complete this thesis work. Then we have to thank our thesis supervisor, Mr. H M Zabir Haque - without whose encouragement, this thesis would have never been accomplished. His constant support, understanding and constructive critiques over the final year have enriched our thesis work to a great extent.

We place on record, our sincere thanks to Prof. Dr. Kazi A Kalpoma, honourable Head of the department, for her continuous encouragement.

We take this opportunity to express gratitude to all the respected faculty members of our department for their help and support. Also, we thank our parents for the unceasing encouragement, support and attention. Finally, we are grateful to everyone who helped us in every possible ways to make our thesis fruitful.

Dhaka
December, 2020

Junaed Mohammed Uddin

Syed Muztaba Ali

Sabbir Hossain

Jeba Maliha

ABSTRACT

Recently, researchers, experts, and companies around the world are rolling out deep learning and image processing-based systems that can fastly process hundreds of X-Ray images to accelerate the diagnosis of chest diseases such as COVID-19, Pneumonia, Effusion, etc. However, it has been quite a challenging task to clinically diagnose chest diseases from the scans. Consequently, there is a dire need for automated diagnosis systems to assist clinicians in making better decisions. This study aims to simplify the chest disease detection process by using custom CNN and transfer learning with (CLAHE, Power Law Transform) and without preprocessing for experts as well as for novices. Moreover, we present a comparison between Deep Convolutional Neural Network (DCNN) for multi-class classification of chest diseases and fined tuned versions of (VGG16, DenseNet121, Inception-ResNet-V2, Inception V3, Resnet50, and Xception). The proposed work has been tested using the chest X-Ray 14 CXI dataset, which contains 34439 images (5679 pneumonia, 12930 Effusion, and 15830 no findings). Furthermore, we have trained CNN and transfer learning architectures to determine which architecture is better based on the ablation experiment. Here, we have done eleven CNN centered ablation experiments in training evaluation based on three approaches of datasets and found that setup-9 gave better results (Accuracy more than 82%) without preprocessing than other setups. But in pre-trained architectures, it can be seen that we have got satisfactory testing accuracy without preprocessing on the dataset using the VGG16 pre-trained model (Accuracy more than 94%) which is better than CNN (Setup-9) and other pre-trained models. (more than 88% accuracy).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The world is developing day by day, but the diseases on health are rapidly increasing for the environment, the adverse climate changes, the lifestyle of humans, and so on. Furthermore, Lung diseases are one of the most traditional medical conditions in the world. In many major chest diseases, Pneumonia has the highest mortality rate among young children and old aged people around the world. It is called the silent killer disease that shows social awareness as compared to other diseases, and therefore the number of infections is increasing multifold per annum [1]. Additionally, COPD **(Chronic obstructive pulmonary disease)** is a common lung disease that affects over 200 million people in the world, 65 million of whom have moderate or severe airway disease, and each year, 3 million people die from this disease. That's why it becomes the third leading cause of death in the world where 334 million people have asthma, and it is the most common chronic disease of childhood which affects 14% of all children globally [2]. Some important reasons for these diseases are overweight, smoking, diabetes and high blood pressure, Cholesterol levels are high in the blood and also, if they have a family history, they're more likely to possess chest diseases. Moreover, age is a factor for chest disease. For example, men over the age of 45 and women over the age of 55 are more likely to have chest diseases. Since the beginning of the 21st century, several coronaviruses have caused deadly Pneumonia in humans. Here, we will provide a concise summary of the epidemiology and history of the type of Coronavirus in particular: Antonine Plague, Swine Flu, Ebola, Covid-19 [3].

Table 1.1: The major pandemics that have occurred over different period

Name	Period	Type/Pre-human host	Death toll
Antonine Plague	165-180	Believe to either smallpox or measles	5M
Cholera Pandemics 1-6	1817-1923	V. cholerae bacteria	1M+
HIV/AIDS	1981-Present	Virus /Chimpanzees	25-35M
Swine Flu	2009-2010	H1N1 virus / Pigs	200000
Ebola	2014-2016	Bats, Civets	11000
Covid-19	2019-Present	Coronavirus – Unknown (possibly Bats or pangolins)	348,319 (as of May 26, 2020)

However, there are several ways to diagnose lung diseases. Needless to mention that the Chest X-ray is one of the most accessible radiological examinations for screening and diagnosis of many lung diseases. Medical experts have used this method for a long time to pursue fractures or unnaturalness in body organs and diagnose important chest diseases like pleurisy, effusion, Pneumonia, bronchitis, nodule, lung cancer, cardiomegaly, pneumothorax by analyzing X-ray images of the chest which have a very high death rate. In that case, X-rays are handy diagnostic tools in revealing pathological alterations. But due to lack of time and adequate health workers, they also find chest X-ray classification to be a tedious task. For the sake of that, many analysts have proposed some automated methods to detect chest disease precisely without human efforts. In this case, the most significant issue is that this paper will emphasize in deep learning and transfer learning to diagnose various chest diseases from X-ray images.

1.1 Objective

Chest X-rays are widely used to diagnose anomalies in the heart and lung region. Automatically detecting these anomalies with high accuracy has greatly enhanced real-world diagnostic processes. Due to several medical conditions in the lungs, such as volume loss, bleeding, lung cancer, bone fractures, tumours, surgery, dental issues, and the diagnosis of pneumonia or any other diseases using chest X-rays becomes very useful but time-consuming. However, indeed this method does not work well when the epidemic turns into a terrible situation. Immediately, there is a dire need for automated diagnosis systems to assist clinicians in making better decisions. For several decades the active area of research has been detection and diagnosis of diseases. In its continuation, various researchers have identified and diagnosed chest diseases using variant methods. Among them, Convolutional Neural Network (CNN) is one of the most frequently used approaches in not only for classifying the natural images but also for classifying different kinds of medical images [4]. The general approach to the Machine Learning approach and Machine Learning approach to the Deep Learning approach

keeps high acceleration because of its high performance. Recently, many researchers have proposed different artificial intelligence (AI)-based solutions for different medical problems and obtained successful results in broad medical issues like breast cancer detection, brain tumour detection, and segmentation, disease classification in X-ray images, etc. Seeing such successful results of this method, we became interested in working on it. Following this phenomenon, we will create an automated system that can quickly detect two types of diseases from X-ray images using deep learning. Furthermore, if there is no disease, then the system will say no disease exists. Those diseases are Effusion and Pneumonia. In a nutshell, we will pre-process our dataset and train by using CNN (Convolutional Neural Network) and Transfer Learning separately. Then we will compare the results between CNN and Transfer Learning to achieve the best results. Our intension is to build an accurate automated system that will detect multiple chest diseases according to such pre-processing techniques and deep learning.

1.2 Chest X-ray

X-rays stand for X-radiation is a picture taken of the inside of something using high energy electromagnetic radiation with short wavelengths starting from 0.1 to 10 nanometers that can pass-through items [5]. The radiologist uses X-rays to help diagnose problems such as broken bones, tumours, dental decay. It is often used in detecting abnormalities within the body. Moreover, the chest x-ray is the most commonly used diagnostic x-ray examination. It helps to create images of nearby structures of the chest and identify abnormalities in the heart, lungs, trachea, blood vessels, spinal cord, and chest bones. It also used to help diagnose and monitor treatment for a variety of lung conditions like pneumonia, emphysema, and cancer.

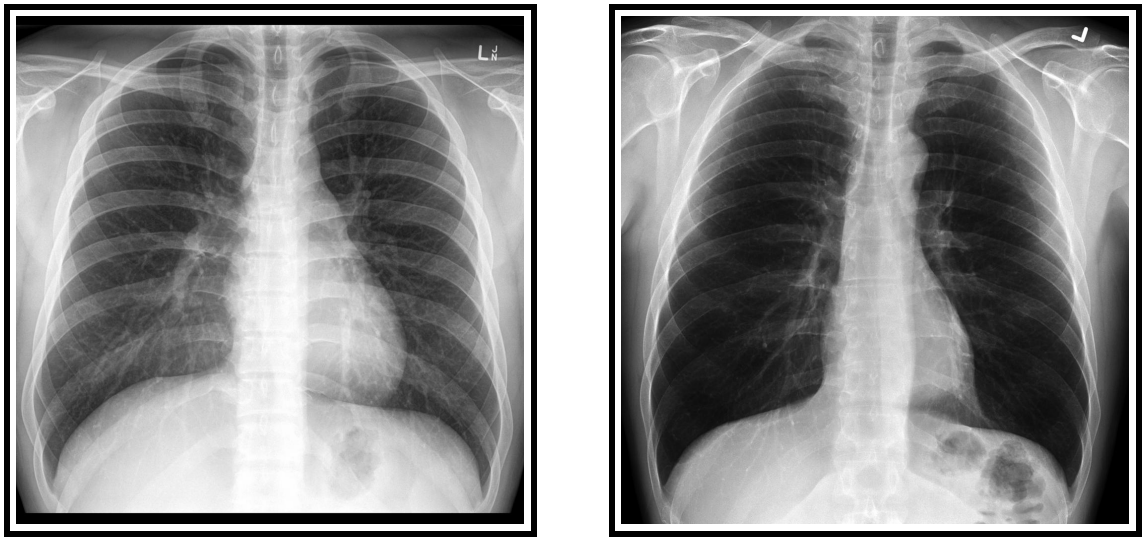
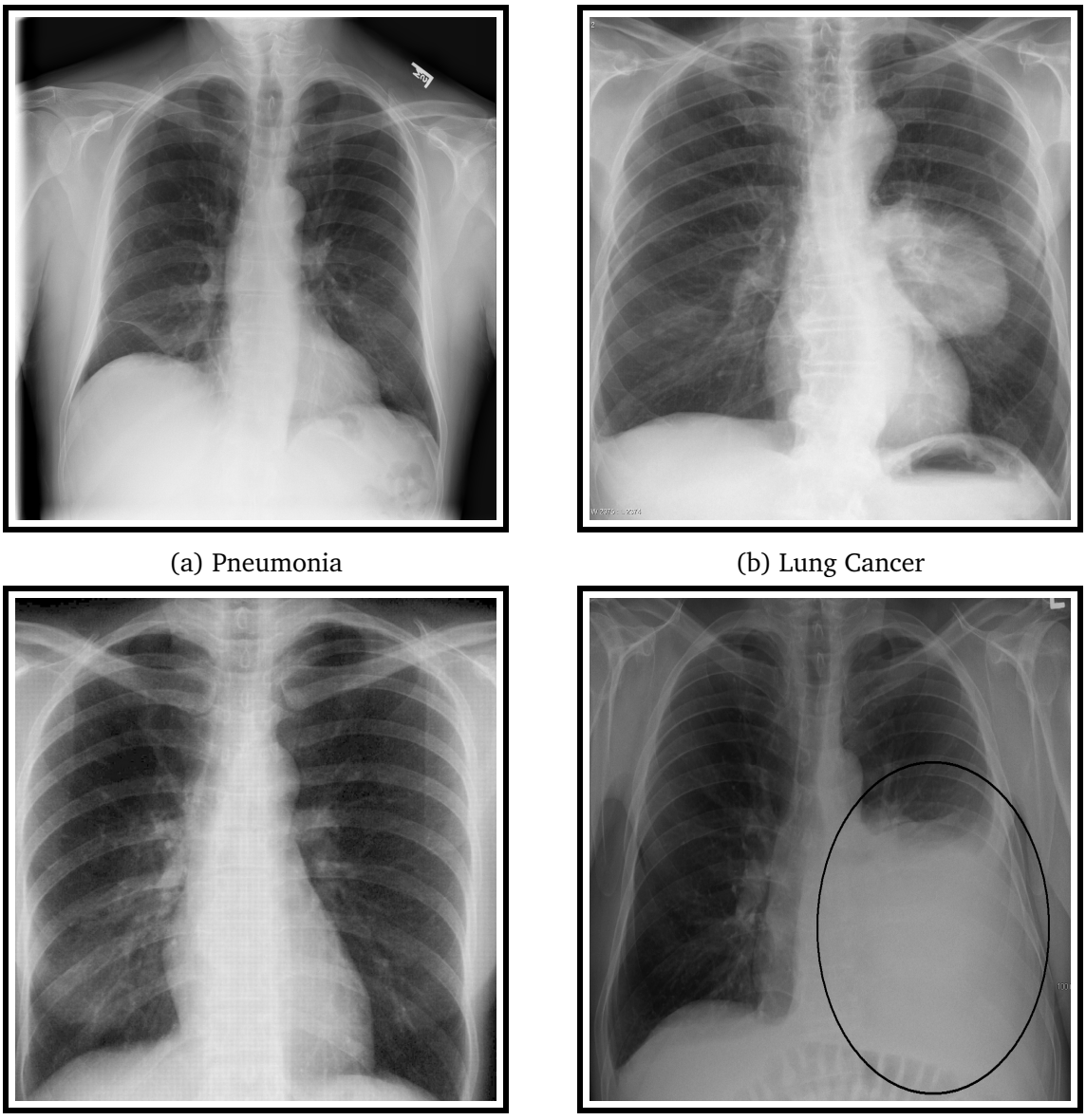


Figure 1.1: Chest X-ray

1.3 Lung Diseases

Lung diseases are one of the most dangerous diseases in the world. Almost ten million people have lung diseases within the U. S alone. Smoking, infections, and genes cause most lung diseases [6]. Furthermore, lungs are a part of a posh system that carries oxygen and transmit carbon dioxide, which expands and relaxes several thousand times a day. So, only lung disease can happen when there are problems in any part of this system. There are many varieties of lung diseases like Asthma, Pneumonia, Bronchitis, Effusion, lung cancer, and so on.



(a) Pneumonia

(b) Lung Cancer

(c) Bronchitis

(d) Effusion

Figure 1.2: Different types of lung diseases

Chapter 2

Literature Review

2.1 Overview

Deep learning models can acquire human-like thinking and intelligence because of their complex structure which is similar to that of the human nervous system. It is one of the most frequently used techniques not only for classifying the natural images but also for classifying different kinds of medical images. For this reason, a significant amount of work done by scientists, researchers by using deep learning along with pre-processing and augmentation techniques to extend the size of the dataset. Satisfactory results have been obtained for various datasets by using this approach in the medical sector. To detect chest diseases precisely, some papers have been proposed based on some pre-trained models (Transfer Learning) and custom model (CNN). Besides, some articles have added additional image pre-processing techniques. However, there are numerous various techniques and methods for the pre-processing and detection stage that are proposed in the literature that have been recognized internationally. The proposed systems are immensely renowned for detecting chest diseases and played an essential role in the medical field for an extended period. This section presents a short overview of some of the crucial contributions to the existing literature.

2.2 Related Works

In this paper, the pneumonia disease was detected from the chest x-rays. The researchers suggested a novel deep learning framework for detecting pneumonia by using transfer learning. The study indicated that the method applied in this paper could be used to minimize human error. Thus, effectively improve the quality of the treatment. However, the authors of this article claimed that their final model achieved an accuracy of 96.4%.

To achieve the result, the researchers applied deep learning in this experiment. Images were resized to 224 x 224. Then the data augmentation is applied to increase the number of images to reduce overfitting. Then transfer learning was used using AlexNet, DenseNet121, InceptionV3, resNet18, and GoogLeNet neural architecture for feature extraction and ensemble classification [7].

In this paper, pneumonia was detected from the chest x-ray using a Deep Convolutional Neural Network (DCNN). The goal of this paper was to build a system that can rapidly detect pneumonia which can be occurred by SARS, Covid-19, etc. to reduce human error. Also, the team wanted to know if there was any Deep Learning technique that can outperform other DCNN techniques. This paper claimed that they got an accuracy of 96%.

To achieve the result, the researchers used a baseline CNN and transfer learning in this experiment. Different image sizes were used for a different architecture. Intensity Normalization, CLAHE, and data augmentation were applied in the preprocessing step. A baseline CNN with 3 convolutional layers and the fine-tuned version of VGG16, VGG19, DenseNet201, ResNet50, Inception-V3, Inception-ResNet-V2, and MobileNet-V2 were used for classification [3].

In this paper, the researchers tried to establish lung disease as the second most common type of disease. Many types of lung diseases such as lung cancer, asthma, chronic pulmonary disease, etc. cause the death of many every year. So, they proposed a system that could detect various types of lung diseases using hybrid deep learning. This paper focused on the performance of the vanilla neural network, VGG, CNN, the fusion of VGG, STN, CNN, and Capsule network.

In this experiment, the researchers used a dataset which was consisted of 112,120 images from fifteen classes. They proposed a basic CNN and hybrid of CNN for feature extraction and classification. Images were rescaled, converted to RGB from grey in the preprocessing steps for faster training. They used CNN+VGG+STN, vanilla CNN (for both RGB and grey images), capsule network (both for Hinton's architecture and custom architecture) for the evaluation. The CNN+VGG+STN was the best performer among them [8].

In this paper, the authors had proposed a system that could detect various types of chest related diseases using transfer learning. Chest related diseases considered as the leading cause of death in the world. This paper tried to identify the diseases without the intervention of a human. This method would help the radiologist to get the work done faster and also to double-check the result. This model would identify fourteen different types of chest disease from chest x-ray images.

To achieve the desired result, the researchers used wavelet transform and transfer learning. They reduced the size of the images for faster training. Data augmentation, wavelet transform, was performed in preprocessing steps. Then they moved the knowledge of an

already-trained network (ResNet50) on the ImageNet dataset to their domain to detect the chest diseases. They ran the experiment two times in the same model. At first, they trained the model just using augmented data, then they trained the model again using wavelet transform and augmented data [9].

In this paper, pediatric pneumonia was detected from the chest x-ray as pediatric pneumonia was poorly studied. CNN could be helpful in this situation, but CNN's could be considered as black boxes and poorly understood. This lack of transparency is regarded as a severe condition of medical screening. Visualization tools were proposed in this paper to explain the model. This paper highlighted the advantages of visualizing CNN's behaviours.

In this experiment, the researchers used a dataset which was consisted of anteroposterior chest radiograph of one to five years of age. The dataset had three classes, i.e., normal, bacterial pneumonia, and viral pneumonia. The researchers used a custom CNN and pre-trained VGG16 network for feature extraction and classification. Then they used Bayesian optimization to search for optimal parameters of the model. They used gradient weighted class activation maps (grad-CAM) and Local Interpretable Model-Agnostic Explanation (LIME) visualization tools to explain predictions and to evaluate the usefulness of the model in decision making [10].

In this paper, the pneumonia disease was detected from the chest x-rays. A deep learning framework was suggested for detecting pneumonia by using transfer learning (ResNet50). The study indicated that the method applied in this paper could be used to minimize human error. Thus, effectively improve the quality of the treatment as detecting pneumonia considered complicated from chest x-ray due to several medical conditions in lunges. The researches of this paper claimed that their final model achieved an accuracy of 96.67%.

In this experiment, the researchers used two publicly available datasets, i.e., RSNA dataset and Chest X-Ray Image (CXI) Dataset. In the preprocessing step, they considered three types of preprocessing for the data, i.e., addition, removal, and transformation of attributes in the data. Then a pre-trained ResNet50 model on ImageNet dataset was used as the classifier. Most of the layers were untouched. A new prediction layer replaced only the final layer [1].

In this paper, the authors proposed and evaluated an end-to-end deep convolutional neural network which can classify chest diseases. They used a deep convolutional neural network instead of traditional techniques because the imaging devices could capture low-level visual information. But human is good at perceiving high-level information. According to the authors, Deep Learning techniques could solve this problem. The authors of this paper claimed that they got an accuracy of 89.77%. The authors noticed that the proposed method best suited for classifying different thorax diseases.

In this experiment, the authors used a large dataset consisted of 112,120 frontal chest x-ray images. The dataset had fifteen classes. One out of these fifteen classes represented

no finding. Others had disease labelled in them. The researcher split their dataset into three categories, i.e., training, validation, and testing set. They used 70% images for training, 10% for validation, and 20% for testing. They also resized the size of the image from 1024x1024 to 224x224 for faster training. Then a CNN architecture was developed for feature extraction and classification. The proposed CNN architecture had three convolutional layers, three activation layers, three pooling layers, and three fully-connected layers [4].

In this paper, the researchers detected many types of lung diseases from the chest X-rays. The main goal of the researchers was to initiate future efforts by promoting public datasets. According to this paper, a tremendous number of X-ray image was stored in many modern hospitals' Picture Archiving and Communication System (PACS). But they were loosely labelled. So, they presented a new chest X-ray database called ChesX-ray8. The researchers showed that the diseases presented in the database could be successfully detected and even spatially located using weakly supervised multi-label classification and disease localization framework.

In this experiment, at first, researchers constructed the "ChestX-ray8" database with the help of their own institutes' PACS system. They used a variety of Natural Language Processing (NLP) to extract images from the system. The database consisted of eight diseases. They used DNorm and MetaMap for pathology detection. Then they set the size of the image to 1024x1024. Then they used a Deep Convolutional Neural Network (DCNN) for disease detection and object localization [11].

This paper was constructed by the Convolutional Neural Network model trained, which extracts a set of chest X-ray images and might detect the presence of pneumonia. Then they have demonstrated the way to classify positive and negative pneumonia data from a collection of X-ray images and build their model from scratch, which separates it from other methods that depend on the transfer learning approach.

They employed several data augmentation methods to artificially increase the dimensions and quality of the dataset. The proposed CNN model has two significant parts, feature extractors, and classifier. By transforming chest X-ray images into sizes smaller than the original, the algorithm begins. Furthermore, the identification and classification of images have done by the CNN framework, which extracts features from the X-ray images and classifies them. To robust the performance of the trained model on different chest X-ray image sizes, they have trained their model several times and got similar results of the training and validation dataset [12].

This paper proposed an automatic computerized system for detecting pneumonia using a Compressed Sensing (CS) based Deep Learning (DL) framework. For automatic detection of pneumonia on X-ray images reduces the desired observations for detecting pneumonia with the specified accuracy compared to the traditional method.

A multi-layer convolutional neural network (CNN) is employed to extract features from CS measurements for classifying the X-ray images of a pneumonia patient. A sub-channel is employed to produce the reconstructed X-ray image, which helps to verify whether pneumonia is present or not. The dataset has been used from Kaggle which contains 5863 X-ray images (JPEG) of two categories (Pneumonia/Normal) whether 70% of the dataset for the training, 25% for the validation and 5% for the test purpose. The performance of the trained model is additionally assessed using the test dataset and compared with the prevailing methods like F-cMeans, DL, and ChexNet, in terms of the prediction accuracy [13].

This paper suggested an approach which is capable of detecting multi-class classification chest diseases using Deep Convolutional Neural Networks architecture which might predict various chest diseases like Pneumonia, Pneumothorax, Atelectasis, Effusion, etc. with important accuracy. It provided other acuteness about the analysis which performed by generating heat maps, visualizations and localized the pathology by generating Class Activation Maps (CAM).

They used the ChestX-ray14 dataset, which contains around thirty thousand X-ray images, and every image is automatically annotated using automatic extraction methods, leading to 14 different classes of diseases. For the detection task, the dataset has been split into training, validation, and test zone. The images were downgraded to 224224 and normalized supported the mean and variance of images within the training set. They performed data augmentation using horizontal flipping. To appraise this network, the ROC score is employed and generated heatmaps to visualize the portion of images for analyzing the model predictions [14].

In this paper, a scientific evaluation of various approaches for CNN-based X-ray classification on ChestX-ray14. While satisfactory results were obtained with networks optimized on the ImageNet dataset, it's supported transfer learning (ResNet-50, ResNet-101, ResNet-38) with and without fine-tuning as well as the training of a dedicated X-ray network from scratch. To increase ChestX-ray14, data augmentation is employed.

The dataset has been divided into 70% training, 10% validation, and 20% testing. In training, different sized patches of the image are sampled, with sizes ranging between 8% and 100% of the image. For validation and testing, Images are resized to 256 * 256 and 480 * 480 pixels for tiny and substantial spatial sizes. To possess an excellent comparison to other groups, it reported results on this segmentation for best-performing architecture with different depths between ResNet38, ResNet-50, and ResNet-101 [15].

This paper proposed the convolutional neural network (CNN), which is intended for the diagnosis of chest diseases. Backpropagation neural networks (BPNNs) with supervised learning and competitive neural networks (CpNNs) with unsupervised learning are carried out for the classification of the chest X-ray diseases for comparative analysis. All the considered networks CNN, BPNN, and CpNN, are trained and tested on the identical chest X-ray database containing different diseases, and the performance of each network is discussed.

Moreover, BPNN and CPNN networks are trained using 620 out of 1000 images, and rest are used for testing. CNN is trained using 70% of 120120 available data, and 30% is used for testing. The general performances of the BPNN and CpNN are tested using 380 images and also the backpropagation networks using 32*32 pixels because of the input image size. It is seen from the paper that CNN has achieved the very best recognition rate for training and testing data, compared to other employed networks [16].

In this work, they appraised the functionality of pre-trained CNN models utilized as feature-extractors followed by different classifiers for the classification of abnormal and normal chest X-Rays to prevent adverse consequences in such remote areas. According to statistical results, the pre-trained CNN models employed along with supervised classifier algorithms, can be very beneficial in analyzing chest X-ray images, specifically to detect Pneumonia. The dataset ChestX-ray14 has been used for this work.

The original 3-channel images were resized from 1024*1024 into 224*224 pixels, and the pre-trained (DenseNet-169) has been used for the feature extraction process. After feature extraction, different classifiers such as Random Forest, Support Vector Machine, etc. were used for the classification task. But the best results were found to be attained when the Support vector machine was used as a classifier. So, in the best-proposed model, features extracted from DenseNet-169 were used with SVM classifier to accomplish better results [17].

In this paper, an improved self-adaptive filtering algorithm is proposed, of which a linear decay function generates the filtering coefficient. It used as a Field Programmable Gate Array (FPGA), and other peripherals have designed and implemented an X-ray medical image pre-processing system based on an advanced self-adapted filter. The proposed algorithm effectively improves the signal-to-noise ratio of the image after 4 to 8 frame iterations.

Recursive self-adaptive filtering algorithm has been applied to X-ray medical image pre-processing, using the integrated circuits (IC), including analogue-to-digital (AD) converter, frame memory, and other control circuits. After the X-ray medical image pre-processing system is designed and implemented, they have applied it to a real DSI system successfully. From the experimental results, the noises are effectively reduced after 4-frame recursions, and the effect is better after 8-frame recursions [18].

An approach has been explored about DCNN which based on abnormality detection in

frontal chest X-Rays in this paper. Some dataset has been used those are publicly available like Indiana's chest X-Ray dataset, JSRT dataset, Shenzhen Dataset. They employed heat maps obtained from occlusion sensitivity as a measure of localization in the CXRs. They applied the techniques developed along the way to the problem of tuberculosis detection on a different dataset.

The efficacy of various DCNs for the detection of heart disease in chest X-rays has been explored where the binary classification against standard chest X-rays is used as a representative example. Each of the DCNs, like AlexNet, VGG-16, VGG-19, ResNet-50, ResNet-101, and ResNet-152, are used to detect cardiomegaly. Moreover, a total of 24 networks were trained on the same training data. The methods include linear averaging, bagging, boosting, stacked regression, etc. Albeit ResNet-101 gives the highest specificity and VGG-16 gives the highest sensitivity, VGG-19 delivers an overall better performance [19].

This Paper propounded an idea about a 26-layer deep learning model for large scale plant classification in the natural environment. The proposed model achieves sensation rate of 91.78% on the BJFU100 dataset, which consists of 10,000 images of 100 plant species. Moreover, The 100 samples of each class are split into 80 training samples, and 20 test samples which Compared with conventional classification methods, data preprocessing on deep learning approaches is much simpler.

In this network, the input images are RGB colour. All the images are rescaled to 224224 pixels. The model parameter is trained by the stochastic gradient descent (SGD) algorithm during the back-propagation phase. To find the best deep residual network among the proposed ResNet-26 model and the original ResNet model of 18, 34, and 50 layers designed for ImageNet, the recommended ResNet-26 results in 91.78% accuracy [20].

In this experiment, they explored the ability of a non-medical dataset (ImageNet) to identify different types of pathologies in chest x-rays. The combination of CNN and GIST features gave the best result. Decaf5 baseline descriptor is an adequate descriptor for chest x-rays retrieval. The results can be improved by fusing the baseline descriptors of Decaf5, Decaf6, and GIST. Future work includes tuning of CNN with actual x-ray data.

They focused on a Decaf pre-trained CNN model, which is an adaptation of a CNN which closely follows the CNN (using ImageNet dataset). For selecting and combining feature sets descriptors like GIST and BAG-of-Visual-Words (BoVW) were used. All features used in their work were normalized. Following normalization, they applied the fusion of different feature groups. Finally, binary classification was done using SVM [21].

They experimented with CNN to identify different types of pathologies in chest x-ray images. A deep learning approach based on non-medical learning was explored. Decaf and PiCodes features together achieved the best performance. Additionally, they showed that Decaf layer 5 achieves better results than layers 6 and 7. It proved large scale non-medical

image databases are enough for medical image recognition tasks.

A CNN was trained over a subset of non-medical images from ImageNet. The decaf implementation of CNN was used to extract the main descriptor. "Picture Codes" (PiCoDes) descriptor is another baseline descriptor that has been used here. LBP and GIST were also used as descriptors. Then the data were standardized. At last, classification was performed using SVM [22].

This paper proposed an algorithm that can detect pneumonia from chest X-rays. The CheXNet is a 121-layer convolutional neural network trained on ChestX-ray14 which contains 112,120 X-ray images from Kaggle. Early diagnosis and treatment of pneumonia can prevent the risk of death. For that reason, they extend their method to detect all 14 diseases in ChestX-ray14 and achieve significant results.

They formulated the pneumonia detection task as a binary classification problem. For the pneumonia detection task, the dataset was randomly split into training, validation, and test set. Then, they normalized before inputting to the network, and the weights of the network were initialized with weights from a model trained on the ImageNet dataset. The dataset has been augmented into training data using random horizontal flipping to extend the quality and size of the dataset. Its performance was compared to other methods after running the algorithm, and finally, they made some changes to detect all 14 diseases and achieved the state of the art results [23].

The primary look at that display that deep studying has the capability for automated image analysis of radiological pictures in the scientific speciality. They used the Vi Di Suite 2.0 software and decided on 28 times with, and 24 cases without hemopericardium. Most effective classification classified cases of hemopericardium correctly with only a few fake positives. They wanted to test the feasibility of strategies for forensic imaging through deep mastering techniques that can discover and section a hemopericardium in positioned up mortem computed tomography. They randomly decided on 50% of the photographs for training, and also the relaxation for validation and repeated this approach 20 instances employing an extraordinary randomization approach. In the second step, a 2nd anomaly detection device comes to be educated to come across and segment blood, the use of manually segmented schooling datasets. To assess the feasibility of Vi Di for segmenting the blood within the pericardium, they calculated the do not forget precision and F-score for the detected region the use of the segmented pixels of the picture as floor truth. For automated detection of hemopericardium, they determined to use the picture category device (Vi Di green), which lets in for supervised photograph class supported pre-classified pics. Subsequently, appropriate algorithms and strategies through which the Vi Di software program handles and strategies the picture data don 't seem to be disclosed. Forensic imaging must appreciably take pleasure in the computerized assessment [24].

In this paper, a unique Hypothesis- CNN-Pooling (HCP) framework, a Convolutional Neural community is taken for inputs. Then a shared CNN is an installation with each speculation, and in the end output effects from particular hypotheses are mixed with max-pooling for the last multi-label predictions. But how CNN incredibly copes with multi-label photos remains an open hassle, mainly because of the complicated underlying item layouts and inadequate multi-label schooling pix. The proposed HCP is based totally on the notion that the enter hypotheses can cover all single gadgets of the given multi-label image, which requires an immoderate detection maintain in mind charge. After that, they retrained the proposed HCP with a squared loss characteristic for the very last prediction. Second, the final absolutely-connected layer of the community is replaced with a c-manner genuinely-related layer, wherein c is the category variety of the intention multi-label dataset. Then an image quality-tuning (I-toes) technique is accompanied to initialize the final really-linked layer via making the use of the target multi-label image set as inputs. They experimentally adopt suggestion strategies, i.e., BING and Edge Boxes for the hypotheses era because of their immoderate computational performance and high object detection. Besides the proposed HCP calls for no bounding subject annotation for schooling, and subsequently can effortlessly adapt to new multi-label datasets [25].

The best sensitivity and specificity outcomes within the present check show that automated dynamic breath sound pictures can be satisfactorily analyzed through the manner of educated physicians to distinguish among sufferers with pneumonia or pleural effusion. They recorded breath sounds from 20 sufferers conventionally diagnosed as having a pleural effusion, 20 sufferers conventionally identified as having pneumonia, and 60 wholesome controls, of whom 20 served as a mastering sample. For deciding whether breath sound distribution maps can differentiate among patients with pneumonia or pleural effusion, an automated machine that detects and shows both breath sound distribution has higher accuracy than the gadget they studied. Recordings have been carried out in a quiet but not soundproof room and the captured lung sound signals were band-skip filtered (150–250 Hz), which allowed best the desired frequency variety of breath sounds. The signals have been processed, and the breath sound distribution turned into displayed as a 2-dimensional dynamic grayscale photograph with 256 grey degrees. Taking a look at participants were enrolled from 3 Israeli fitness offerings. There has been no high distinction among the breaths sound photograph evaluation of “unusual” or “ordinary” for sufferers identified as having pleural effusion pneumonia. Additionally, just like radiographs, the assessment of the dynamic lung sounds photograph depends on the reader’s capability to figure between qualitative everyday and strange characteristics [26].

This study proposed an intelligent system that developed an accurate, rapid, and cost-effective model of malaria diagnosis using smeared thin blood smear images. ANN (Artificial Neural Network) is used for classification. The image sizes are rescaled to 300x300 pixels

using MATLAB function resize, and all the images are converted from RGB to grayscale, using MATLAB function gb2gray. Noise is removed from images by doing filtering operation using a square median. Intensity Features and Threshold Features are extracted from the processed images. Then the neural network is trained with the backpropagation algorithm [27].

In this paper, Multilayer Perceptron Backpropagation, which is one of the architectures of ANN, has been used to classify the images. Images are preprocessed by enhancing the contrast and filtering noise. Images are converted from RGB to HSV. K-means clustering and morphological algorithm are used for pattern-clustering and post-reconstruction. A morphological algorithm is used to extract filling the hole and eliminate the unwanted cell. Histogram based texture consists of features, i.e., mean, standard deviation, skewness, energy, entropy, smoothness, and kurtosis are obtained. Then the classification algorithm is applied [28].

This paper aim is to develop a sensitive, unsupervised malaria detection system that will cut down the cost of the material and will reduce human reliance. Two classification algorithm SVM (Support Vector Machine) and Neural Network (NN) have been used to detect whether the sample is affected by the malaria parasite or not. Images are converted from RGB to Gray Scale for faster processing. Using the GLCM (Gray Level Co-occurrence Matrix), the features were extracted from the pictures and stored in a database. The extracted features are mean, standard deviation, skewness, kurtosis, energy. Then the classification algorithms are applied to the features [29].

In his paper proposed an automated image recognition techniques based on a convolutional neural network. Therefore massive data have been applied to both thick and thin malaria blood smears for microscopic diagnosis. Additionally, they proposed a frequent model system based on a convolutional neural network (CNN) to automatically classify individual cells in thin blood smears as either infected or uninfected. For this analysis, image size is taken 44x44 with three colour channels and tries to obtain the normalized technique to uplift local contrast, brightness, and blanch the entire dataset using an eigenvalue decomposition method. Therefore, a 17 layer CNN model is applied and also implemented a ten-fold cross-validation technique over the whole data set where 90% and 10% of the images are used for training and testing, respectively in classification. Finally, the SVM classifier is used for feature extractor based on the CIFAR-100 data set [30].

Considering this paper, they discussed the methods of compiling a pathologist-synthesized image dataset for a deep neural network training and the methods of data augmentation to significantly increase the size of the dataset in light of the excessive problem involved in the training of deep neural networks. They compared the accuracy of the classification obtained by deep convoluted neural networks through training, verification, and testing with different

combinations of datasets. This comparative study indicates that data augmentation in the feature domain appears to be more robust in preserving the accuracy of higher classification [31].

This paper is based on a method of deep learning in end-to-end systems that perform both feature extraction and grading directly on blood cells. They focused on improving malaria detection from patches divided into microscopic images of red blood cell smears by introducing deep blood neural networks. They showed that using transfer learning, which is considered a prominent technique in computer vision, it is entirely possible to achieve absolute rather good performance compared to other traditional machine learning techniques [32].

Table 2.1: Summary Table

Papers	Datasets	Methods	Results
1.	Guangzhou Women and Children's Medical Center	AlexNet, DenseNet121, ResNet18, Inception V3, GoogLeNet	Test accuracy of 96.39%, with an area under the ROC curve of 99.34%.
2.	A hospital-scale chest X-ray image database, namely "ChestX-ray8", from PACS	Caffe framework, AlexNet, GoogleNet, VGGNet-16, and ResNet-50	-
3.	The dataset named chest X-Ray & CT dataset	DL architectures (CNN, VGG16, VGG19, DenseNet201, Inception-ResNet-V2, Inception-V3, Xception, Resnet50, and MobileNet_V2)	Accuracy is more than 96%
4.	ChestX-ray14 database formulated by the U.S. National Institute of Health Clinical Center, PACS	Wavelet Transform, DNN, ResNet50	-
5.	ChestX-ray14 database formulated by the U.S. National Institute of Health Clinical Center, PACS	CNN, LeNet-5, Deep Learning	The average accuracy of 89.77% is achieved for the classification of different diseases

Table 2.1: Summary Table

Papers	Datasets	Methods	Results
6.	ChestX-ray14 database formulated by the U.S. National Institute of Health Clinical Center, PACS	CNN, VGG, STN, CapsNet basic, Vanilla gray	As a result of mode-CNN, F 0.5 score is 68% with accuracy of 71%
7.	There are two datasets. Radiological Society from North America (RSNA) has published this dataset and another dataset (CXI) is developed in the Guangzhou Women and Children's Medical Center Guangzhou, China.	CNN, ResNet-50, Xception, and VGG16	The trained model has achieved an accuracy of 96.76%
8.	ChestX-ray14 database formulated by the U.S. National Institute of Health Clinical Center, PACS	CNN, VGG16, Local Interpretable Model-Agnostic Explanations (LIME)	The trained model has achieved an accuracy of 94.3%
9.	The dataset named chest X-Ray & CT dataset which obtained from Kaggle	Data Augmentation, Convolutional Neural Network (CNN)	The final results obtained are training accuracy: 95.31%, and validation accuracy of 93.73%
10.	The dataset named chest X-Ray & CT dataset which obtained from Kaggle	F-cMeans, CS based DL, and ChexNet, Conventional CS, PM	Proposed approach enables detection of pneumonia with 97.34% prediction accuracy
11.	ChestX-ray14 database formulated by the U.S. National Institute of Health Clinical Center, PACS	ResNet, Deep Neural Networks, Class Activation Mapping (Grad-CAM)	Model trained with significant accuracy is more than 80%

Table 2.1: Summary Table

Papers	Datasets	Methods	Results
12.	ChestX-ray14 database formulated by the U.S. National Institute of Health Clinical Center, PACS	ResNet-50, RestNet-101, ResNet-38, Grad-CAM, CNN	Model predicted with a very high AUC of 99.83%
13.	ChestX-ray14 database formulated by the U.S. National Institute of Health Clinical Center, PACS	CNN, backpropagation neural networks (BPNN), and competitive neural networks (CpNN)	Accuracy is 92.4% obtained from CNN
14.	ChestX-ray14 database formulated by the U.S. National Institute of Health Clinical Center, PACS	DenseNet, VGG, ResNet, Xception, SVM, Deep Convolutional Neural Networks, Transfer Learning, Random Forest, Naive Bayes, K-nearest neighbors	Model predicted with a very high AUC of 80.02%
15.	Source not mentioned	Self-adaptive filter, FPGA, digital-to-analog (DA) converter	-
16.	There are three publicly available datasets. Indiana Dataset from Indiana University School of Medicine, JSRT Dataset from Japanese Society of Radiological Technology (JSRT), Shenzhen Dataset from Medical College, Shenzhen, China	AlexNet, VGG-16, VGG-19, ResNet-50, ResNet-101, ResNet-152	Accuracy is 92%
17.	The BJFU100 dataset which is collected from Beijing Forestry University campus	ResNet-18, ResNet-34, and ResNet-50	Accuracy is 91.78% obtained from ResNet26

Table 2.1: Summary Table

Papers	Datasets	Methods	Results
18.	The Dataset were acquired from the Diagnostic Imaging Department of Sheba Medical Center (Tel-Hashomer, Israel)	Pre-Learned CNN (Decaf), GIST, SVM BoVW, Late Fusion	Accuracy is 94%
19.	The Dataset were acquired from the Diagnostic Imaging Department of Sheba Medical Center (Tel-Hashomer, Israel)	Pre-Learned CNN (Decaf), GIST, SVM BoVW, Late Fusion	Accuracy is 93%
20.	ChestX-ray14 database formulated by the U.S. National Institute of Health Clinical Center, PACS	ChexNet, Class Activation Mapping (CAMs)	-
21.	PASCAL Visual Object Classes Challenge (VOC) datasets	AlexNet-24, VGGNet-16, SVM, Hypotheses CNN-Pooling (HCP) framework	Accuracy is 93.2%
22.	Source not mentioned	Deep learning, PMCT	-
23.	Center for Disease Control (CDC) and Reference Laboratory of Malaria, Ministry of Health	Median filtering, Back Propagation Neural Network (BP-ANN)	96% accuracy in parasite detection
24.	Parasitology Laboratory, Faculty of Medicine, University of Gadjah Mada, Indonesia	K-means Clustering, morphological algorithm, multilayer perceptron backpropagation	The proposed method achieves accuracy of 87.8%
25.	Source Not Mentioned	Support Vector Machine (SVM), Neural Network (NN)	Accuracy using support vector machine (SVM) is 97.25%
26.	Chittagong Medical College Hospital, Bangladesh	Convolutional Neural Network (CNN), transfer learning	Average classification accuracy of the CNN model is 97.37%

Table 2.1: Summary Table

Papers	Datasets	Methods	Results
27.	Dataset from University of Alabama at Birmingham	Deep convolutional neural networks	Accuracy is 95%
28.	National Institute of Health named NIH Malaria Dataset	Deep convolutional neural networks	Accuracy is 92.12%

Chapter 3

Background Study

In the past years, many researchers have revealed some detection systems for Chest diseases using different techniques to increase the accuracy of the classification of the diseases to select an appropriate treatment which includes Bronchoscopy, Chest Tube Procedure, CT Scan, CT Scan-Guided Lung Biopsy, Lung Function Tests, X-ray, etc. Among them, an X-ray scan of the chest is a modality often used by radiologists to diagnose many chest-related diseases at an early stage. A chest X-ray produces images of the heart, lungs, airways, blood vessels, and spinal cord and chest bones that help physicians to identify and treat the condition. But those tests need much time for the detection process, and sometimes they have a lower accuracy rate. So, an automated system is required to minimize this abridgment. There are some newly proposed methods based on Learning Vector Quantization (LVQ), Deep neural networks (DNN), which are powerful classification algorithms. In that case, we will work on Deep Learning and related preprocessing system, which is explained in the following sections.

3.1 Description of Lung Diseases

3.1.1 Pneumonia

Pneumonia is a disease that causes an infection in one or both lungs. A variety of microscopic organisms that will cause pneumonia, including bacteria, viruses, and fungi. Infection with a virus or bacteria causes a type of inflammation of the air sacs in the lungs called alveoli [33]. As a result, the alveoli become full of pus, which causes phlegm, fever, cough, and difficulty of breathing. The main varieties of pneumonia are Bacterial pneumonia and Viral pneumonia. Bacterial pneumonia is caused by various bacteria and viral pneumonia, which is caused by various viruses and liable for about one-third of all pneumonia cases. Pneumo-

nia is characterized by a variety of symptoms such as fever, heavy sweating, anorexia, loss of energy and excessive fatigue, rapid breathing, and chest pain [34]. Sometimes pneumonia is so severe that it can even lead to death. Furthermore, some tests have long been used to diagnose pneumonia. Among these, chest X-rays, blood tests, bronchoscopy, etc. are notable.

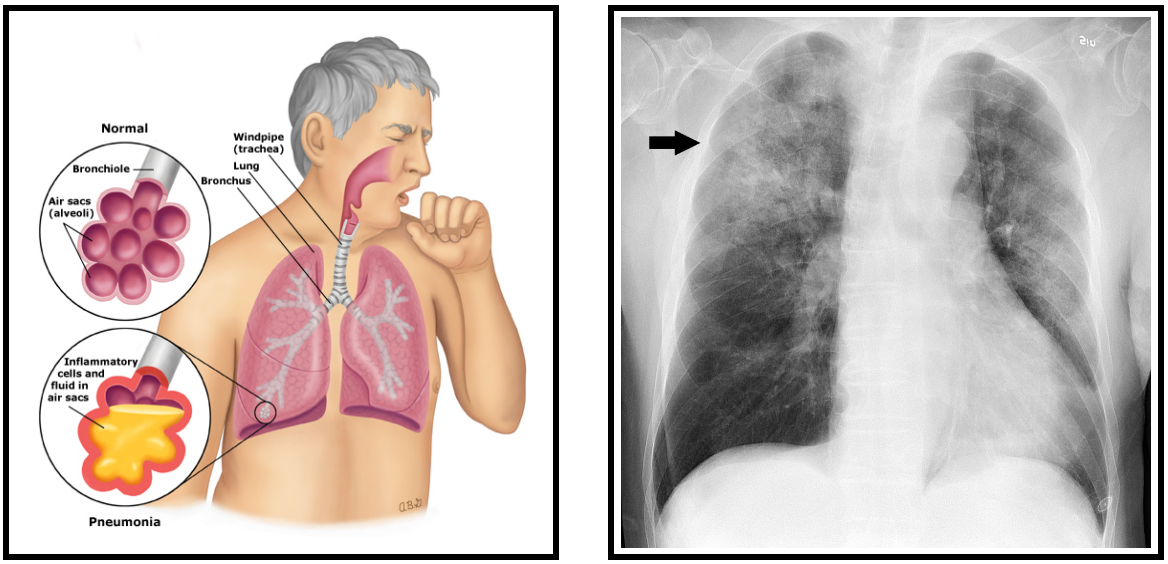


Figure 3.1: Pneumonia

3.1.2 Pleural Effusion

An effusion is a type of fluid formation in the space between the layers of tissues and the chest cavity called the pleural space. It is also called effusion or pulmonary infusion. In a healthy lung, these membranes hold a small amount of fluid between the lungs and the chest so that the lungs can expand and contract properly during respiration [35]. The amount of this fluid is more in the pleural space of the infected person. The disease is detected in the lungs by X-ray of the chest, which is often seen as a white area at the base of the lungs. There are many causes of pleural effusions such as Heart failure, Kidney failure, Infection, Lung cancer, etc fig. 3.2.

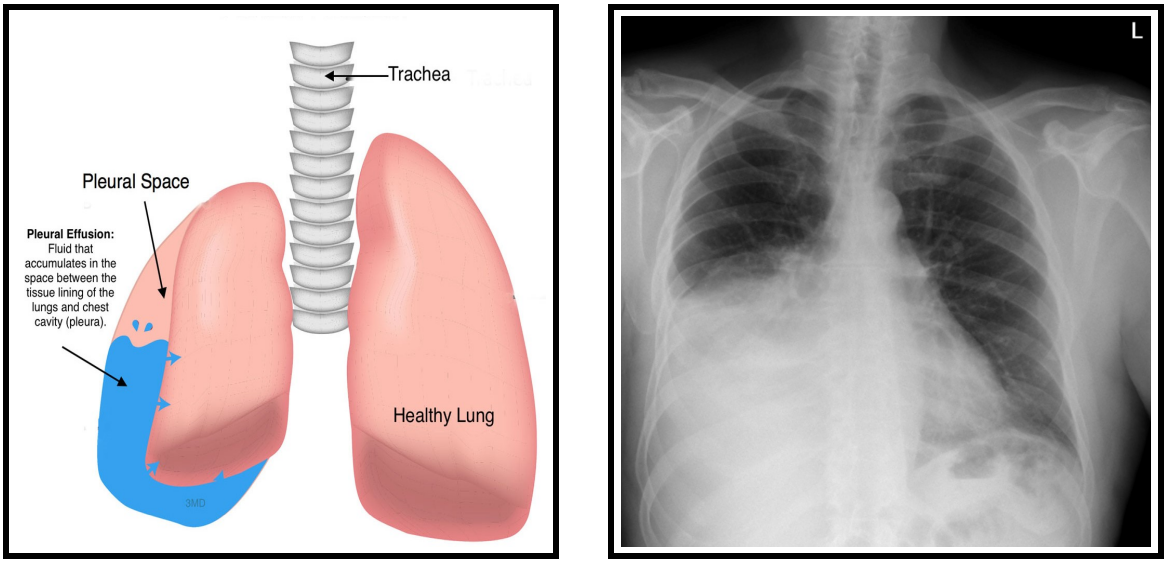


Figure 3.2: Pleural Effusion

3.2 Image Enhancement

Image enhancement is the process of combining digital images to make the results more suitable for image display or further image analysis. Enhancing an image provides better contrast and a more detailed image as compare to a non-enhanced image. It is used to improve medical images, images captured in remote sensing, images from satellite, etc. Furthermore, it can remove noise, increase sharpness, or illuminate an image and enhanced image details, making it easier to identify key features.

3.2.1 CLAHE (Contrast Limited Adaptive Histogram Equalization)

Adaptive histogram equalization (AHE) is an image-processing technique used to improve contrast in the images. It differs from conventional histogram equalization in the respect that the adaptive method computes several histograms. It is corresponding to a distinct section of the image and uses them to redistribute the lightness values of the image [36]. Therefore, it is suitable for improving the local contrast and enhancing the definitions of edges in each image region. Moreover, Clip-limit is the contrast limit for localized changes in contrast, which is favorable between 2 to 3 fig. 3.3.

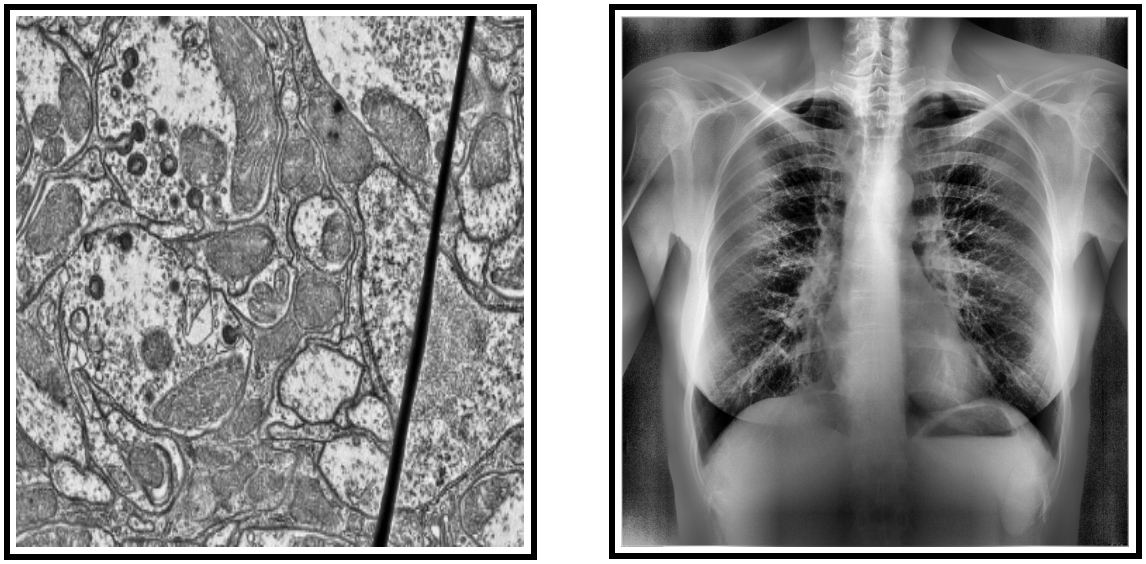


Figure 3.3: Contrast Limited AHE

3.2.2 Power Law Transform

The mathematical form of Power-law transformation function is

$$S = c * r^{\gamma} \tag{3.1}$$

Where S and r is the output and input pixel values, respectively, and c and γ are the positive constants. It is also known as gamma correction and generated for r values normalized from 0 to 1 [37]. If rmax has significant value (in the range 150-255), then it is seen from the above graph that contrast stretching occurs by choosing $\gamma > 1$. Similarly, for $\gamma < 1$, then the reverse result will be arisen, shown in the figure below.

Figure 3.4: Power Law Transform

3.3 Deep Learning

Deep learning (DL) is an artificial intelligence function that mimics the procedures of the human brain in creating patterns to be used in processing and decision making. It is a subset of Machine Learning (ML). Our brain's neurons inspire deep learning. It uses multiple layers to extract a feature from raw input [38]. There are many types of deep learning architectures such as Deep Neural Network, Deep Belief Network, Recurrent Neural Network, and Convolutional Neural Network. In Deep Learning (DL), the model analyzes the data, learns to perform classification directly from images, text, or sounds and uses what has been learned from previous data to make predictions or determination about new data. Furthermore, the models were trained by using a large set of labeled data following by neural network architectures that comprise many layers. Therefore, this learning can occur mostly in two ways, i.e., Supervised Learning and Unsupervised Learning.

3.4 Supervised and Unsupervised Learning

Supervised learning occurs when a model already knows the label of the data, which will be used to train the model. This means that some data has already been tagged with the correct answer and the labelled data helps to predict the outcome. Contrariwise, unsupervised learning is the opposite of supervised learning. In that learning, the model does not know the label of data that will be used to train it. It deals with unlabeled data and can perform more complex processing tasks than supervised learning. Besides, supervised learning is the most commonly used and takes more time to successfully building [39].

3.5 Convolutional Neural Network (CNN)

In neural networks, the recognition of the Convolutional Neural Network (CNN) images is one of the major categories for the classification of images. Object detection, face detection, etc. are some of the areas where CNN is widely used. Moreover, CNN image classification takes input images, determines the importance (learning weight and bias) of different aspects of the image, and can distinguish one from the other and classify them into specific categories like dog, cat, etc. Computers see an input image as an array of pixels, depending on the resolution of the image. Based on the resolution of the image, it will see h*w*d (h = height, w = width, d = dimension) in fig. 3.5.

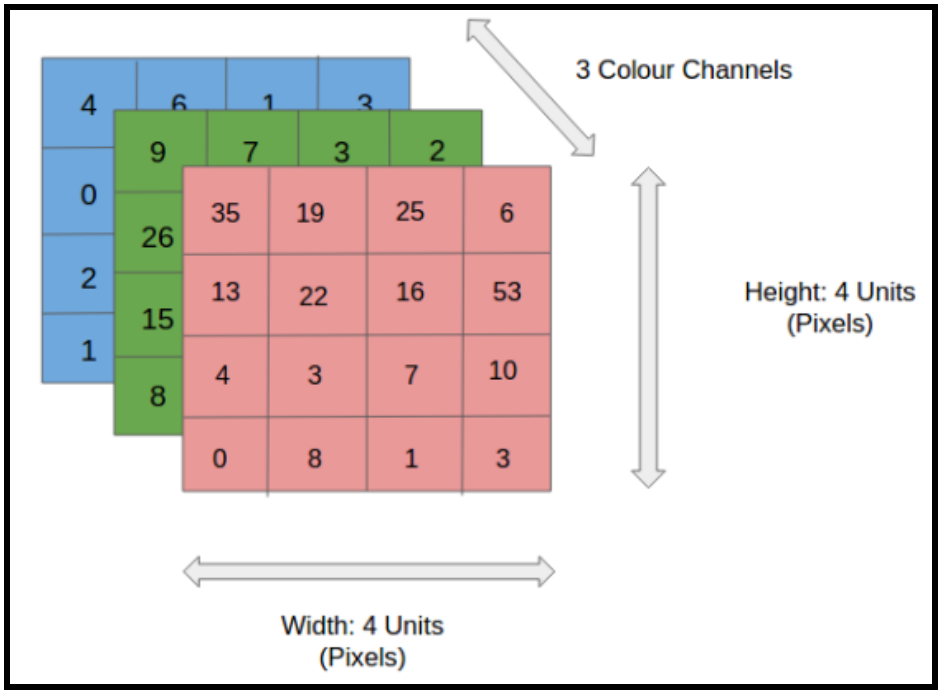


Figure 3.5: Array of RGB Matrix

Technically, to train and test in deep learning CNN models, each input image will pass it through multiple convolution layers with filters (kernels), pooling, fully connected layers (FC), and apply the Softmax function to classify an object with probabilistic values between 0 and 1 [40]. So, the complete flow of CNN to process an input image and classifies the objects based on values is given below-

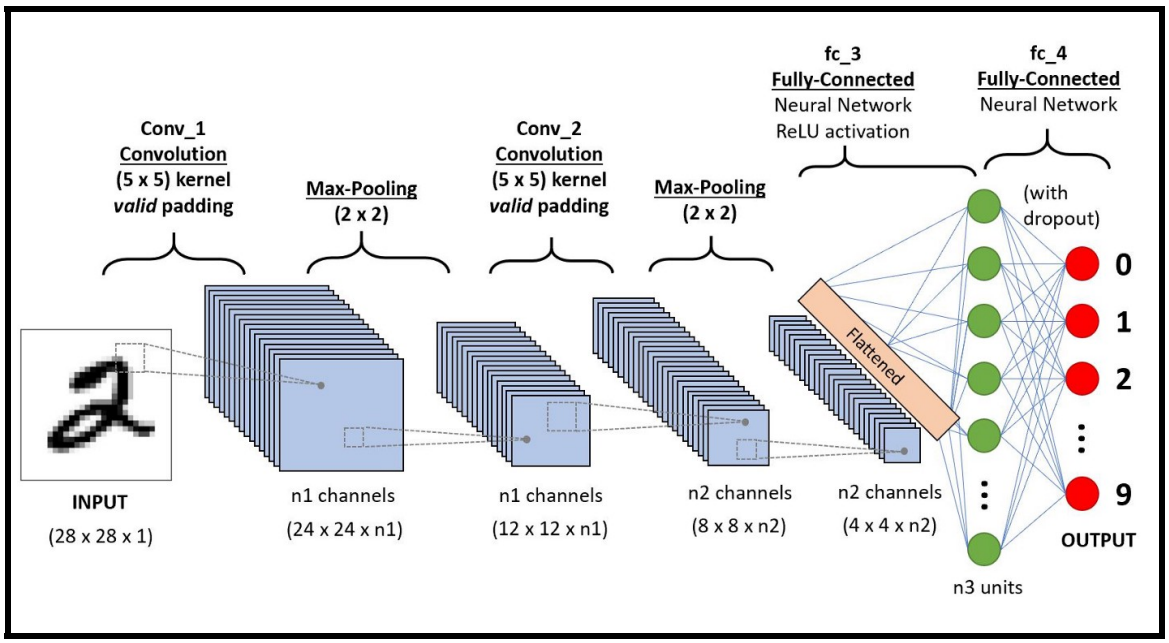


Figure 3.6: Typical architecture of a CNN

3.5.1 General Structure of Convolutional Neural Networks

A convolutional neural network generally consists of three types of layers: an input layer, an output layer, and multiple hidden layers. The hidden layers of a CNN typically include a series of convolutional layers that convolve with multiplication or other dot product.

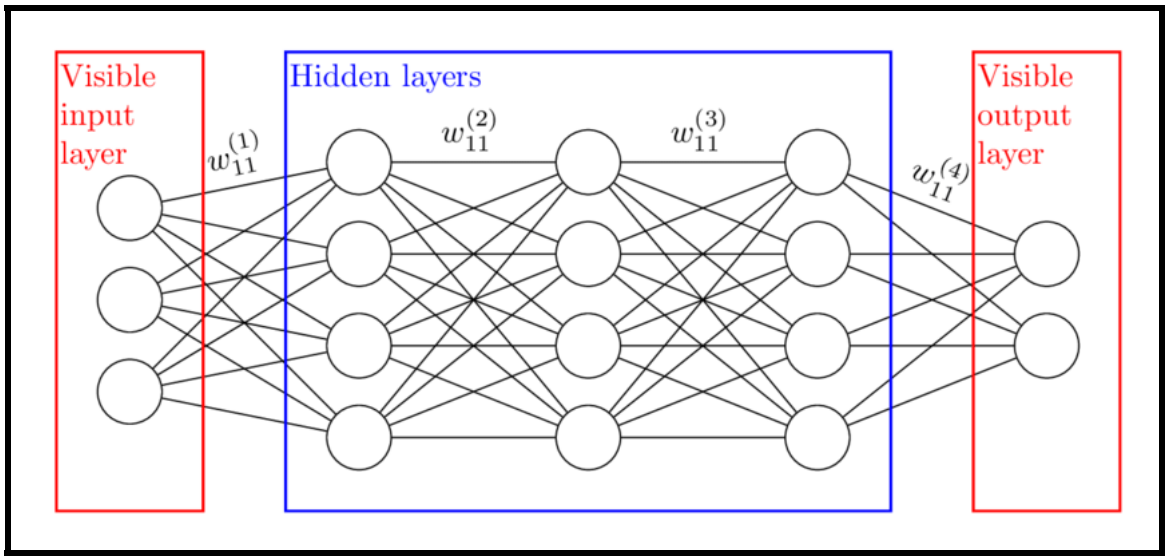


Figure 3.7: General Structure of Convolutional Neural Networks

- **Input Layer:** The input layer of a neural network is composed of artificial input neurons. It brings the initial data or input into the system for further processing by subsequent layers of artificial neurons.
- **Hidden Layer:** The input from the input layer is then passed to the hidden layers. There can be multiple hidden layers in a model depending on one's needs. Each hidden layer can hold a different number of neurons, which must be defined while creating the model. The size of the neuron in the hidden layer is generally more significant than the number of features. The output of each layer is achieved by computing matrix multiplication from the previous layer's output with the learnable weights of previous layers. After that, the activation function is used to make the network nonlinear.
- **Output Layer:** The output from the hidden layer is passed to the output layer, where a function like sigmoid or softmax converts each class's output into the probability of each class.

3.5.2 Convolution Layer

The convolutional layer is the fundamental building block of a CNN consisting of a set of learnable filters or kernels with a small acceptable field.The Convolutional layer is the first

layer that extracts features from the input image [40]. It maintains the relationship between the pixels by learning image features using small squares of input data. Mathematically, a convolution takes two functions f and g defined as-

$$(f * g)(i) = \sum_{j=1}^{m} g(j).f\ (i - j + m/2) \tag{3.2}$$

which is the dot production of the input matrix and a kernel function.

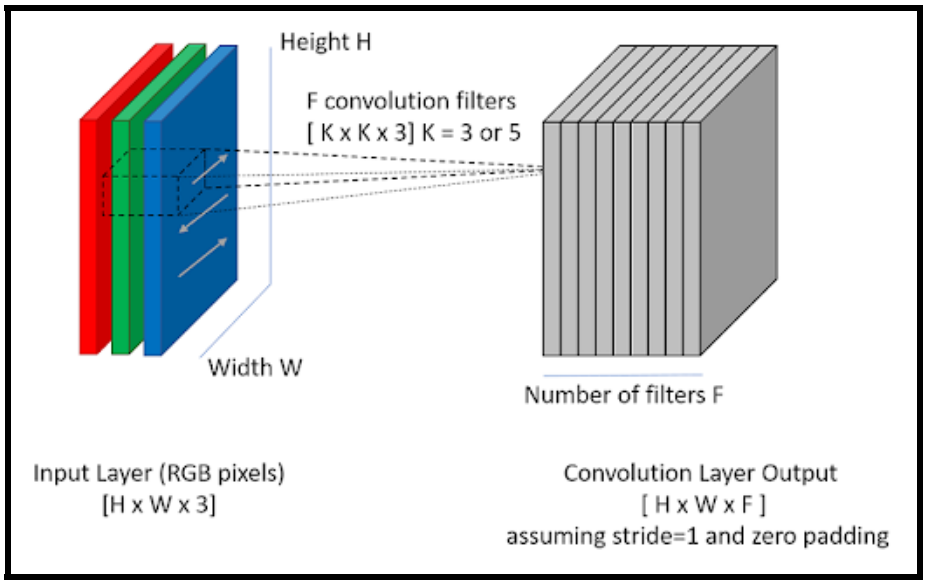


Figure 3.8: Convolution Layer

3.5.3 Pooling Layer

The pooling layer is a fundamental layer on CNN. It would decrease the number of parameters when the inputs are too large. Spatial pooling is also called submapping or downsampling which reduces the size of each map but contains important information.The pooling layer operates on each feature map independently [41] fig. 3.9. Spatial pooling can be of different types:

- Max Pooling Layer
- Average Pooling Layer

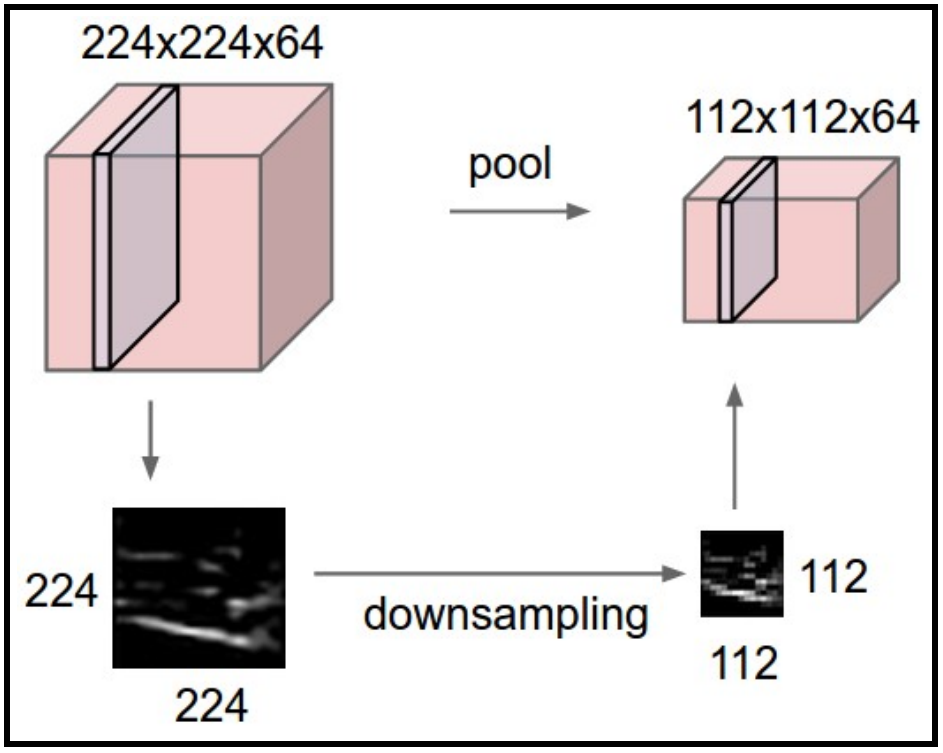


Figure 3.9: Pooling

3.5.3.1 Max Pooling Layer

The most common pooling layer used in the convolutional neural network is max pooling, which is a pooling operation that selects the maximum element from the region of the feature map covered by the filter or kernel [41]. It connects every neuron in one layer to every neuron in another layer and similar to Multi-Layer-Perceptron Neural Network(MLP). Thus, the output after the max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

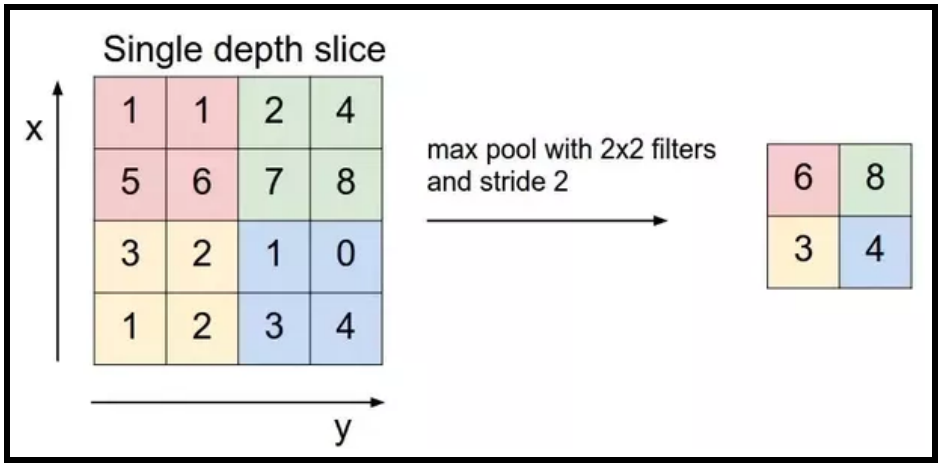


Figure 3.10: Max Pooling

3.5.3.2 Average Pooling Layer

An average pooling layer makes down-sampling by dividing the input into rectangular pooling regions and computing the average values of each region [41]. It calculates the average value for each patch on the feature map.

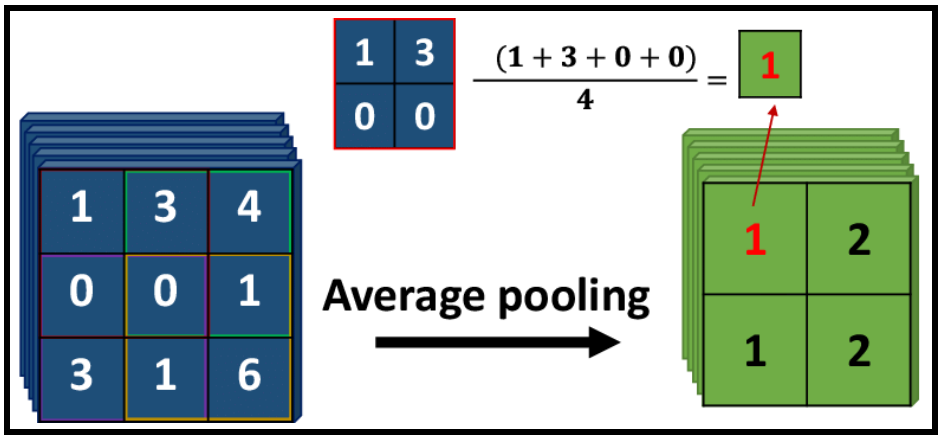


Figure 3.11: Average Pooling

3.5.4 Fully Connected Layer

Fully connected layers in neural networks are layers where all inputs from one layer are connected to each activation unit in the next layer. The first / input layer has three feature units, and the next hidden layer has four activation units. Each level has a bias unit of 1 [40].

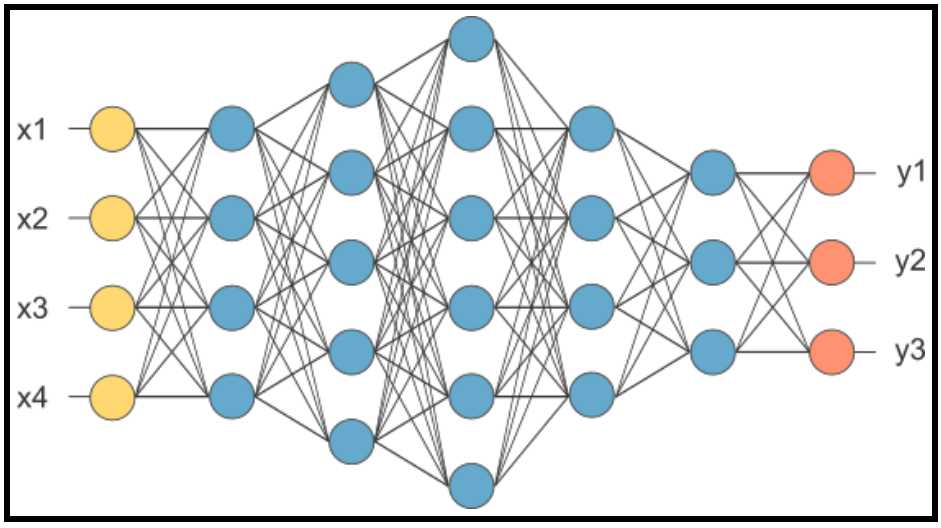


Figure 3.12: Fully Connected Layer

In the figure above, the feature map matrix is converted to a vector (x1, x2, x3). With the fully connected layers, it combined these features to create a model. Finally, it has an

activation function such as softmax or sigmoid to classify the outputs as a cat, car, truck, etc.

3.5.5 Activation Function

The activation function is used to determine whether a neuron should be fired or not by calculating the weighted sum of each connection pointing to the same neuron. The activation function helps to introduce non-linearity into the output of a neuron. There are major types of activation functions like ReLU (Rectified Linear Unit), step, Sigmoid, tanh, Softmax, etc.

3.5.5.1 ReLU

ReLU stands for a rectified linear unit is the most popular activation function, which normally applied in the hidden layer of Neural Network. R(z) = max (0, z). This function gives output z if z is positive otherwise 0 [42]. It is the most ordinarily used activation function on neural networks, especially CNN.

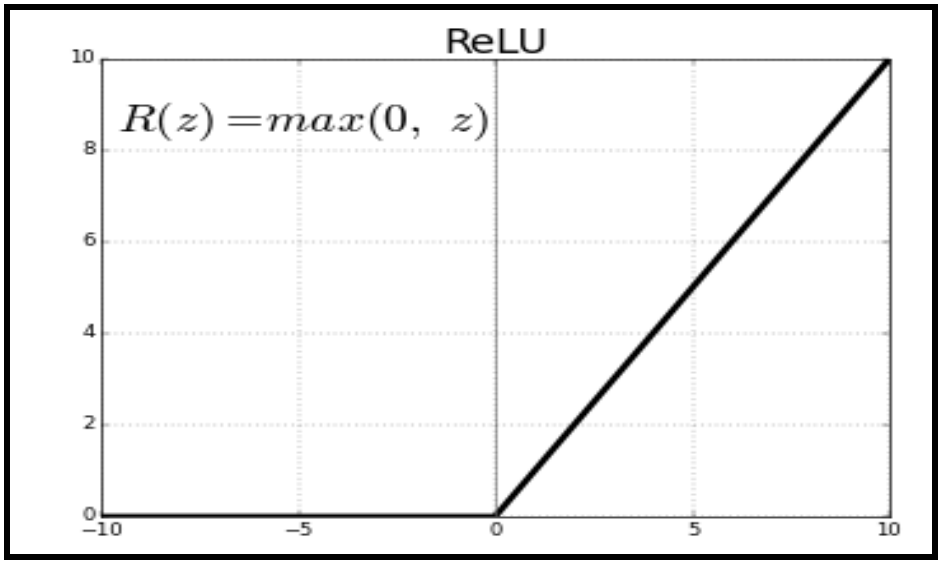


Figure 3.13: ReLU

3.5.5.2 Sigmoid Function

The sigmoid function's graph looks like 'S' shaped and it is a type of activation function mainly defined as a squashing function. Moreover, Squashing functions limit the output to a range between 0 and 1, making these functions useful in the prediction of probabilities [42]. It is beneficial when we need to predict the probability as output fig. 3.14.

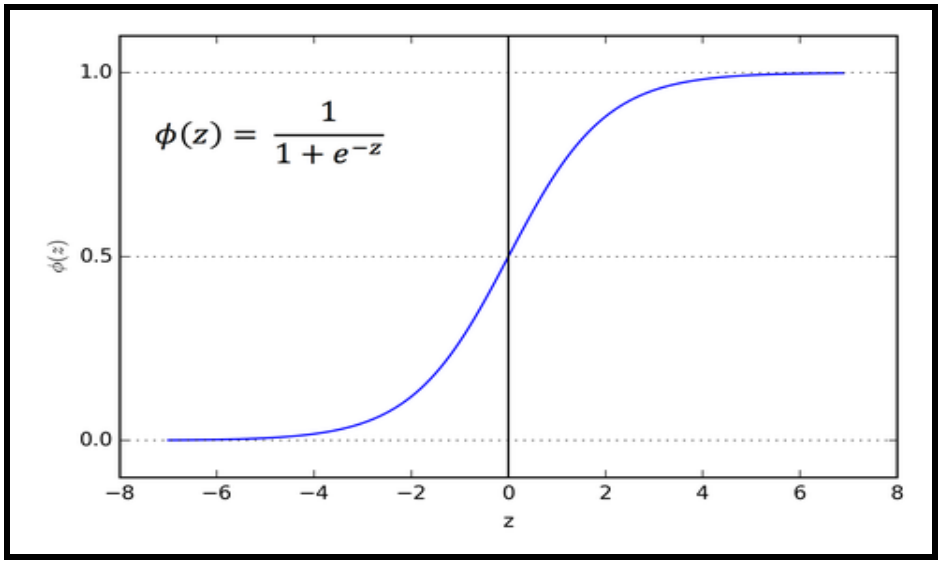


Figure 3.14: Sigmoid

3.5.5.3 Softmax Function

The softmax function is a kind of sigmoid function, but it is especially helpful when dealing with classification problems. It is mainly used to normalize neural network output to fit between zero and one [42]. It is also used to represent the certainty "probability" in the network output.The formula for softmax is as follows:

$$Softmax(y_i) = \frac{e^{y_i}}{\sum_{i=0}^{n} e^{y_i}} \tag{3.3}$$

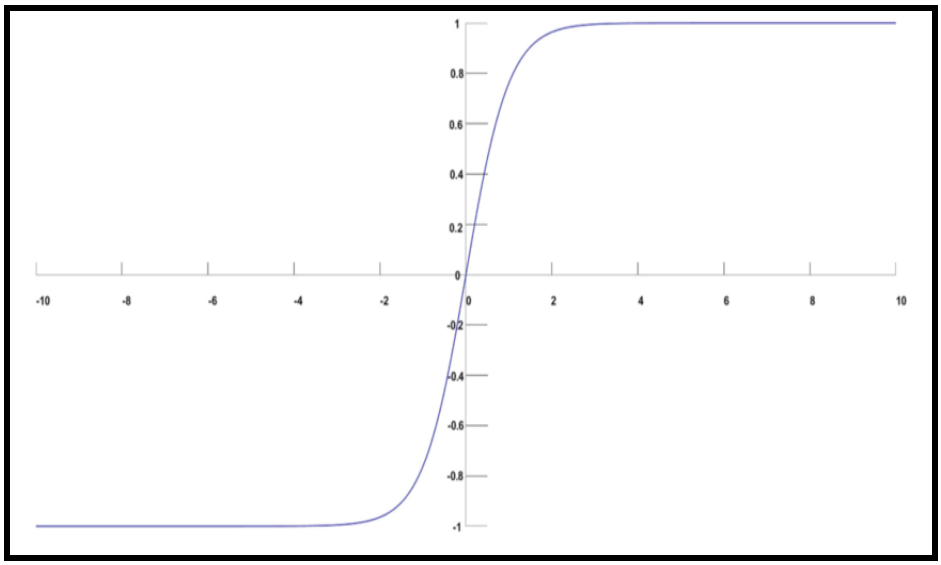


Figure 3.15: Softmax

3.5.5.4 Step Function

The step function is generally known as a common activation function in the neural networks. The function generates a binary output where the value of x is greater than or equal to zero, and the value of 0 is less than zero when the output is one [42]. As it seems, a step function is non-differentiable at zero.

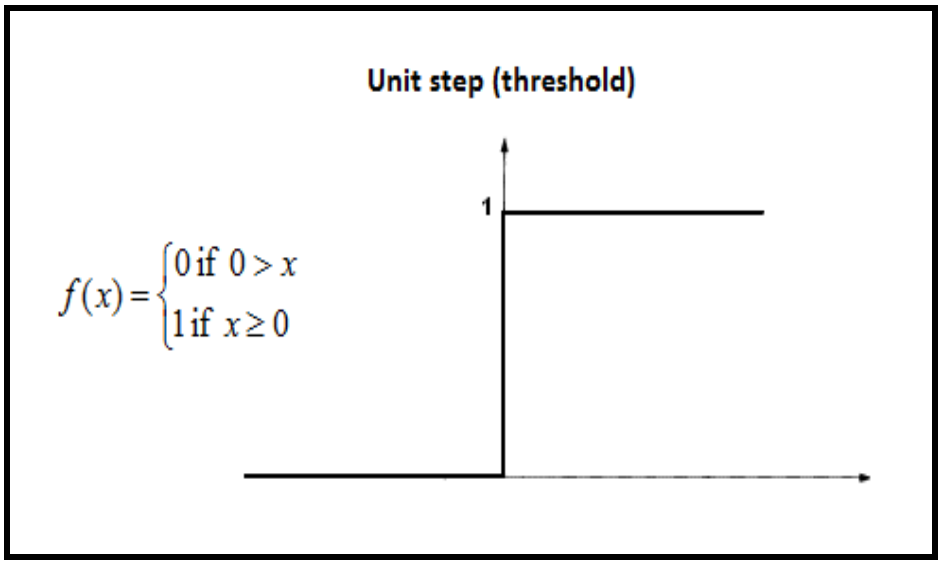


Figure 3.16: Step Function

3.5.6 Hyperparameters of Neural Network

Hyperparameters are the variables that consider the network structure (ex- Number of Hidden Units) and determine how the network is trained (ex- Learning Rate). Furthermore, Hyperparameter values are set before training (before optimizing the weights and bias). Some hyperparameters are given below.

3.5.6.1 Optimizer

Optimizers are algorithms or methods used to change the attributes of your neural network, such as weights and learning rate, to reduce the losses. Optimizers help to get results faster. There are several optimizers in deep leaning. One of them is Adam optimizer, which is an adaptive learning rate optimization algorithm that computes individual learning rates for different parameters [43].

3.5.6.2 Epoch

In Deep Learning, an epoch is a hyperparameter that is defined before training a model. One epoch is when an entire dataset is passed both forward and backward through the neural network. Usually, training a neural network takes more than a few epochs. For example: If we divide a dataset of 2000 training examples into 500 batches, then 4 iterations will complete 1 epoch.

3.5.6.3 Learning Rate

The amount of weight that is updated during training is referred to as step size or learning rate. Moreover, the learning rate is a configurable hyperparameter used in the training of neural networks with a small positive value that ranges between 0.0 and 1.0. Moreover, A traditional default value for the learning rate is 0.1 or 0.01.

3.5.6.4 Batch Size

The batch size is another hyperparameter of neural network that defines the number of samples to work through before updating the internal model parameters, and it must be more than or equal to one and less than or equal to the number of samples in the training dataset.

3.5.6.5 Loss Function

Loss function configuring is one of the most significant steps to ensure the model will work in an intended manner in any deep learning project. The loss function can give the neural networks a lot of practical flexibility and will define precisely how the network output is connected to the rest of the network. From a simple point of view, the loss function (J) can be defined as a function that takes two parameters- Predicted Output and True Output.This function will calculate how poorly our model is performing compared to the actual value of what the output is supposed to be. If Y_{pred} is too far from Y, the loss value will be very high. However, if both values are approximately equal, the loss value will be deficient. Therefore, it needs to keep a loss function, which can effectively penalize a model when training a dataset [44] fig. 3.17.

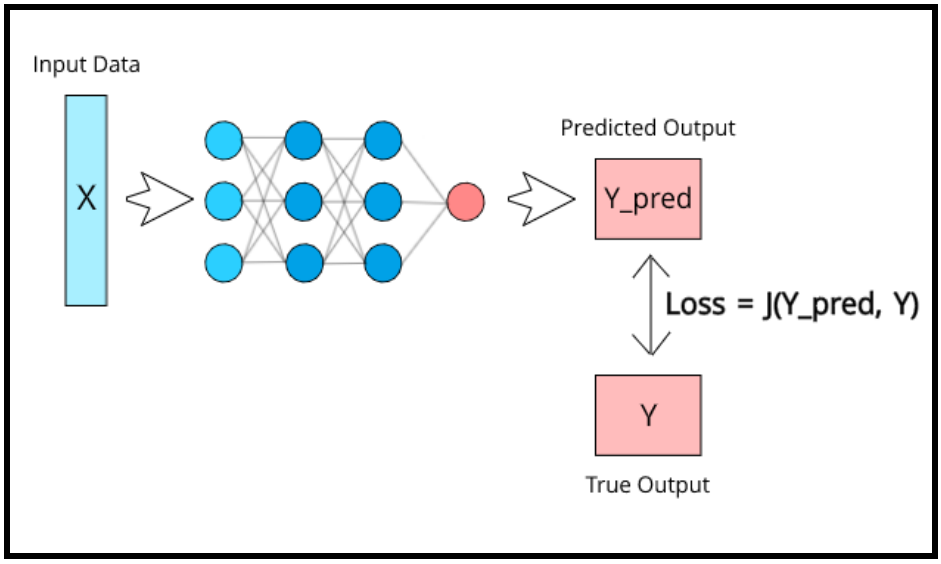


Figure 3.17: Loss Function

3.5.7 Multiclass Classification

Multiclass classification is appropriate when a model is needed to predict one possible class output every time. Since the probability is being worked out here, it would be understandable to apply sigmoid to all output nodes to get a value between 0-1 for all outputs. Moreover, the sigmoid function ensures that all the output nodes have values between 0–1, and the sum of all output node values equals to 1 always [44].

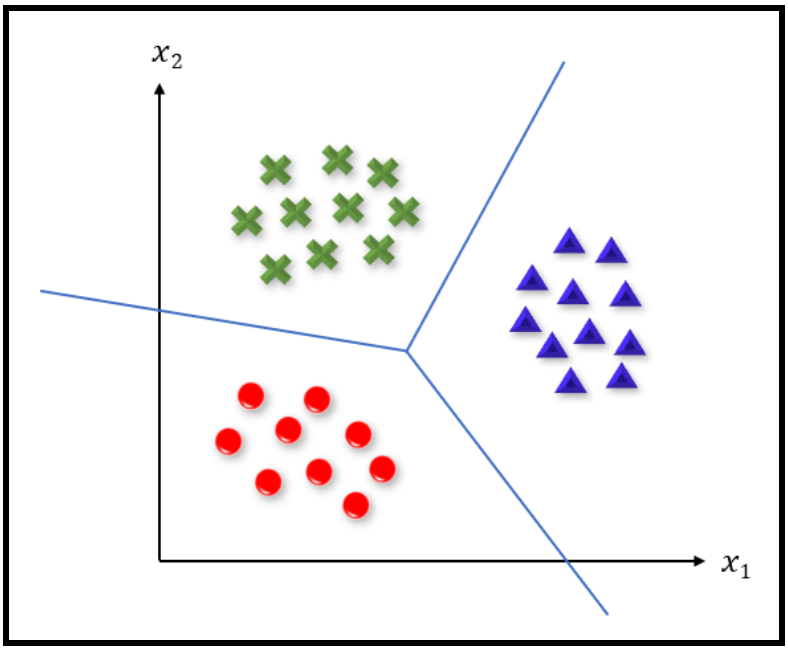


Figure 3.18: Multiclass classification

3.5.8 Logistic Regression

Logistic regression is the learning algorithm used in supervised learning problems when the output y are all zero or one. The goal of logistic regression is to minimize the error between its predictions and training data. In regression analysis, logistic regression is estimated parameters of a logistic model (a form of binary regression) [45].

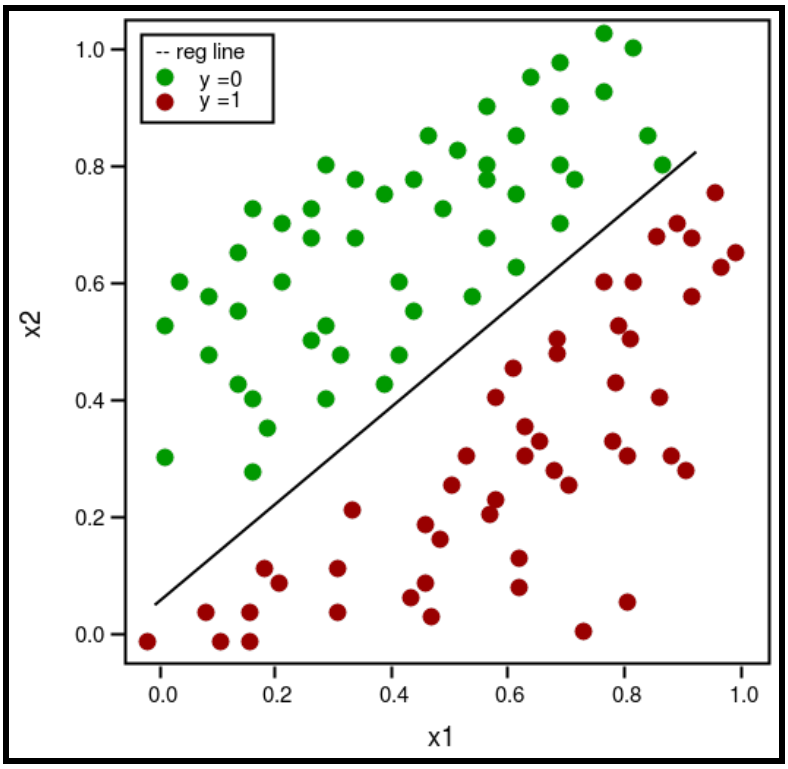


Figure 3.19: Logistic regression

3.5.9 Training Set

The training dataset is the set of data used to train the model. During each epoch, the model will be trained over and over again on the same data in our training set, and it will continue to learn about the features of this data fig. 3.20.

3.5.10 Validation Set

The validation set is a set of data separate from the training set used to validate the model during training. This validation process helps give information that may assist in adjusting hyperparameters. It is exceptionally helpful to see whether the model is overfitting or underfitting or doing just fine fig. 3.20.

3.5.11 Testing Set

The test set is a set of data that is used to test the model after the model has already been trained. This test set is separate from both the test set and the validation set fig. 3.20.

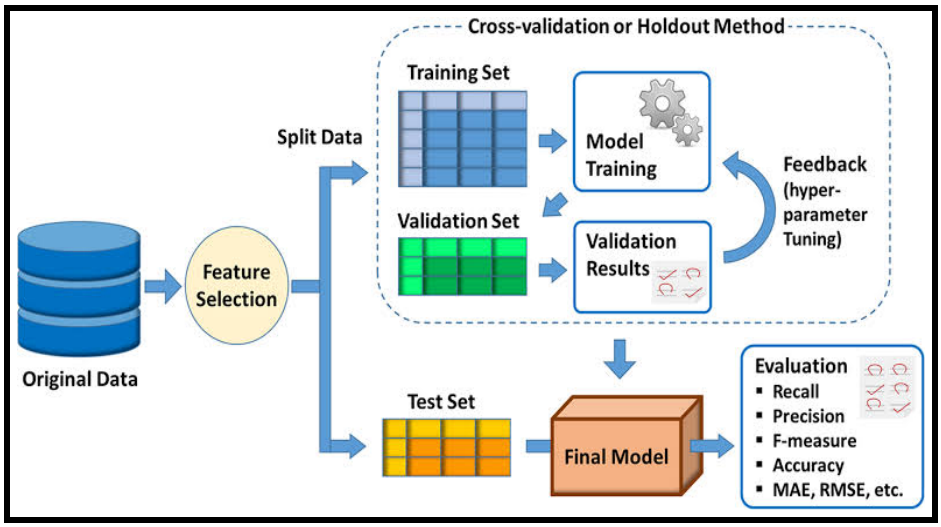


Figure 3.20: Training, Validation and Testing Sets

3.6 Evaluation Metric

Evaluation metrics are used to measure the quality of the statistical or Machine learning model. After extracting the appropriate feature, the last step is to classify the attained data and assign it to a specific class. There are many different types of evaluation metrics available to test a model [2]. These include classification accuracy, F1 Score, Recall, Precision, confusion matrix, etc., which defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.4}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.5}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.6}$$

$$F1 = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{3.7}$$

Where, TP : True Positive, FP : False Positive, TN : True Negative, and FN : False Negative

3.7 Transfer Learning

Transfer learning is a technique in deep learning by which A neural network model is first trained on a problem similar to the one being solved. Moreover, one or more layers from the trained model are used in a new model trained on the problem of interest. Transfer learning has the benefit of reducing the training time for a neural network model and resulting in lower generalization error [46]. Here, several pre-trained models that are quite effective and renowned such as VGG16, AlexNet, GoogleNet, etc. However, there are two main approaches to implementing transfer learning. They are:

- **Weight Initialization**

  Weight initialization aims to prevent layer activation outputs from exploding or vanishing during a forward pass through a deep neural network. Moreover, the weights in re-used layers may be used as the starting point for the training process and adapted to the new problem [47].

- **Feature Extraction**

  Feature extraction is such a process of dimensionality reduction by which an initial set of input data is reduced to more flexible groups for processing. It enables the deep learning algorithm to train faster. It reduces the complexity of a model and makes it easier to interpret [48].

3.7.1 Pretrained Architectures

Deep learning architectures is highly used for the diagnosis of the diseases since 2016. The most investigated DL techniques are Xception, DenseNet121, Resnet50, Inception V3, Inception-ResNet-v2, and VGG16.

3.7.1.1 Xception

Xception is a convolutional neural network that is 71 layers deep. It is an improved version of Inception architecture and involves depthwise separable convolutions. More precisely,

Xception replaces the standard Inception modules with depthwise separable convolutions. It showed good results compared to VGG16, Resnet, and Inception in classical classification problems. Xception has an image input size of 299x299. [3].

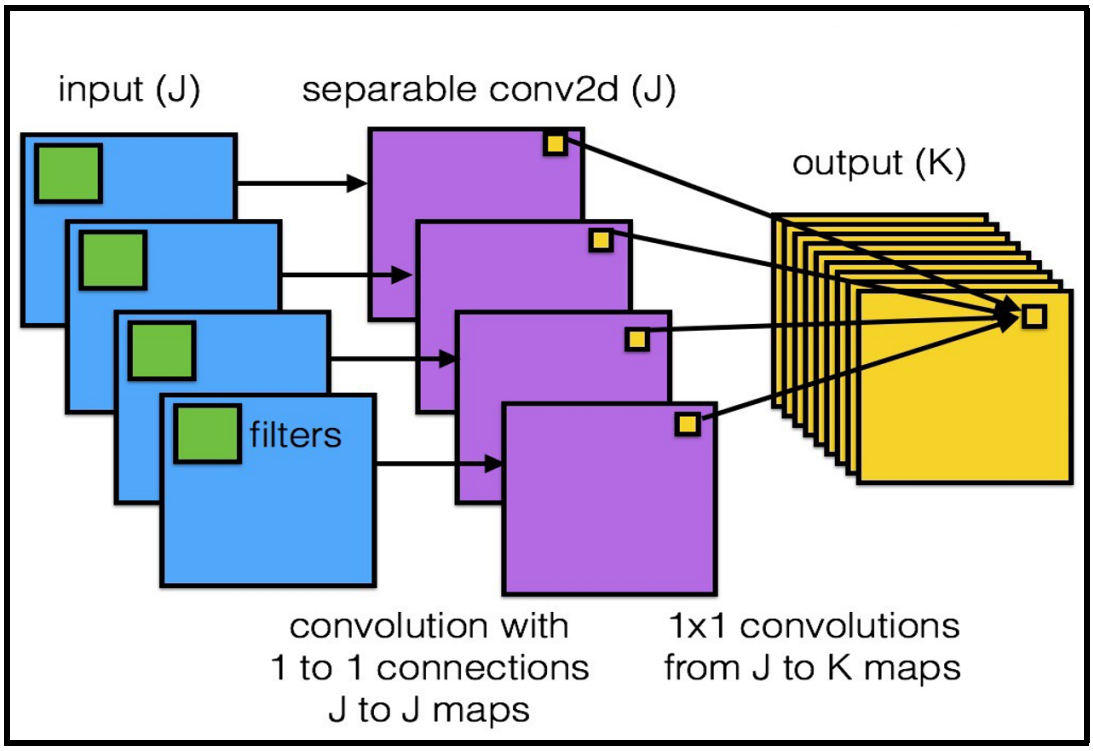


Figure 3.21: Xception

3.7.1.2 DenseNet121

DenseNet121 (Dense Convolutional Network) is a convolutional neural network that is 121 layers deep and accepts an image input size of 224 224. DenseNet121 is an improvement of ResNet that includes dense connections among layers. It connects each layer to every other layer in a feed-forward fashion. Unlike traditional convolutional networks with L layers that have L connections, DensNet121 has L(L+1)/ 2 direct connections. Indeed, unlike conventional networks, DenseNet can improve performance by increasing the computation requirement, reducing the number of parameters, encouraging feature reuse, and reinforcing feature propagation. DenseNet architecture requires fewer parameters than a traditional CNN. Another problem with DenseNet is the training time because every layer has its input from previous layers. However, DenseNet solves this issue by giving access to the gradient values form the loss function and the input image. It significantly reduces the computation cost and makes this model a better choice [3].

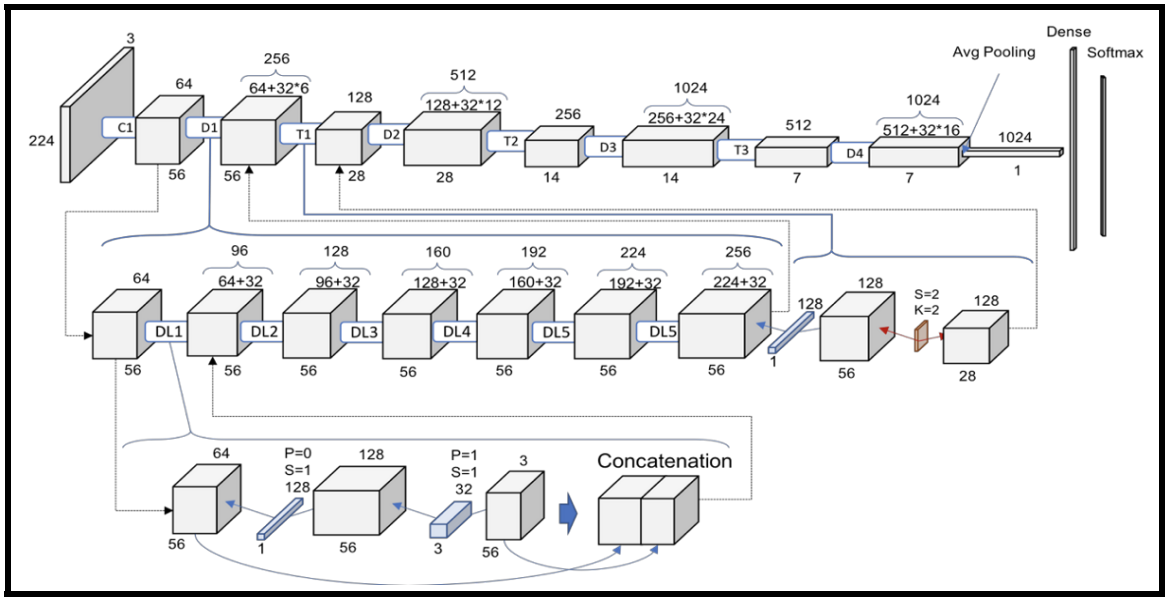


Figure 3.22: DenseNet121

3.7.1.3 Inception-ResNet-v2

Inception-ResNet-v2 is a convolutional neural network trained on more than a million images from the ImageNet database. It is a hybrid technique combining the inception structure and the residual connection. The model accepts images of 299299 images, and its output is a list of estimated class probabilities. The advantages of Inception ResnetV2 are converting inception modules to Residual Inception blocks, adding more Inception modules, and adding a new type of Inception module (Inception-A) after the Stem module. [3].

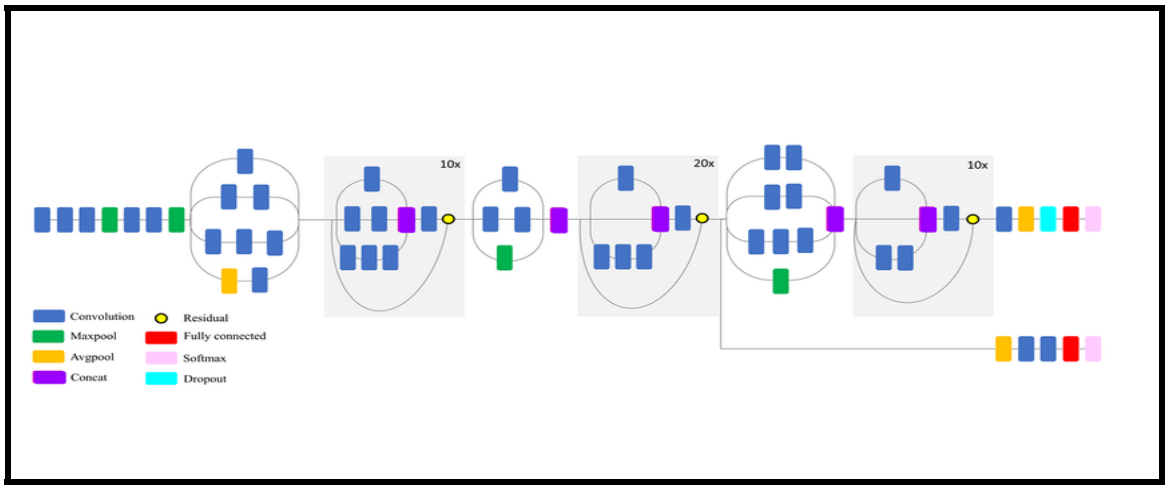


Figure 3.23: Inception-ResNet-v2

3.7.1.4 ResNet50

Resnet50 is a deep residual network developed by and is a subclass of convolutional neural networks used for image classification. The major innovation is the introduction of the new architecture network-in-network using extra layers. The Renset50 consists of five steps, each with a convolution and Identity block, and each convolution block has three convolution layers, and each identity block has three convolution layers. Resnet50 has 50 residual networks and accepts images of 224 224 pixels. The ResNet model comes with a residual learning framework to simplify the training of deeper networks. The architecture is based on the reformulation of network layers as learning residual functions to the layer inputs. The depth of the remaining network is eight times deeper than VGG nets, but its complexity is lower [3].

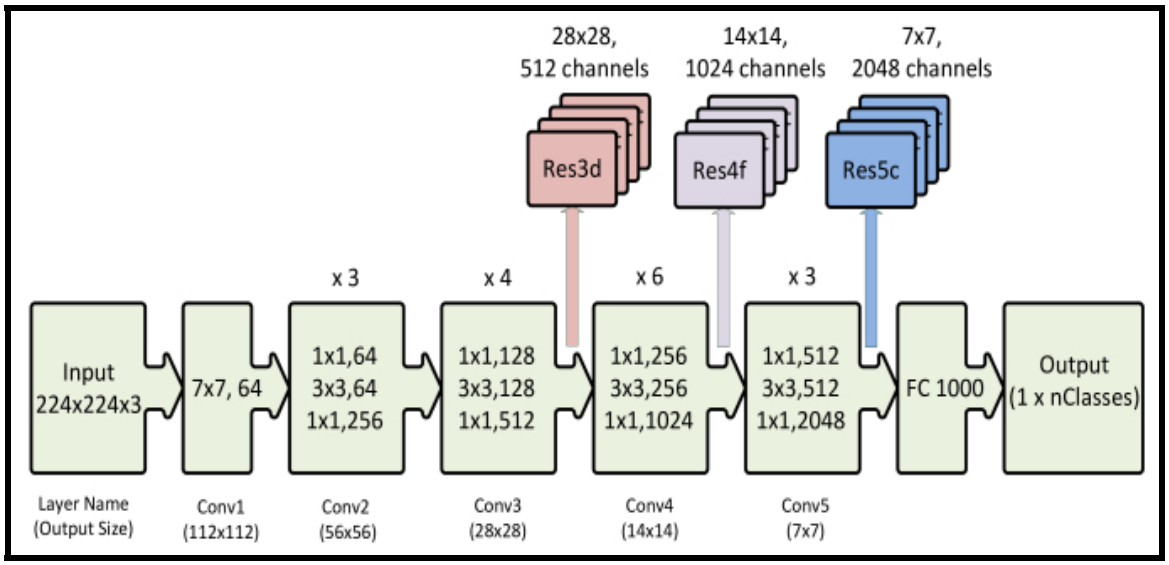


Figure 3.24: ResNet50

3.7.1.5 Inception V3

The Inception V3 model allows for increasing the depth and width of the deep learning network but maintaining the computational cost constant at the same time. This model was trained on the original ImageNet dataset with over 1 million training images. It works as a multi-level feature generator by computing 1*1, 3*3 and 5*5 convolutions. It allows the model to use all kinds of kernels on the image and get the results from all of those. All such outputs are stacked along the channel dimension and used as input to the next layer. This model achieved top performance for computer vision tasks by using some advanced techniques [3].

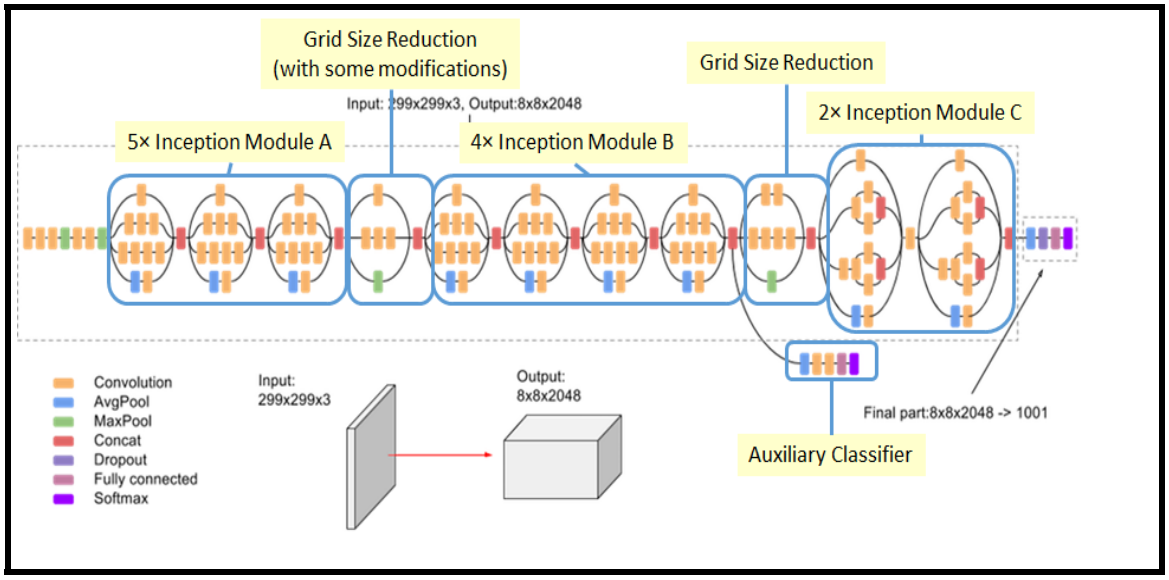


Figure 3.25: Inception V3

3.7.1.6 VGG16

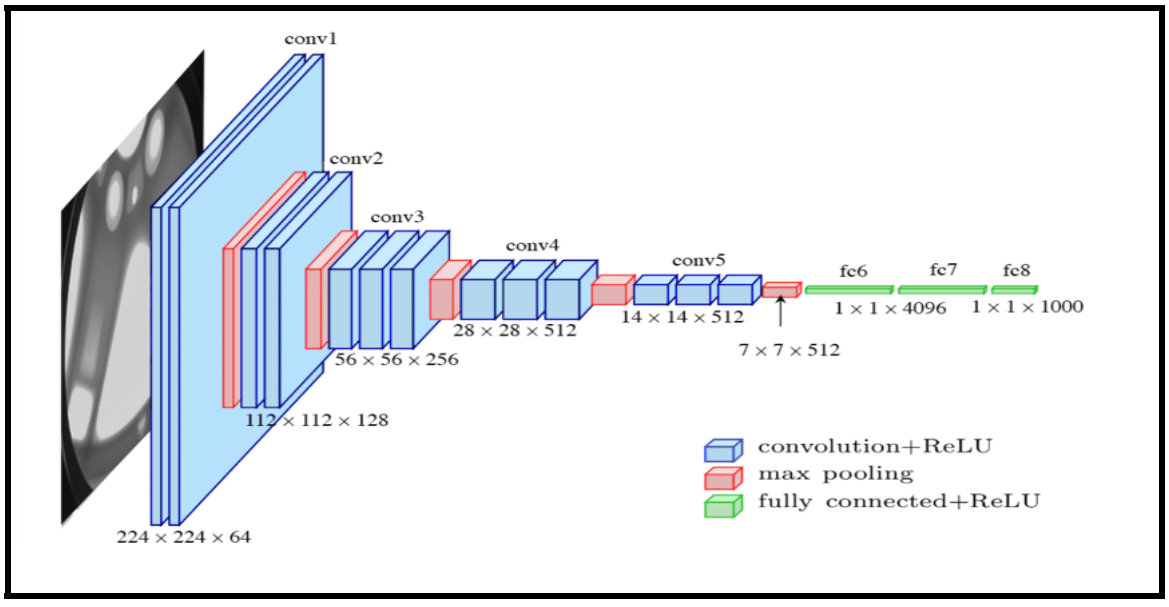


Figure 3.26: VGG16

VGG is a convolution neural net (CNN) architecture. The principal characteristic of this architecture is instead of having a large number of hyperparameters. They concentrated on simple 33 size kernels in convolutional layers and 22 size in max-pooling layers. In the end, it has 2 FC (Fully Connected layers) trailed by a softmax for output. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes, are then followed by a softmax classifier. The most familiar VGG models are VGG16 and VGG19, which include 16 and 19 layers, respectively. The difference between VGG-16 and VGG-19 is that VGG-19 has one more layer in each of the three convolutional blocks [3].

Chapter 4

Proposal

4.1 Motivation

Respiratory diseases have become the foremost cause of death and disability in the world. About 65 million people suffer from the chronic obstructive pulmonary disease (COPD), and 3 million people die from it, where around 2 million pneumonia-related deaths are reported every year. The corona epidemic (COVID-19) has spread around the world, with the number of victims, and the death rate is steadily rising. One of the significant symptoms of this terrible infectious disease is shortness of breath and severe pneumonia, which is one of the vital diseases of the chest. According to the statistics, it is explicit that how terrible chest diseases are for our human civilization. To detect these chest diseases, many researchers have proposed some renowned methods, one of which is X-ray. However, detecting different chest diseases from X-rays is indeed very time-consuming. If there is no emergency case, the results are usually ready within 1 or 2 days. Therefore, the radiologist sends the report to the physician who will discuss the results and explain what they mean. But when the epidemic becomes terrible, very difficult to give results in a short time by testing thousands of people who are being infected at every moment. Moreover, the hospital does not have enough health workers to examine so many people. Thus, an automated system is immensely needed to prevent this issue that could help at the early detection and classification of chest diseases. Because early detection of chest diseases can save many lives. Indeed, Computer-aided systems are now of great interest to reduce the number of casualties by detecting it in the early stages and helping out the radiologists in making important decisions. To build this automated system, we will first find out if there is any disease. If found, then we will also classify the disease using Convolutional Neural Network (CNN) and Transfer Learning with improved accuracy that reduces the workload of human experts.

4.2 Proposed Method

Our main objective is to reimplement an efficient model that can correctly classify any diseases like Pneumonia, Brain Tumor, heart diseases, etc. To facilitate the work, we have divided the main task of the project into four subtasks. In this case, the subtasks are described step by step in the following sections. These are -

- Dataset Preprocess
- Trained with Convolutional Neural Network (CNN)
- Trained with Transfer Learning
- Comparison of the results between CNN and Transfer Learning

4.2.1 Dataset Preprocess

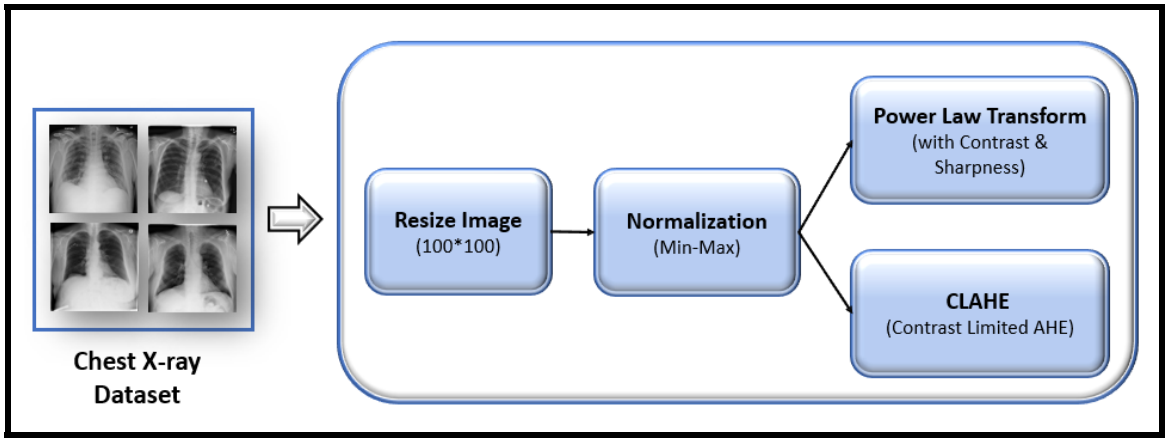


Figure 4.1: Dataset Preprocess

First of all, we have used two datasets for this research. One is a **ChestX-ray14 database**, and the other is **Chest X-Ray Image (CXI) dataset**, which contains 112,120 and 5856 images with resolution 1024*1024. But from these datasets, we will work with four diseases, including no finding.

In that case, we have resized the chest x-ray images into 100*100 for reducing computational cost and normalize those images by using **min-max normalization** to make every image have the same scale. Then we have applied **CLAHE (Contrast Limited Adaptive Histogram Equalization)** to the X-ray image before training to redistribute the lightness values of the image and **Power Law (Gamma) transformation** with contrast and sharpness to x-ray images before the training so that the images maintain a balance of intensity and details.

4.3 Trained with Convolutional Neural Network (CNN)

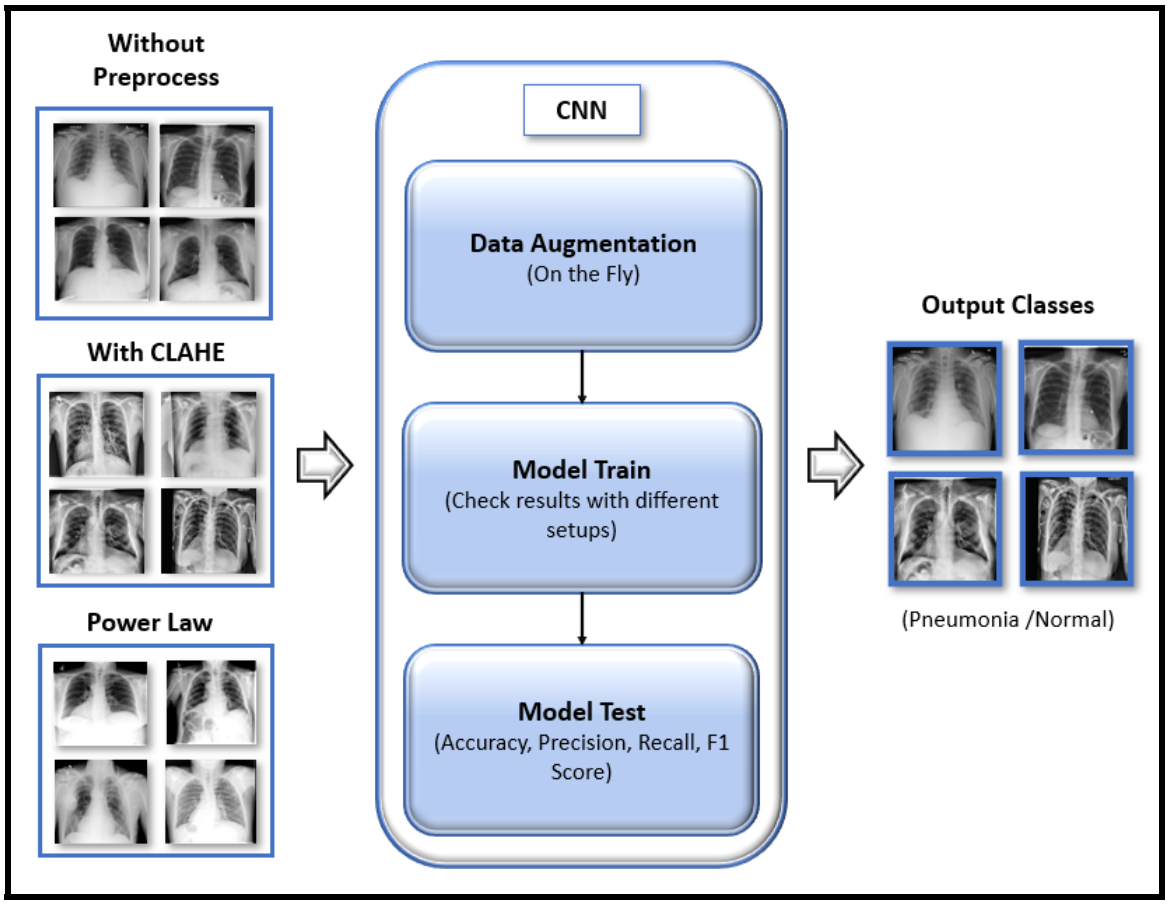


Figure 4.2: Trained with Convolutional Neural Network

Moreover, we have trained our dataset in three ways by using the Convolutional Neural Network. The first way is to train CNN without preprocess on the dataset. The second is to train CNN with CLAHE and the last one is to train after applying Power law transform. We have trained the CNN model by augmenting the data using on the fly method in three conversions of the dataset and evaluate the results with different setups. Then, we have classified the output classes, either normal or different diseases by testing our model successfully. Here we have used CNN in this study for evaluation purposes and observed which approach (with preprocessing or without preprocessing) will be beneficial for getting better accuracy.

4.4 Trained with Transfer Learning

Similarly, we have trained transfer learning such as **Xception, DenseNet121, Resnet50, Inception V3, Inception-ResNet-v2, and VGG16** without and with preprocessing (CLAHE

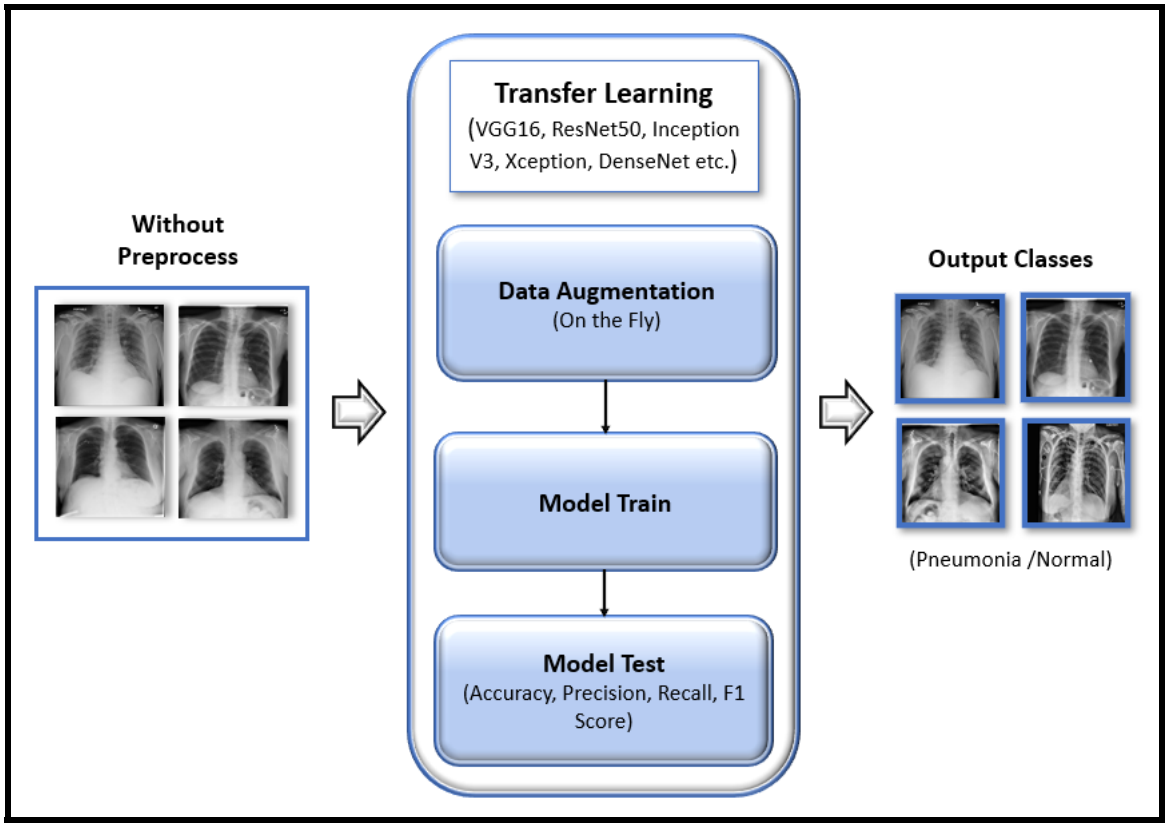


Figure 4.3: Trained with Transfer Learning

Power-law transformation). To avoid the risk of overfitting, we have used data augmentation using on the fly approach and evaluate the results by testing these architectures successfully. Here is the remarkable thing that the input size of the pre-trained models will change according to the above architectures.Then, we will compare each of the pre-trained models to analyze which one gives better results.

4.5 Comparison of the results between CNN and Transfer Learning

Finally, we have compared the best results of transfer learning and CNN by using the evaluation metric to explore which one ultimately gives satisfactory results by classifying chest X-ray diseases with no findings if there are no existing diseases. The full working procedure of our project is given below fig. 4.4-

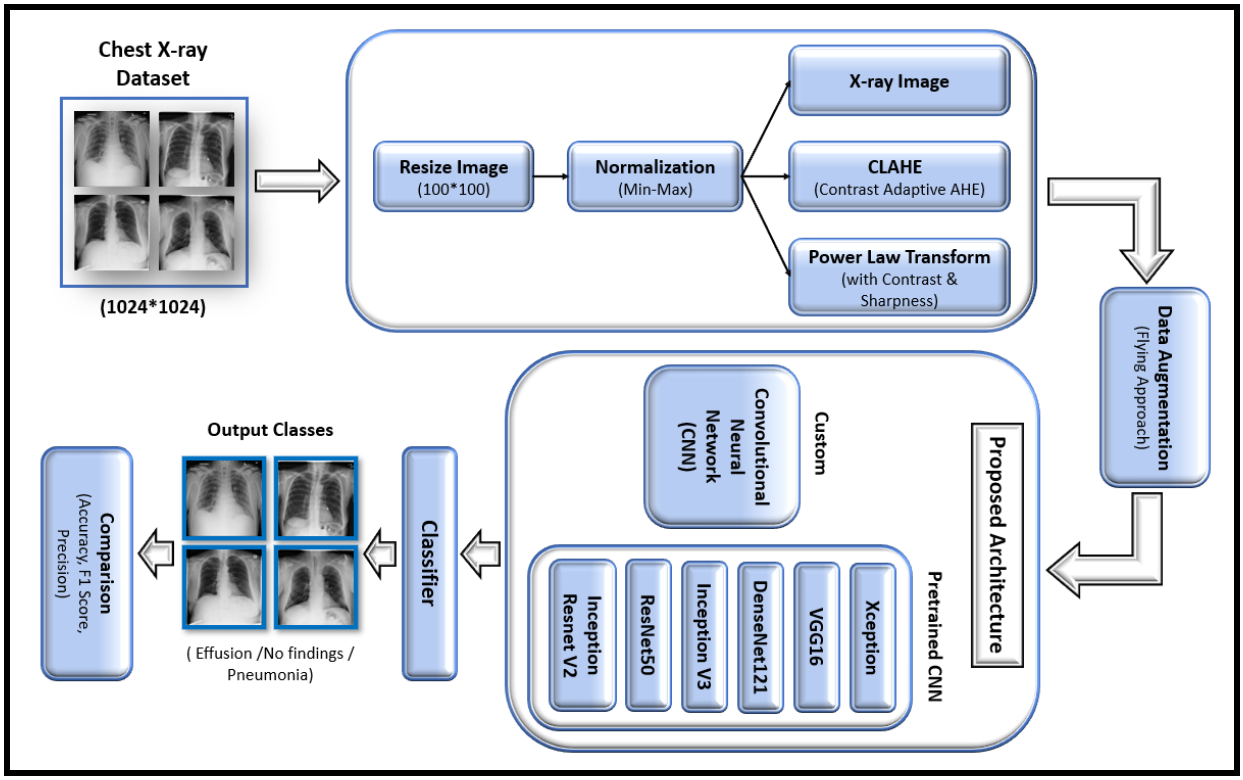


Figure 4.4: Flowchart of our system

Chapter 5

Implementation

5.1 Overview

Our goal is to detect chest diseases like- Pneumonia, Effusion, and no findings from X-ray images using Custom CNN and different architectures of Transfer Learning. So, If the system detects the chest diseases, then it will also classify the diseases. So far, several papers have been published, using various methods to identify chest X-ray diseases. In this case, we have done the ablation experiments by using CNN and compare the results between three datasets in training evaluation. Furthermore, CNN has gotten good results without preprocessing the dataset. Then, we have tested our best setup in CNN and pre-trained architecture. In this study, the images were resized to 100 * 100 and data augmentation was used to avoid overfitting problems, and the rest of the hyperparameters were perfectly tuned using different setups and datasets. We have used python 3.6.8 and Keras 2.2.4 to implement this system.

5.2 Experimental Setup

All of these experiments were performed on a computer with Windows 10 based system with Intel® Core (TM) i7-6700HQ CPU @2.60GHz processor, 1 TB HDD, 500 GB NVMe M.2 SSD, 8 GB RAM with 2133 MHz, a CUDA-enabled Nvidia® GTX 960M 4GB graphical processing unit (GPU), Python® 3.7.6, Keras® 2.2.4-tf with TensorFlow® 2.1.0 backend, and CUDA compilation tools, release 10.2, cuDNN 7.6.0 dependencies for GPU acceleration.

5.3 Collecting Dataset

Analysis of Chest X-Ray is one of the challenging tasks in medical science. There are thousands of datasets available for chest X-Ray, but all the datasets are limited to a few thousands of images. For this study, we worked with two datasets which are described below.

5.3.1 Chest X-ray 14 Dataset

Neural networks generally need numerous sample images for the training purpose. The dataset used for this research is the Chest X-ray 14 database. This dataset is obtained from the online database formulated by the U.S. National Institute of Health Clinical Center, PACS. This database has 60% scans from all frontal chest X-ray scans done within the hospitals. It consists of 112,120 images of frontal X-Ray scans collected from 30,805 patients with fourteen different diseases and obtained through natural language processing of their associated radio-logical reports. There are fourteen common thoracic pathologies found in the chest during X-ray scans with resolution 1024*1024 [11].

5.3.2 Chest X-Ray Image (CXI) Dataset

This present work introduces a publicly available image dataset which contains X-Ray and computed tomography (CT) images. This dataset, named chest X-Ray CT dataset, which is composed of 5856 images (jpeg format) and has two categories (4273 pneumonia and 1583 normal). This dataset is developed in the Guangzhou Women and Children's Medical Center, Guangzhou, China. The pre-processing is performed on all the images within the dataset to remove low-quality scans. Images are checked and classified by two specialist clinicians and further by a third party radiologist to avoid any misclassification [49]. The dataset is further subdivided into three folders: training, validation, and test sets, and each folder contain images from both categories: pneumonia and normal.

In this study, we will work on three classes or diseases like Pneumonia, Effusion and No findings. Note that our first dataset has some pneumonia disease which is not enough for training purpose. So we worked on combining the two datasets to reduce this issue. We split our dataset into training, validation and testing sets having the ratio of 70:20:10 respectively which are given below.

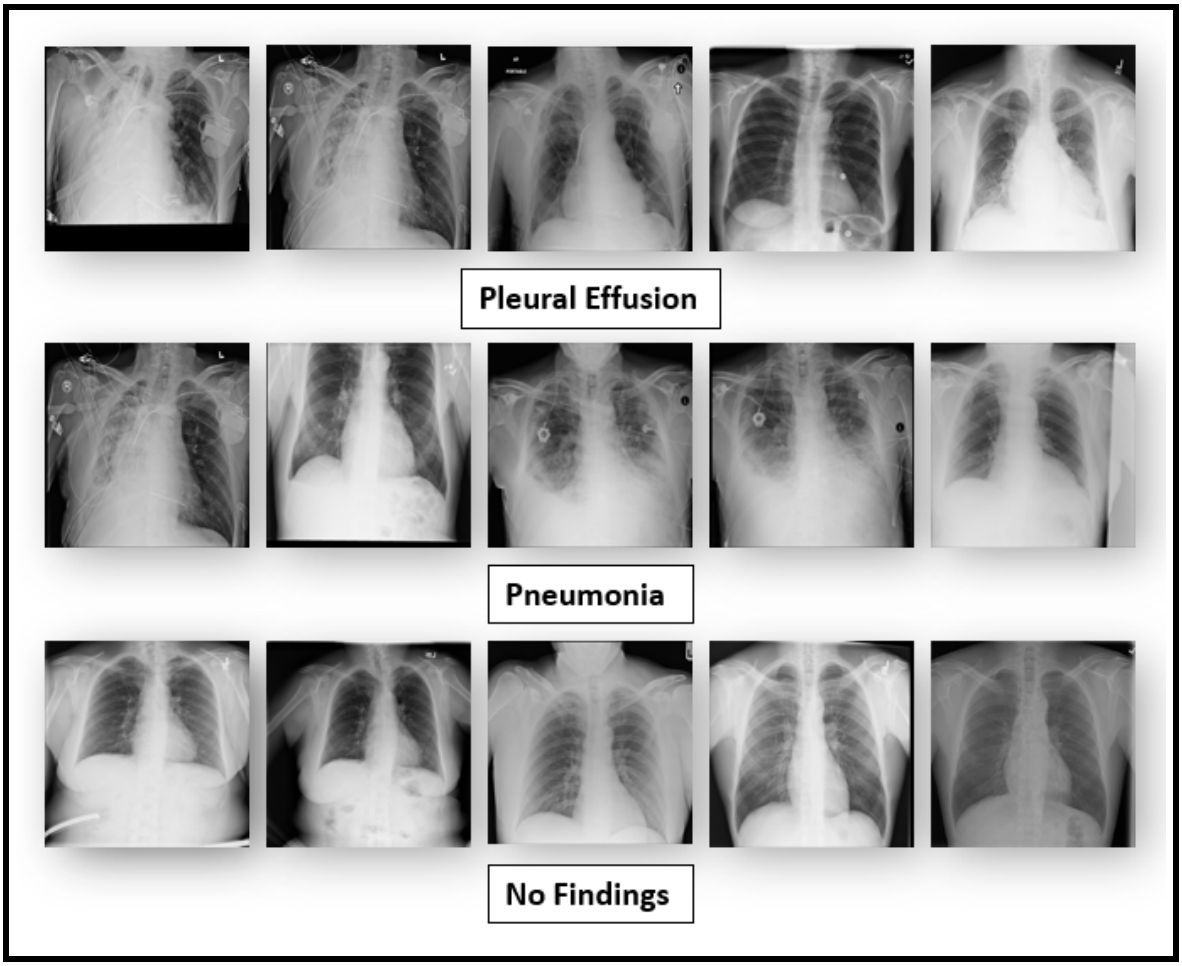


Figure 5.1: Datasets

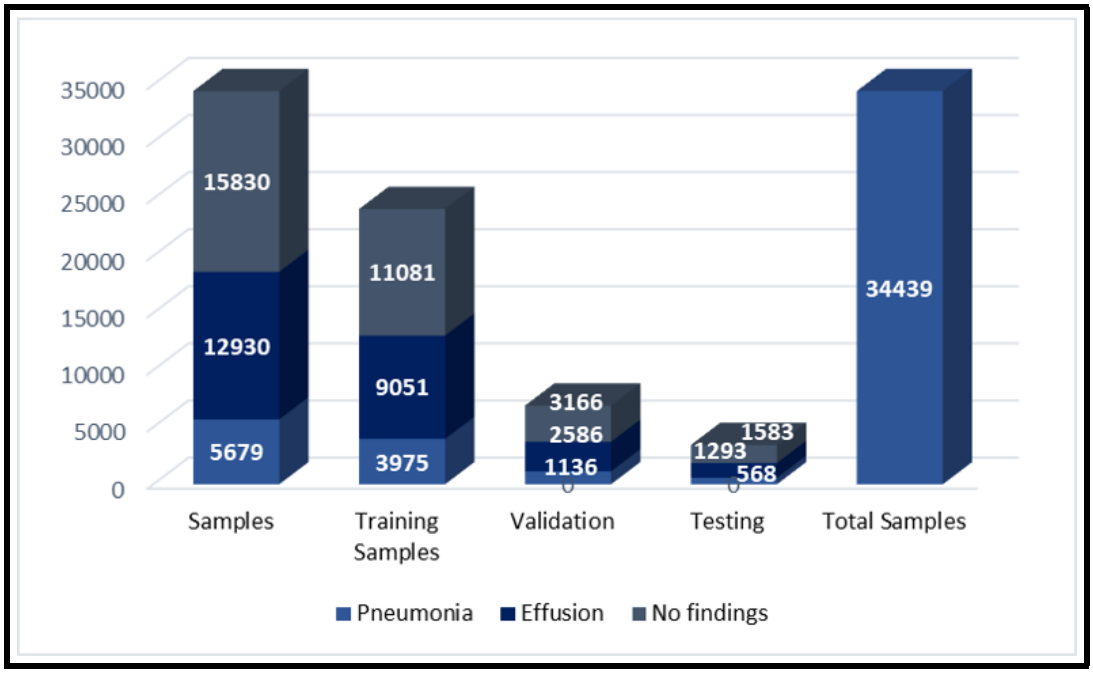


Figure 5.2: Statistics of the Dataset

Table 5.1: Dataset partition and their Characteristics

Category	Samples	Training Samples	Validation Samples	Testing Samples	Type	Depth	Total Samples
Pneumonia	5679	3975	1136	568	PNG	8-bit	34439
Effusion	12930	9051	2586	1293	PNG	8-bit	
No findings	15830	11081	3166	1583	PNG	8-bit	

5.4 Data Splitting

The datasets are split into three sets which are training, validation and testing sets and having the ratio of 70:20:10 respectively.(why should we use data splitting in deep learning)

5.5 Image preprocessing

Image Pre-processing is a common name for operations with images at the lowest level of abstraction, where both input and output are intensity images. The aim of image pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features essential for further processing. However, geometric transformations of images like rotation, scaling, translation are classified among pre-processing. In this section, several pre-processing techniques are discussed in detail, which has been experimented with, and the outcomes are reported.

5.5.1 Normalization

A normalization is an approach that is applied during data preparation to change the values of numeric columns in a dataset to use a standard scale when the features in the data have different ranges. It turns the range of pixel intensity values of the images. It also called contrast stretching or histogram stretching. Several normalization techniques are available like Decimal Scaling, Min-Max Normalization. We have applied Min-Max normalization on our dataset, where the minimum value of that feature is transformed into a 0. The maximum value is transformed into a 1, and every other value is converted into a decimal between 0 and 1. The following equation achieves this rescaling (Min-Max normalization).

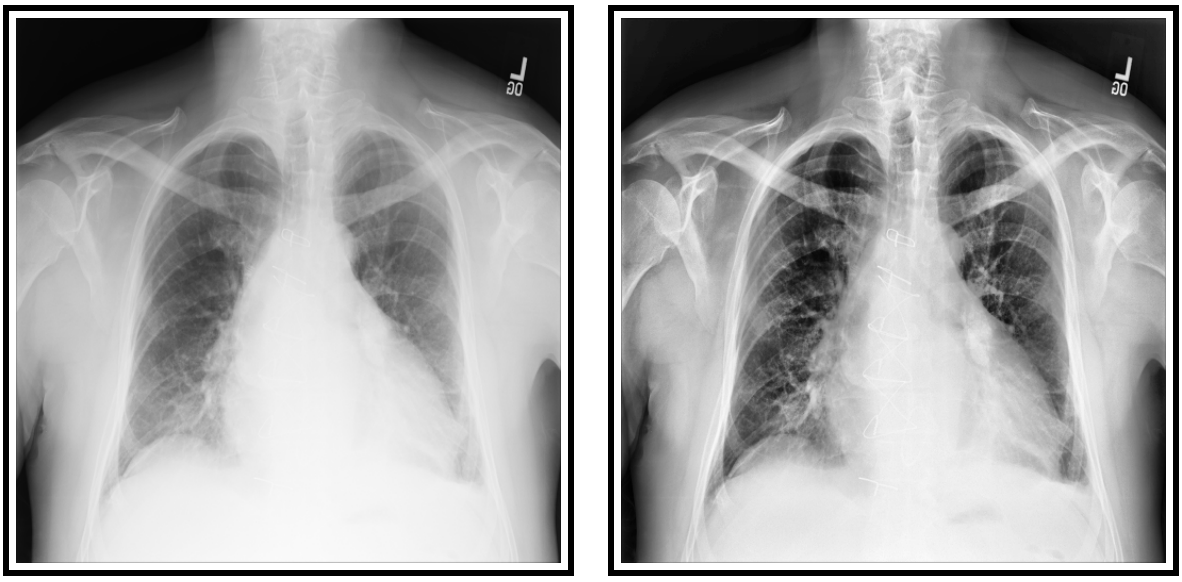
$$X = \frac{x - x_{min}}{x_{max} - x_{min}} = \frac{x}{255} \tag{5.1}$$

5.5.2 Resize

Preprocess Images for Deep Learning. The images must match the network's input size to train a network and make predictions on new data. If the size of the images is larger than required, then the images need to be cropped or resized for reducing computation cost while training. Moreover, the resolution of the images in our dataset is 1024 * 1024. We have reduced the size of the images to 100 * 100 for the convenience of custom model training to reduce the time complexity.

5.5.3 Contrast Limited Adaptive Histogram Equalization (CLAHE)

Adaptive histogram equalization (AHE) is a computer image processing technique used to improve contrast in images. It prevents the over-amplification of noise, which results in the AHE technique. CLAHE uses a contrast amplification limiting procedure for each neighboring pixel, which forms a transformation function to reduce the noise problem. Here, the block size is 8*8, and the clip-limit (contrast limit) is 3.0 for localized changes in contrast. Moreover, a clip-limit of 2 to 3 is a good starting place.

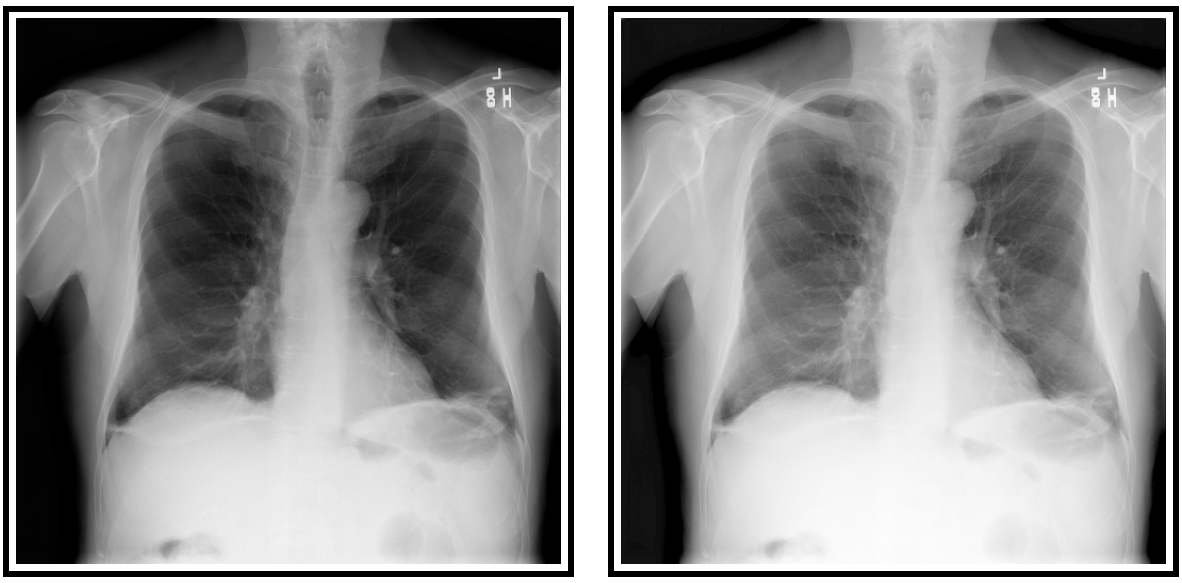


(a) Input Image (b) Output Image

Figure 5.3: CLAHE

5.5.4 Power Law Transform

The power-law is a suitable multi-purpose function for contrast manipulation in the spatial domain. Usually, this transformation is used to correct the imperfect intensity levels of an image. The gamma of different display devices is different. By changing the gamma γ value, the mapping nature of the input to output intensities also changes. Variation in the value of varies the enhancement of the images. Let rmax be the dominant peak in the histogram. Since our images are dark (rmax lies in the range 0-100), we set the gamma value to 0.6 as usual, choosing $\gamma < 1$ leads to contrast stretching.

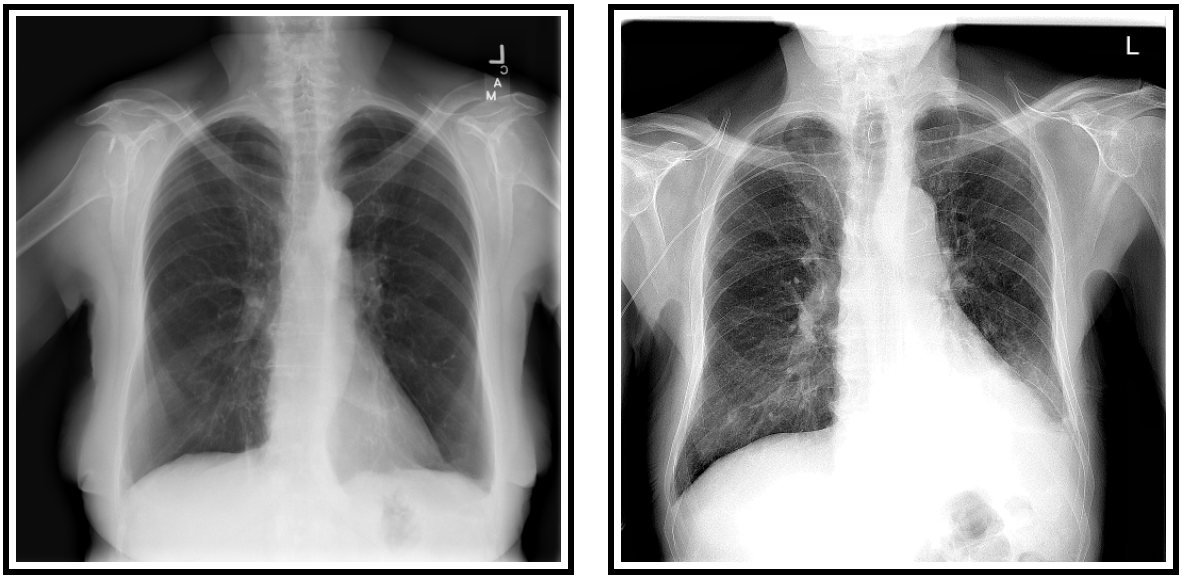


(a) Input Image (b) Output Image

Figure 5.4: Power Law Transform

5.5.5 Sharpness & Contrast

Image sharpening refers to any enhancement technique that highlights edges and fine details in an image. Acutance explains how quickly image information transitions at an edge, and so high acutance results in sharp changes and detail with clearly defined borders. Here, we set a sharpness factor to 8.0. Contrast stretching is a simple image enhancement technique that attempts to improve the contrast in an image by 'stretching' the range of intensity values. Here, we set a contrast factor to 1.4 on our X-ray images which are given below.



(a) Input Image (b) Output Image

Figure 5.5: Sharpness & Contrast

5.5.6 Data Augmentation

Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. It can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class. There are several techniques, such as cropping, padding, and horizontal flipping are commonly used to train large neural networks. There are three types of data augmentation one of them is in-place data augmentation or on-the-fly data augmentation. This type of data augmentation is what Keras ImageDataGenerator class implements. We used this kinda augmentation to ensure that our network, when trained, sees new variations of our data at every epoch and it is called “on-the-fly” data augmentation because this augmentation is done at training time. So, when our model is being trained, ImageDataGenerator class as “intercepting” the original data, randomly transforming it, and then returning it to the neural network for training, all the while the Neural Network has no idea the data was modified. Here is the description of Data augmentation parameters which we used in our system are given below.

Table 5.2: Data Augmentation Parameter details

Argument	**Parameter value**	**Description**
Rescale	1 / 255.0	Scale images from integers 0-255 to floats 0-1
Rotation range	90	Degree range of the random rotations
Vertical shift range	0.2	The parameter value of horizontal shifts (20%) is a fraction of the given dimension
Horizontal shift range	0.2	The parameter value of vertical shifts (20%) is a fraction of the given dimension
Shear range	0.2	Controls the angle in counter clockwise direction as radian in which our image will allowed to be sheared
Zoom range	0.2	Allows the image to be "zoomed out" or "zoomed in"
Horizontal flip	True	Controls when a given input is allowed to be flipped horizontally during the training process
Fill mode	Nearest	This is the default option where the closest pixel value chosen and repeated for all the empty values

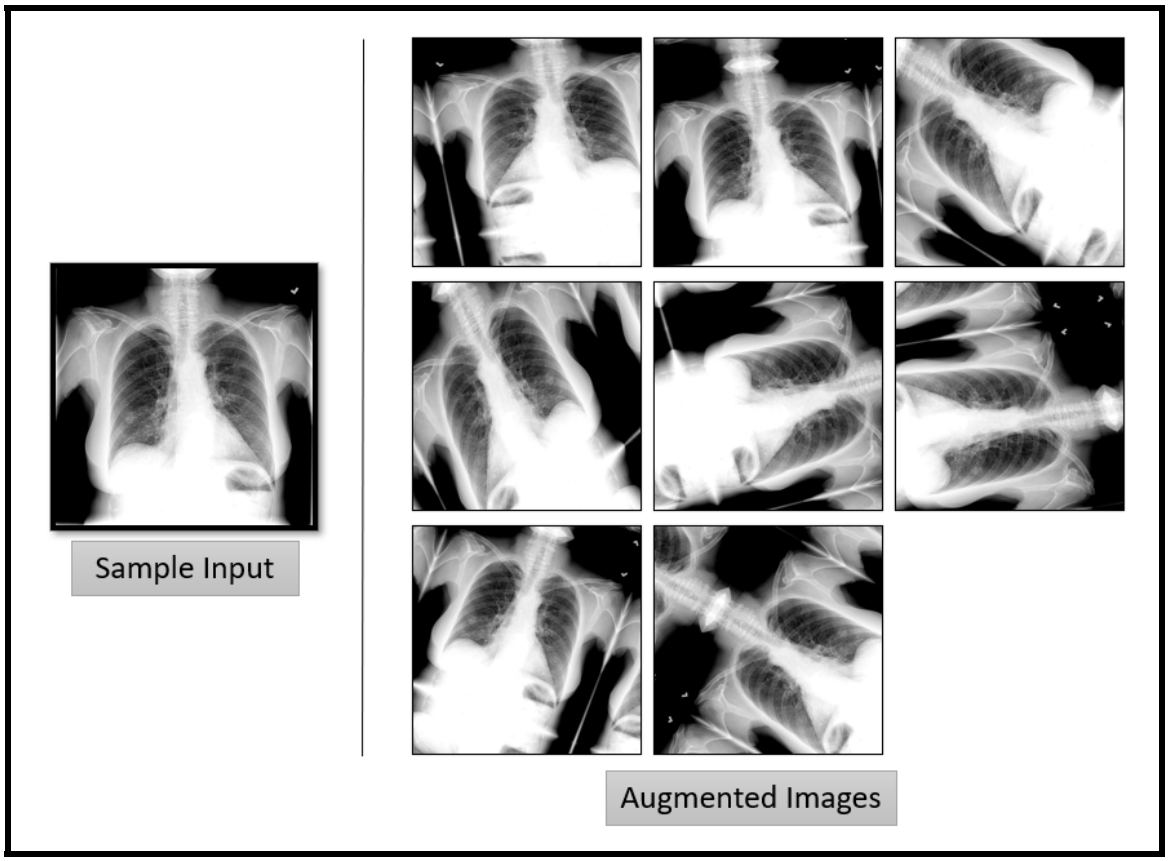


Figure 5.6: Data Augmentation

5.6 Proposed baseline CNN architecture

Generally, a CNN model consists of five layers: the input layer, convolutional layers, pooling layers, full-connection layers, and the output layer. Moreover, it is known that the CNN model can be trained end-to-end to allow the feature extraction and selection, finally, classification or prediction. How the network understands an image or process, the image is intricate, but it is known that features obtained in different layers of a network work better than human-built features.

The proposed baseline CNN for our experiment has the following architecture:

- **Input Layer**: In our experiment, the inputs are X-Ray images. The parameters are defining the image dimension (100 x 100).
- **Convolutional Layers**: A convolution is a linear operation that consists of the multiplication of a set of weights with the input. It's designed for two-dimensional input; the multiplication is performed between a two-dimensional array of weights and an array of input data. In the proposed architecture, we have three layers with a filter of size 3 x 3 and same paddings.

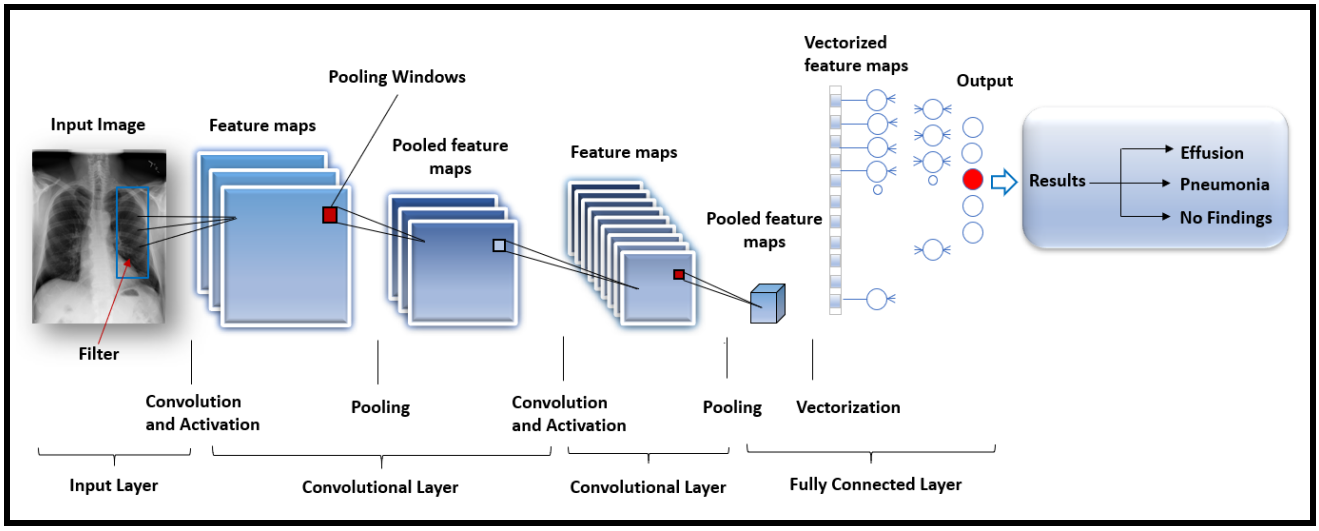


Figure 5.7: Baseline of CNN Architecture

- **Pooling Layers**: Pooling layers is a technique to downsample feature maps by summarizing the presence of features in patches of the feature map. There are two types of pooling methods that are average pooling and max pooling. In the proposed architecture, we used Max pooling to calculate the maximum value in each patch for every feature map. The max-pooling is set to 2 x 2 with a stride of 2.
- **ReLU Layers**: We have used 8 ReLU layer for each convolutional layer.

- **Fully Connected Layers**: It treats the input data as a simple vector and produce an output as a single vector. We have one inner-product layer in this model: the last one, a fully connected output layer with softmax activation.

5.7 Architecture and Hyperparameters of Custom Models

Here we have trained eleven setups of a custom model using three approaches. The approaches are model trained without preprocessing, model trained with CLAHE, and model trained on power-law transform with sharpness and contrast. Not all custom models have the same architecture. Furthermore, there is a difference in performance between the setups of custom models - 1, 2, 3, 4, and 6 due to the difference in hyperparameters. Again the Architectures are equal in between the setups of 5, 7, 8, 9, 10, and 11. So, their performance is pretty much the same. Since the custom model has been used for all setups; therefore, these typical Implemented Architectures are given below for ease of description.

5.7.1 Setup-1's Architecture and Values of Hyperparameters

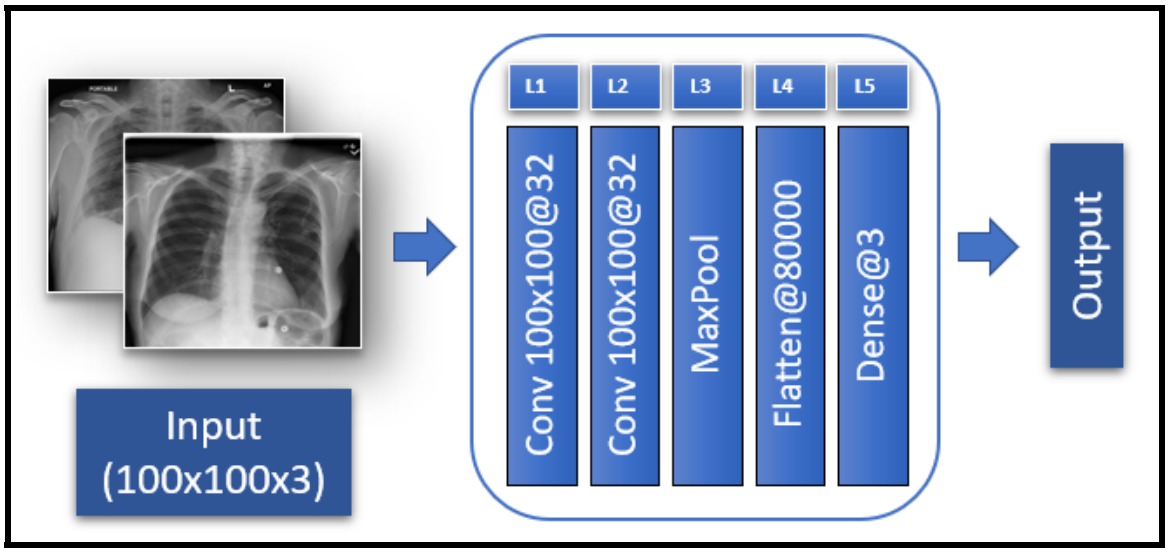


Figure 5.8: Architecture of Setup-1

Setup-1 is five layers custom architecture of Neural Network with 3-way softmax activation. The Setup-1 consists of two convolutional layers, one max pool layer, one flatten layer, and two dense layers. The input dimensions are 100x100 to reduce the computational cost, and ReLU is used as an activation function for convolution layers. The filter (kernel) size is set to 3x3 which is used to extract the features from the images. A pool size of 2x2 is used in max-pooling layers. 'Adam' optimizer is used with a learning rate of 0.001. Here, the batch size is 32 and as for loss function *'categorical-cross-entropy'* is used.

Table 5.3: Hyperparamerter value's of Setup-1

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.001
Activation Function	ReLu
Kernel Size	3x3
Layer	5
Loss Function	categorical_crossentropy
MaxPooling	2x2

5.7.2 Setup-2's Architecture and Values of Hyperparameters

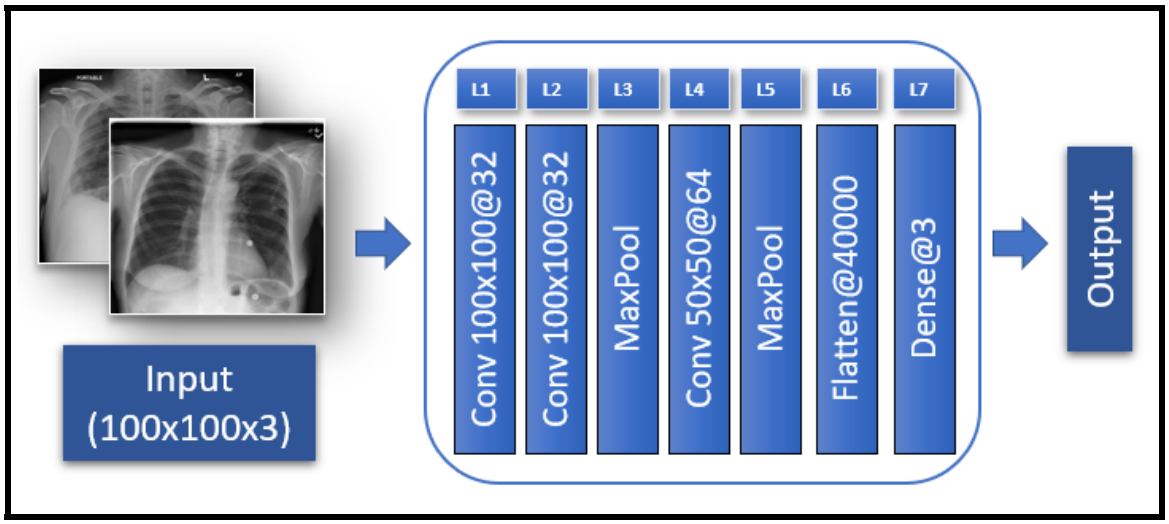


Figure 5.9: Architecture of Setup-2

Setup-2 is seven layers custom architecture of Neural Network with 3-way softmax activation. The Setup-2 consists of three convolutional layers, two max pool layers, one flatten layer, and one dense layer. The input dimensions are 100x100 to reduce the computational cost, and ReLU is used as an activation function for convolution layers. The filter (kernel) size is set to 3x3 which is used to extract the features from the images. A pool size of 2x2 is used in max-pooling layers. 'Adam' optimizer is used with a learning rate of 0.001. Here, the batch size is 32 and as for loss function *'categorical-cross-entropy'* is used.

Table 5.4: Hyperparamerter value's of Setup-2

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.001
Activation Function	ReLu
Kernel Size	3x3
Layer	7
Loss Function	categorical_crossentropy
MaxPooling	2x2

5.7.3 Setup-3's Architecture and Values of Hyperparameters

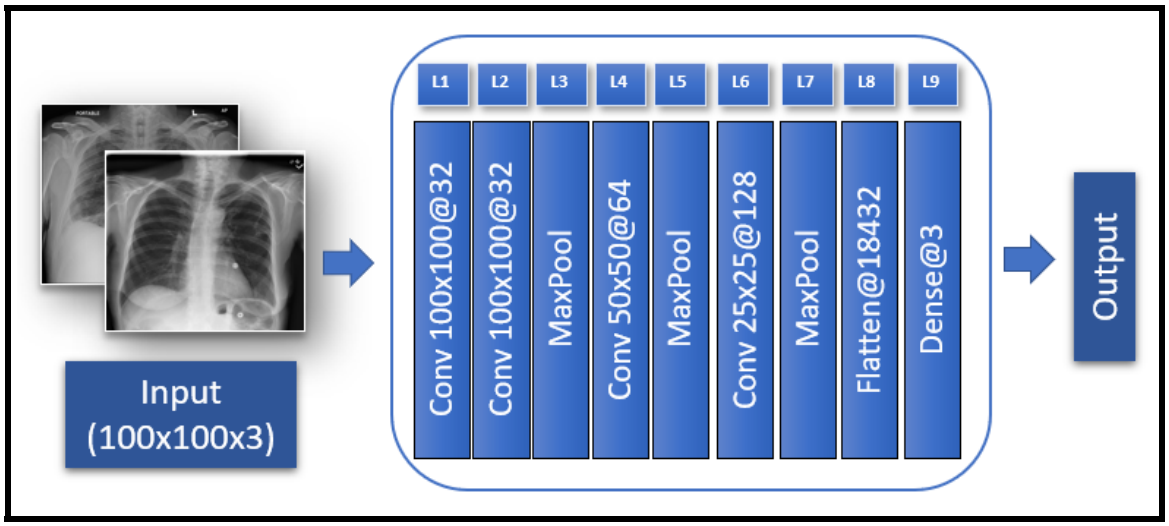


Figure 5.10: Architecture of Setup-3

Setup-3 is nine layers custom architecture of Neural Network with 3-way softmax activation. The Setup-3 consists of four convolutional layers, three max pool layers, one flatten layer, and one dense layer. The input dimensions are 100x100 to reduce the computational cost, and ReLU is used as an activation function for convolution layers. The filter (kernel) size is set to 3x3 which is used to extract the features from the images. A pool size of 2x2 is used in max-pooling layers. 'Adam' optimizer is used with a learning rate of 0.001. Here, the batch size is 32 and as for loss function *'categorical-cross-entropy'* is used.

Table 5.5: Hyperparamerter value's of Setup-3

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.001
Activation Function	ReLu
Kernel Size	3x3
Layer	9
Loss Function	categorical_crossentropy
MaxPooling	2x2

5.7.4 Setup-4's Architecture and Values of Hyperparameters

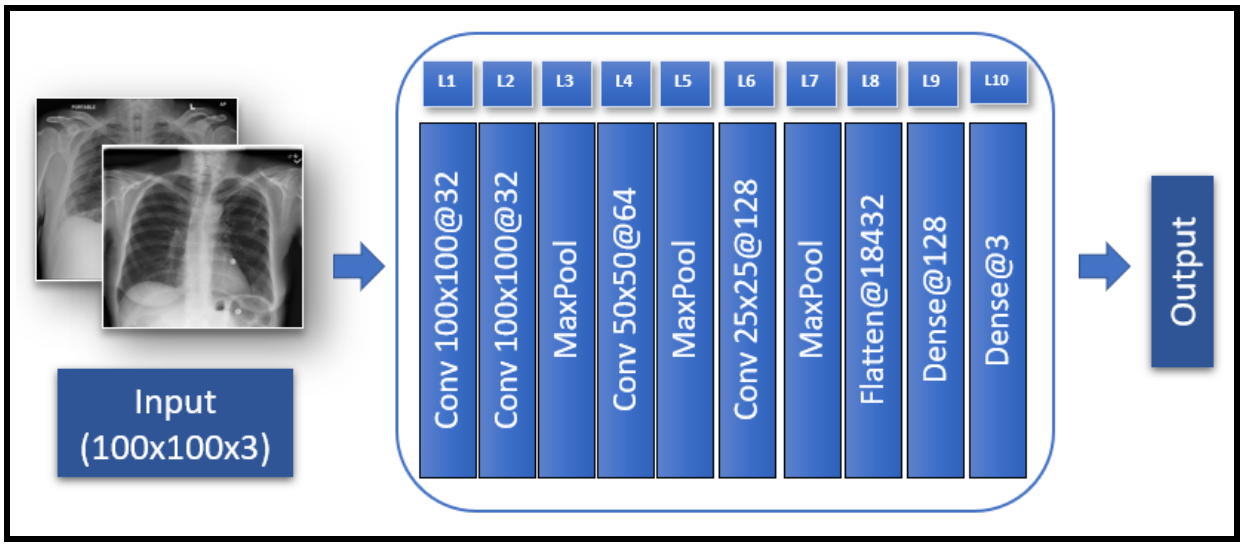


Figure 5.11: Architecture of Setup-4

Setup-4 is ten layers custom architecture of Neural Network with 3-way softmax activation. The Setup-4 consists of four convolutional layers, three max pool layers, one flatten layer, and two dense layers. The input dimensions are 100x100 to reduce the computational cost, and ReLU is used as an activation function for convolution layers. The filter (kernel) size is set to 3x3 which is used to extract the features from the images. A pool size of 2x2 is used in max-pooling layers. 'Adam' optimizer is used with a learning rate of 0.001. Here, the batch size is 32 and as for loss function *'categorical-cross-entropy'* is used.

Table 5.6: Hyperparamerter value's of Setup-4

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.001
Activation Function	ReLu
Kernel Size	3x3
Layer	10
Loss Function	categorical_crossentropy
MaxPooling	2x2

5.7.5 Setup-5,7,8,9,10 and 11's Architecture and their different Hyperparameter values

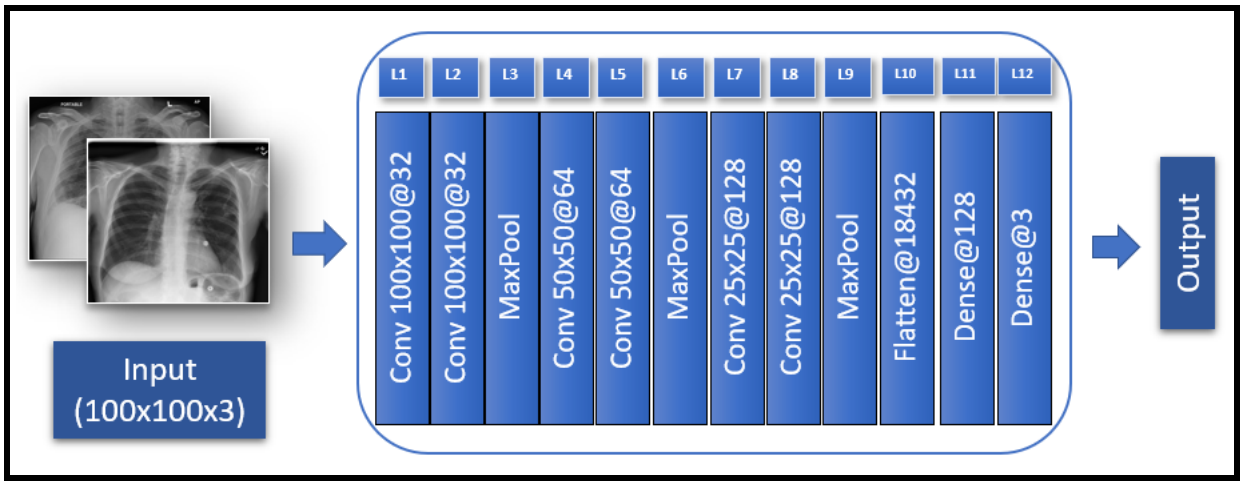


Figure 5.12: Architecture of Setup-5,7,8,9,10,11

Here, Setup-5,7,8,9,10 and 11's architectures are similar. It consists of twelve layers of the custom architecture of Neural Network with 3-way softmax activation with eight convolutional layers, four max pool layers, one flatten layer, and two dense layers. The input dimensions are 100x100 to reduce the computational cost. A pool size of 2x2 is used in max-pooling layers. Here, the batch size is 32 and as for loss function *'categorical-cross-entropy'* is used. Although the architecture is the same, the setups are different due to differences in some hyperparameters. In this case, the values of different hyperparameters for different setups are given in the table below.

5.7.5.1 Hyperparamerter value's of Setup-5

Table 5.7: Hyperparamerter value's of Setup-5

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.001
Activation Function	ReLu
Kernel Size	3x3
Layer	12
Loss Function	categorical_crossentropy
MaxPooling	2x2

5.7.5.2 Hyperparamerter value's of Setup-7

Table 5.8: Hyperparamerter value's of Setup-7

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.001
Activation Function	ReLu
Kernel Size	5x5
Layer	12
Loss Function	categorical_crossentropy
MaxPooling	2x2

5.7.5.3 Hyperparamerter value's of Setup-8

Table 5.9: Hyperparamerter value's of Setup-8

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.001
Activation Function	ReLu
Kernel Size	7x7
Layer	12
Loss Function	categorical_crossentropy
MaxPooling	2x2

5.7.5.4 Hyperparamerter value's of Setup-9

Table 5.10: Hyperparamerter value's of Setup-9

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.01
Activation Function	ReLu
Kernel Size	5x5
Layer	12
Loss Function	categorical_crossentropy
MaxPooling	2x2

5.7.5.5 Hyperparamerter value's of Setup-10

Table 5.11: Hyperparamerter value's of Setup-10

Parameter's Name	Values
Batch Size	32
Optimizer	SGD
Learning Rate	0.01
Activation Function	ReLu
Kernel Size	5x5
Layer	12
Loss Function	categorical_crossentropy
MaxPooling	2x2

5.7.5.6 Hyperparamerter value's of Setup-11

Table 5.12: Hyperparamerter value's of Setup-11

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.01
Activation Function	tanh
Kernel Size	5x5
Layer	12
Loss Function	categorical_crossentropy
MaxPooling	2x2

5.7.6 Setup-6's Architecture and Values of Hyperparameters

Setup-6 is fifteen layers custom architecture of Neural Network with 3-way softmax activation. The Setup-4 consists of eight convolutional layers, four max pool layers, one flatten layer, and two dense layers. The input dimensions are 100x100 to reduce the computational cost, and ReLU is used as an activation function for convolution layers. The filter (kernel) size is set to 3x3 which is used to extract the features from the images. A pool size of 2x2 is used in max-pooling layers. 'Adam' optimizer is used with a learning rate of 0.001. Here, the batch size is 32 and as for loss function *'categorical-cross-entropy'* is used.

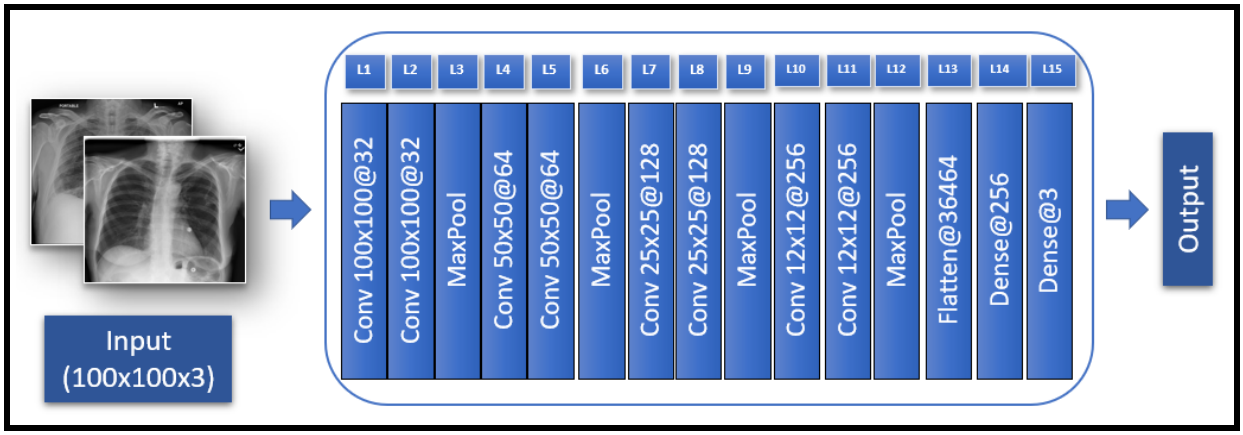


Figure 5.13: Architecture of Setup-6

Table 5.13: Hyperparamerter value's of Setup-6

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.001
Activation Function	ReLu
Kernel Size	3x3
Layer	15
Loss Function	categorical_crossentropy
MaxPooling	2x2

5.8 Transfer Learning

In transfer learning, we have used different architectures, like **VGG16, Inception V3, Xception, ResNet50, Inception ResNet V2, and DenseNet121**. We have trained every pre-trained architecture without preprocessing the dataset. Several hyperparameters have differed due to different architectures of transfer learning. Input size has changed depending on the architecture. Therefore, input sizes, various hyperparameters of the architectures are discussed in several sections below.

5.8.1 VGG16

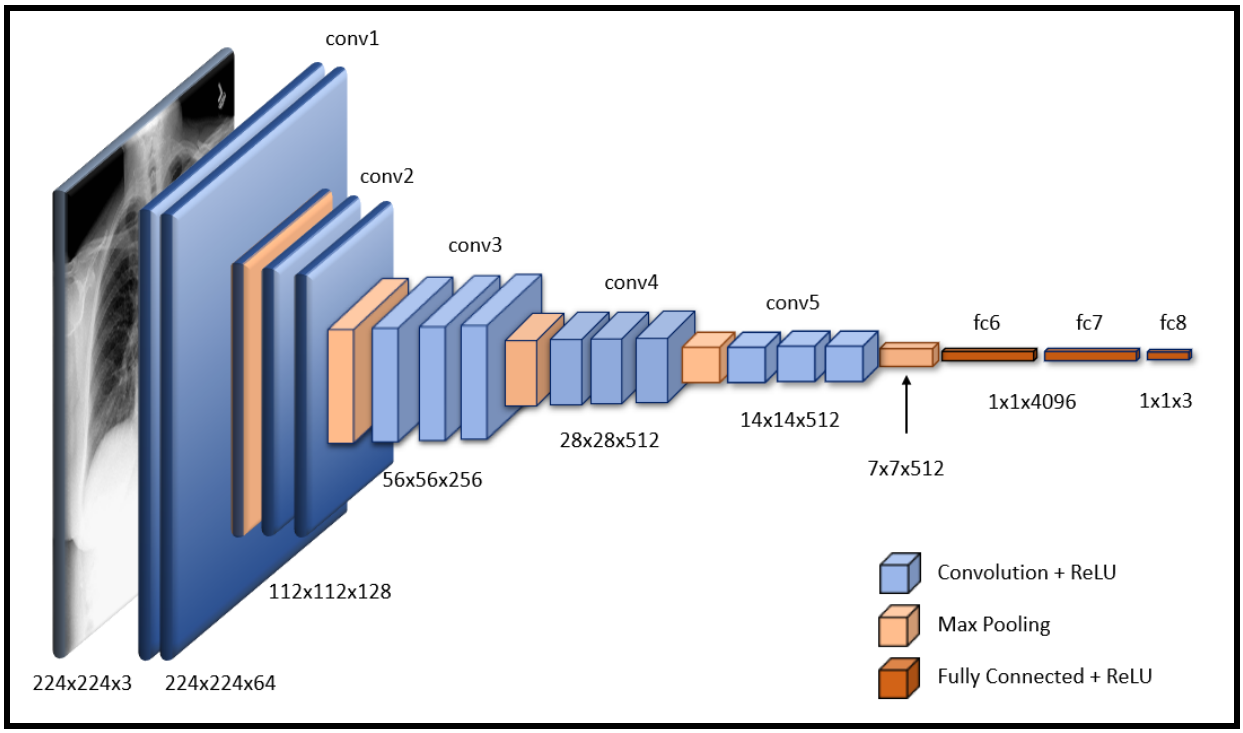


Figure 5.14: VGG16 Architecture

To leverage transfer learning, the VGG16 architecture along with its weights that were pre-trained on ImageNet has been used in this work. This architecture consists of 16 layers. Each of these convolutional blocks is input to an activation function. Here as well Rectified Linear Units (ReLU) is used as the activation function as well. We have updated the network as follows. From the VGG16 architecture, originally present fully connected layers pre-trained on ImageNet with a 1000-way softmax activation are discarded. This is followed by adding a single fully connected layer with 3-way softmax activation to get the probability results for the classification problem.

Table 5.14: VGG16 based model configuration and Hyper-parameter setting

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.01
Input Size	224x224
Layer	16
Loss Function	categorical_crossentropy
Pre-trained Weights	ImageNet

5.8.2 Inception V3

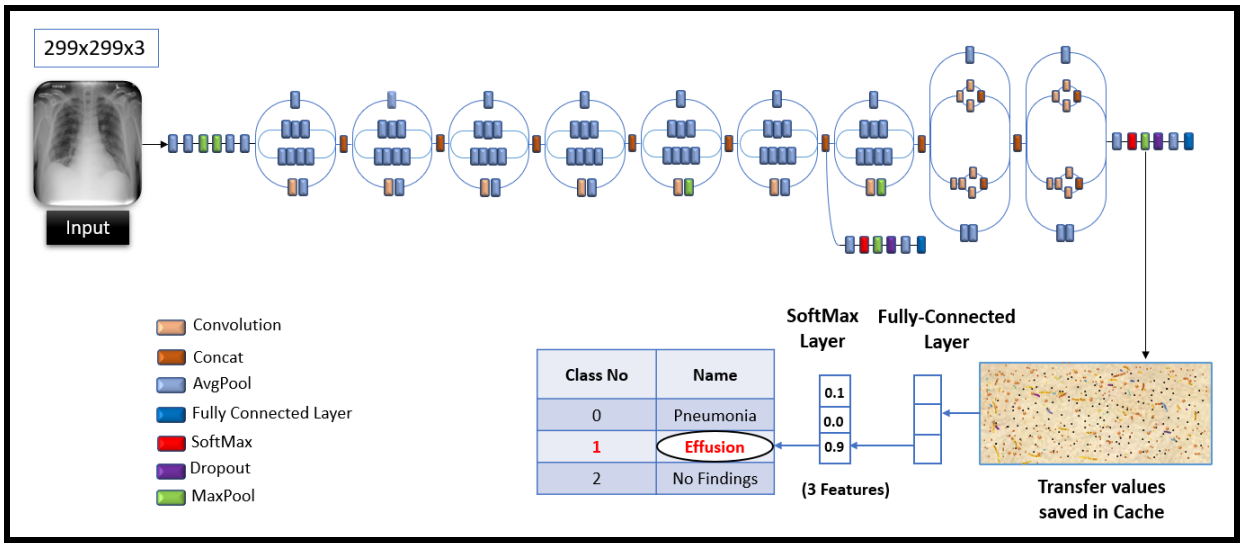


Figure 5.15: Inception V3 Architecture

To leverage transfer learning, the Inception V3 architecture along with its weights that were pre-trained on ImageNet has been used in this work. This architecture consists of 48 layers. Each of these convolutional blocks is input to an activation function. Here as well Rectified Linear Units (ReLU) is used as the activation function as well. We have updated the network as follows. From the Inception V3 architecture, originally present fully connected layers pre-trained on ImageNet with a 1000-way softmax activation are discarded. This is followed by adding a single fully connected layer with 3-way softmax activation to get the probability results for the classification problem.

Table 5.15: Inception V3 based model configuration and Hyper-parameter setting

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.01
Input Size	299x299
Layer	48
Loss Function	categorical_crossentropy
Pre-trained Weights	ImageNet

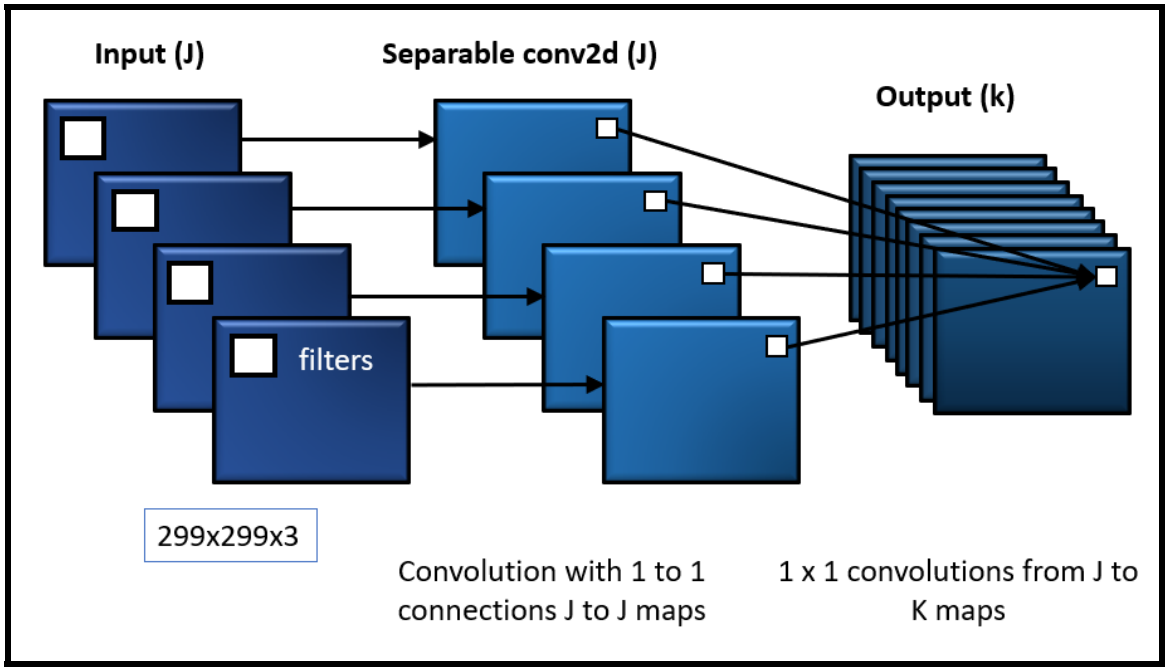


Figure 5.16: Xception Architecture

5.8.3 Xception

To leverage transfer learning, the Xception architecture along with its weights that were pre-trained on ImageNet has been used in this work. This architecture consists of 71 layers. Each of these convolutional blocks is input to an activation function. Here as well Rectified Linear Units (ReLU) is used as the activation function as well. We have updated the network as follows. From the Xception architecture, originally present fully connected layers pre-trained on ImageNet with a 1000-way softmax activation are discarded. This is followed by adding a single fully connected layer with 3-way softmax activation to get the probability results for the classification problem.

Table 5.16: Xception based model configuration and Hyper-parameter setting

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.01
Input Size	299x299
Layer	71
Loss Function	categorical_crossentropy
Pre-trained Weights	ImageNet

5.8.4 ResNet50

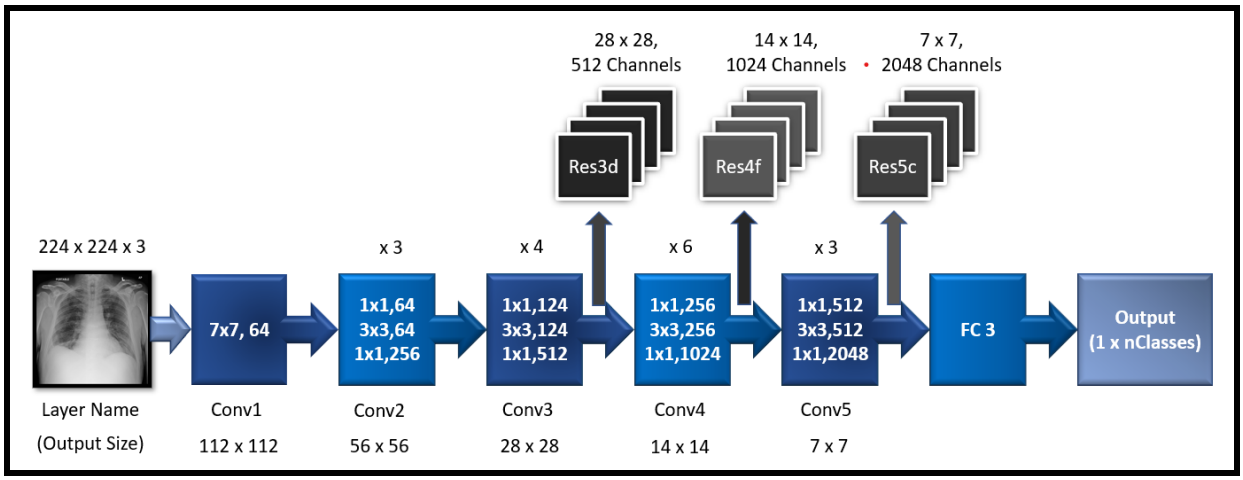


Figure 5.17: ResNet50 Architecture

To leverage transfer learning, the ResNet50 architecture along with its weights that were pre-trained on ImageNet has been used in this work. This architecture consists of 50 layers. Each of these convolutional blocks is input to an activation function. Here as well Rectified Linear Units (ReLU) is used as the activation function as well. We have updated the network as follows. From the ResNet50 architecture, originally present fully connected layers pre-trained on ImageNet with a 1000-way softmax activation are discarded. This is followed by adding a single fully connected layer with 3-way softmax activation to get the probability results for the classification problem.

5.8.5 Inception-Resnet-V2

To leverage transfer learning, the Inception-Resnet-V2 architecture along with its weights that were pre-trained on ImageNet has been used in this work. This architecture consists of

Table 5.17: ResNet50 based model configuration and Hyper-parameter setting

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.01
Input Size	224x224
Layer	50
Loss Function	categorical_crossentropy
Pre-trained Weights	ImageNet

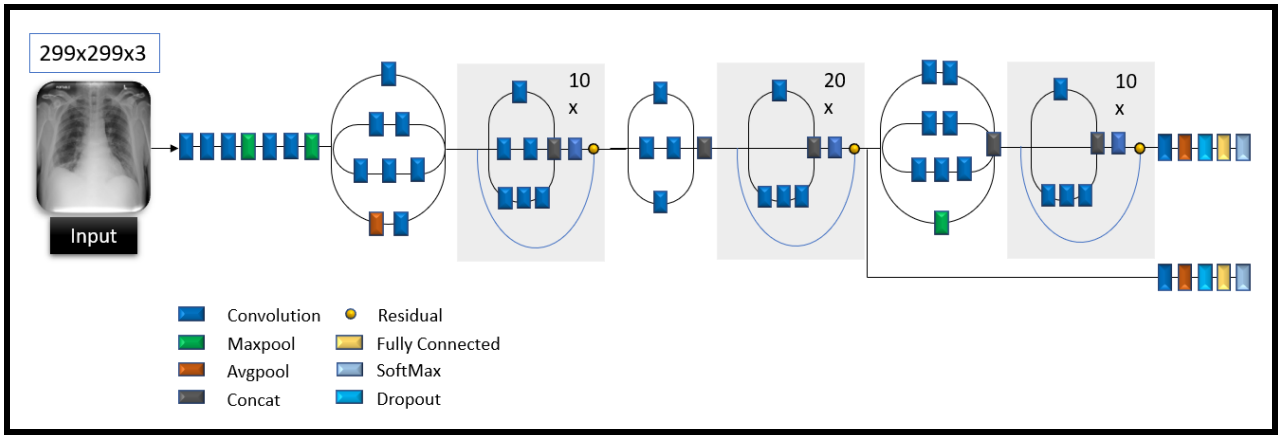


Figure 5.18: Inception-Resnet-V2 Architecture

164 layers. Each of these convolutional blocks is input to an activation function. Here as well Rectified Linear Units (ReLU) is used as the activation function as well. We have updated the network as follows. From the Inception-Resnet-V2 architecture, originally present fully connected layers pre-trained on ImageNet with a 1000-way softmax activation are discarded. This is followed by adding a single fully connected layer with 3-way softmax activation to get the probability results for the classification problem.

5.8.6 DenseNet121

To leverage transfer learning, the DenseNet121 architecture along with its weights that were pre-trained on ImageNet has been used in this work. This architecture consists of 164 layers. Each of these convolutional blocks is input to an activation function. Here as well Rectified Linear Units (ReLU) is used as the activation function as well. We have updated the network as follows. From the DenseNet121 architecture, originally present fully connected layers pre-trained on ImageNet with a 1000-way softmax activation are discarded. This is

Table 5.18: Inception-Resnet-V2 based model configuration and Hyper-parameter setting

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.01
Input Size	299x299
Layer	164
Loss Function	categorical_crossentropy
Pre-trained Weights	ImageNet

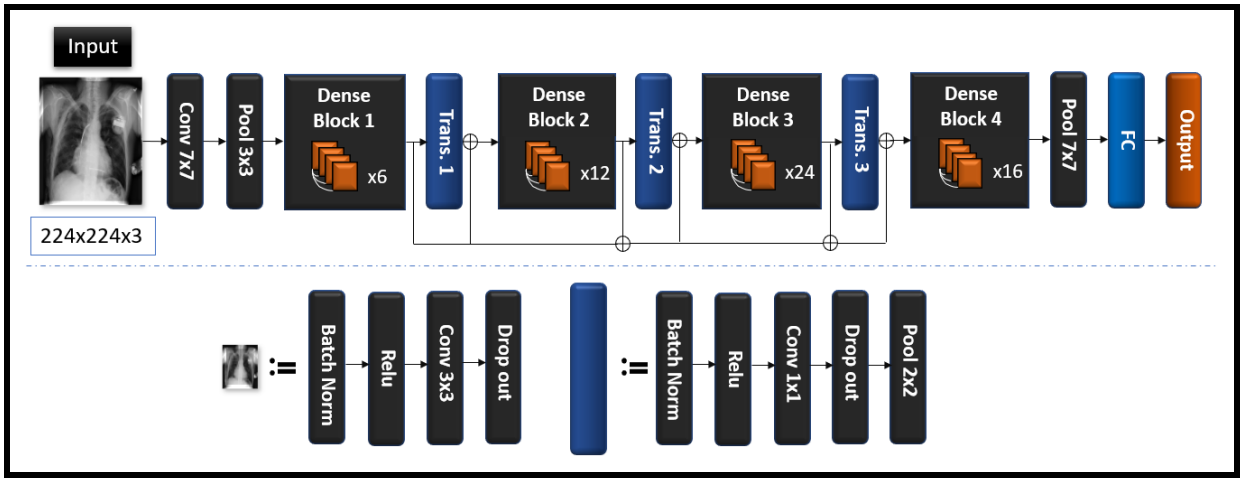


Figure 5.19: DenseNet121 Architecture

followed by adding a single fully connected layer with 3-way softmax activation to get the probability results for the classification problem.

Table 5.19: DenseNet121 based model configuration and Hyper-parameter setting

Parameter's Name	Values
Batch Size	32
Optimizer	Adam
Learning Rate	0.01
Input Size	224x224
Layer	121
Loss Function	categorical_crossentropy
Pre-trained Weights	ImageNet

Chapter 6

Experimental Process and Results

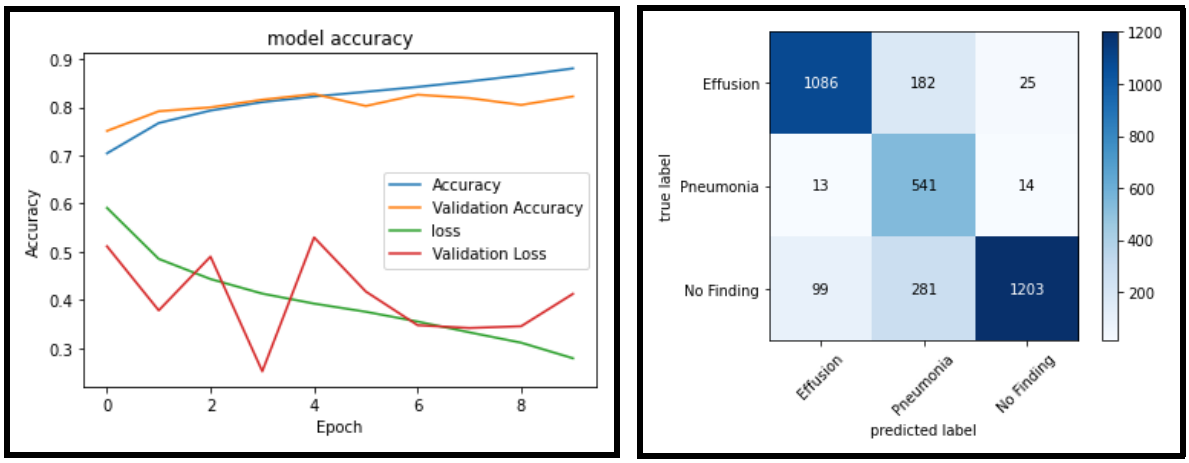
In this chapter, we have discussed the training accuracy, validation accuracy, and their loss of custom CNN and pre-trained models. We have also discussed the whole experimental process, ablation analysis to see which one ultimately provides better accuracy, and comparison between both architectures. In this experiment, we have used three approaches of the dataset- without preprocess, with CLAHE, and Power Law with sharpness and contrast. Each result is explained in a specific section through graphs and tables.

6.1 Experimental process of CNN

In each setup, we have changed the layers and hyperparameters of the CNN model one by one for checking the result without preprocessing, with CLAHE and with Power Law Transformations. We have changed the layer from setup-1 to setup-6 and checked the result by keeping ReLU as the activation function and Adam as the optimizer. But from setup-7 to setup-11, we have kept the layer as setup-5 but varied filter size, learning rate, and optimizer. In setup-6, we have increased the layer number from twelve to fifteen. But that downgraded the validation accuracy. So, in setup-7, we varied the filter size from 3x3 to 5x5, and that improved validation accuracy. Then in setup-8, we again varied the filter size from 5x5 to 7x7, and that decreased the result. In setup-9, we kept everything as setup-7 but changed the learning rate from 0.001 to 0.01, and that again increased validation accuracy. In setup-10, we just changed the optimizer from setup-9 from Adam to SGD, and that decreased the validation accuracy compared to setup-9's. Finally, in setup-11, we kept everything as setup-9 except the activation function. Here we changed it from ReLu to tanh. As seen, accuracy is precise for twelve layers and optimum for the conversion of three datasets in the setup-9. So, through this ablation experiment, it can be said that setup-9 is better than other setups. Then we have calculated the testing result of setup-9 by evaluation

metric and store the results in the corresponding table.

6.1.1 CNN without Preprocess



(a) Training Accuracy and Loss

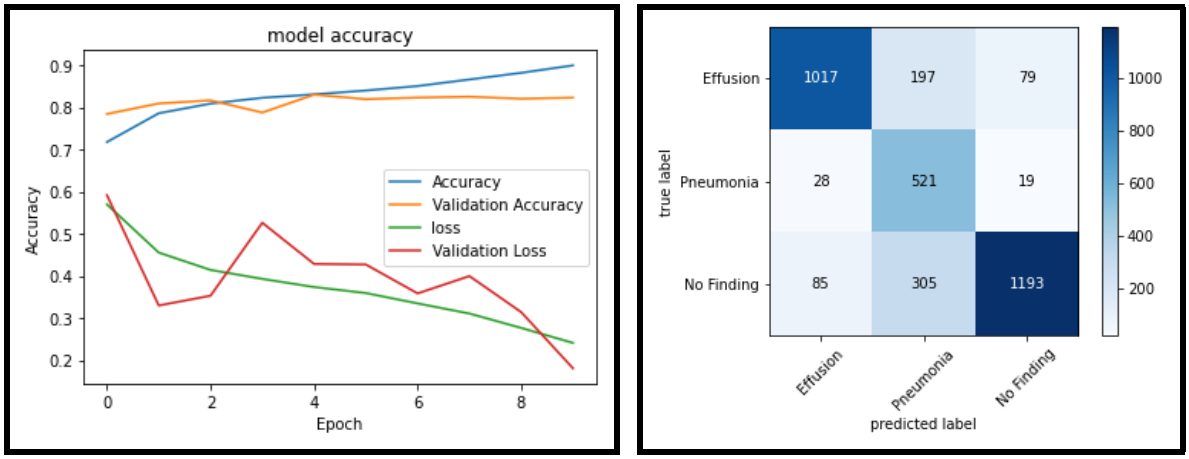
(b) Confusion Matrix

Figure 6.1: Training Accuracy, Loss and Confusion Matrix of Setup-9

From the above fig. 6.1, it is evident that this setup is really good. Here, the training and validation accuracy and loss is shown in fig. 6.1a and confusion matrix of setup-9 in fig. 6.1b. This setup-9 of CNN is trained without Preprocess. Here, we have used early stopping. If the validation accuracy of a setup didn't increase after consecutive five epochs then the training has stopped. Moreover, the setup converges after 6 epochs and achieves a training accuracy of 83.89% and validation accuracy 83.72%. After the model has been trained, it is tested using 3444 images. The result after testing the setup is presented above through a confusion matrix. From the fig, Confusion matrix, Precision, Recall, and the F1 score is calculated for each class and also the accuracy, macro average precision, recall and F1 score of the entire setup which is stated in the table 6.1 below.

Table 6.1: CNN Setup-9 without Preprocess (normal)

	Precision	**Recall**	**F1 Score**	**Support**
Effusion	0.9065	0.8399	0.8719	1293
Pneumonia	0.5388	0.9525	0.6883	568
No Finding	0.9686	0.7599	0.8517	1583
Macro avg.	0.8046	0.8508	0.8040	
Accuracy	82.17%			



(a) Training Accuracy and Loss

(b) Confusion Matrix

Figure 6.2: Training Accuracy, Loss and Confusion Matrix of Setup-9

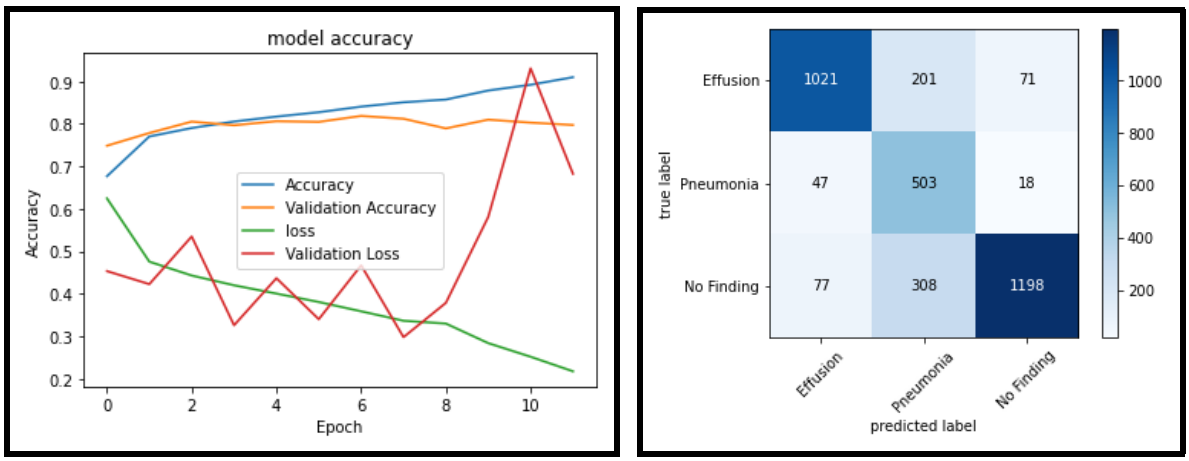
6.1.2 CNN with CLAHE

From the above fig. 6.2, it is evident that this setup is really good. Here, the training and validation accuracy and loss is shown in fig. 6.2a and confusion matrix of setup-9 in fig. 6.2b. This setup-9 of CNN is trained with CLAHE. Here, we have used early stopping. If the validation accuracy of a setup didn't increase after consecutive five epochs then the training has stopped. Moreover, the setup converges after 5 epochs and achieves a training accuracy of 83.05% and validation accuracy 82.81%. After the model has been trained, it is tested using 3444 images. The result after testing the setup is presented above through a confusion matrix. From the fig, Confusion matrix, Precision, Recall, and the F1 score is calculated for each class and also the accuracy, macro average precision, recall and F1 score of the entire setup which is stated in the table 6.2 below.

Table 6.2: CNN Setup-9 with CLAHE

	Precision	**Recall**	**F1 Score**	**Support**
Effusion	0.9000	0.7865	0.8394	1293
Pneumonia	0.5093	0.9173	0.6549	568
No Finding	0.9241	0.7536	0.8302	1583
Macro avg.	0.7778	0.8191	0.7748	
Accuracy	79.30%			

6.1.3 CNN with Power Law



(a) Training Accuracy and Loss

(b) Confusion Matrix

Figure 6.3: Training Accuracy, Loss and Confusion Matrix of Setup-9

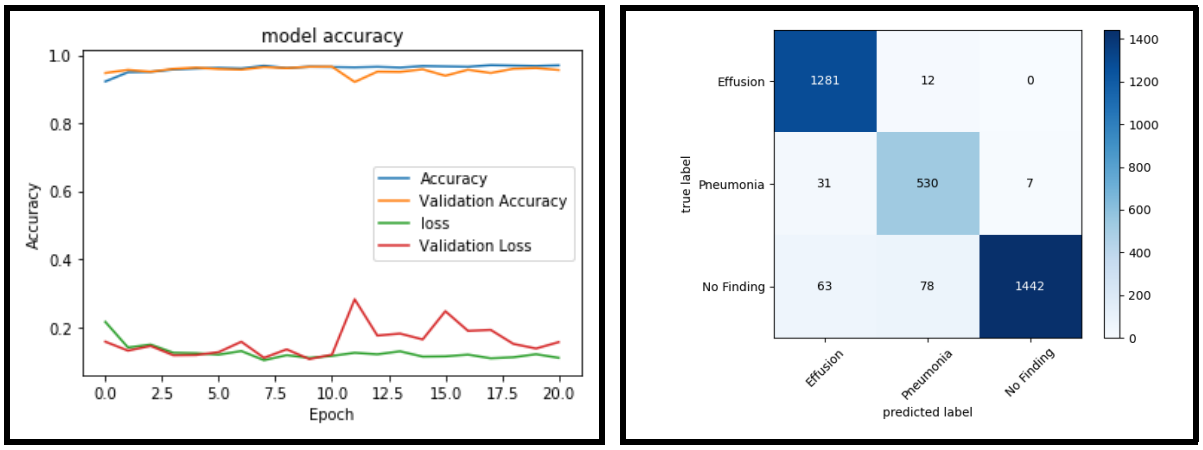
From the above fig. 6.3, it is evident that this setup is really good. Here, the training and validation accuracy and loss is shown in fig. 6.3a and confusion matrix of setup-9 in fig. 6.3b. This setup-9 of CNN is trained with Power law. Here, we have used early stopping. If the validation accuracy of a setup didn't increase after consecutive five epochs then the training has stopped. Moreover, the setup converges after 7 epochs and achieves a training accuracy of 84.06% and validation accuracy 81.87%. After the model has been trained, it is tested using 3444 images. The result after testing the setup is presented above through a confusion matrix. From the fig, Confusion matrix, Precision, Recall, and the F1 score is calculated for each class and also the accuracy, macro average precision, recall and F1 score of the entire setup which is stated in the table 6.3 below.

Table 6.3: CNN Setup-9 with Power Law

	Precision	**Recall**	**F1 Score**	**Support**
Effusion	0.8917	0.7896	0.8375	1293
Pneumonia	0.4970	0.8856	0.6367	568
No Finding	0.9308	0.7567	0.8348	1583
Macro avg.	0.7732	0.8106	0.7697	
Accuracy	79.04%			

6.2 Transfer Learning without Preprocess (Normal)

6.2.1 VGG16



(a) Training Accuracy and Loss

(b) Confusion Matrix

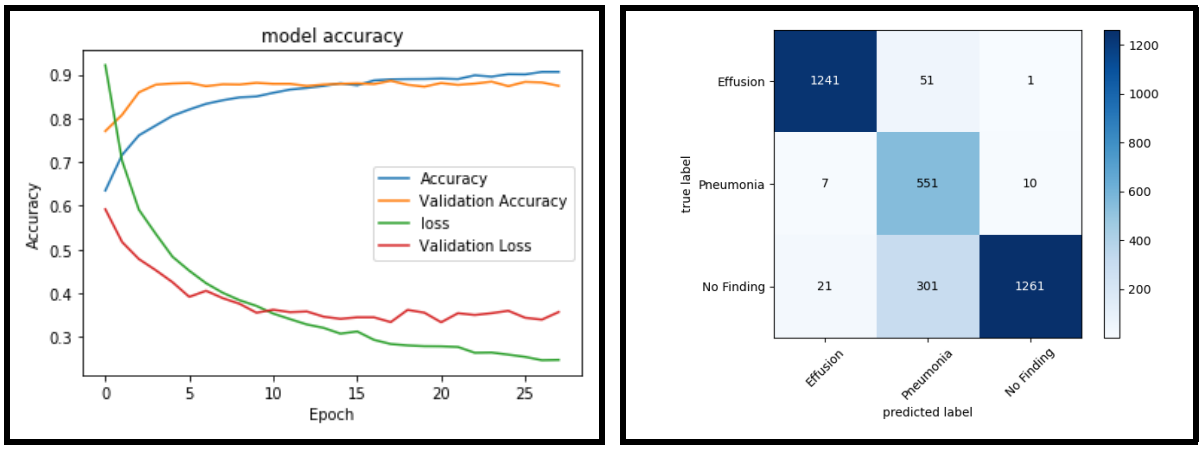
Figure 6.4: Training Accuracy, Loss and Confusion Matrix of VGG16

From the above fig. 6.4, It is evident that this VGG16 pre-trained model is performing well. Here, the training and validation accuracy and loss is shown in fig. 6.4a and confusion matrix of VGG16 in fig. 6.4b. This VGG16 is trained without preprocess. Here, we have used early stopping. If the validation accuracy of a setup didn't increase after consecutive ten epochs then the training has stopped. Moreover, VGG16 converges after 10 epochs and achieves a training accuracy of 96.98% and validation accuracy 96.83%. After the VGG16 has been trained, it is tested using 3444 images. The result after testing the pretrained model is presented above through a confusion matrix. From the fig, Confusion matrix, Precision, Recall, and the F1 score is calculated for each class and also the accuracy, macro average precision, recall and F1 score of the entire setup which is stated in the table 6.4 below.

Table 6.4: VGG16 without Preprocess (Normal)

	Precision	Recall	F1 Score	Support
Effusion	0.9316	0.9907	0.9602	1293
Pneumonia	0.8545	0.9331	0.8921	568
No Finding	0.9952	0.9109	0.9512	1583
Macro avg.	0.9271	0.9449	0.9345	
Accuracy	94.45%			

6.2.2 Inception V3



(a) Training Accuracy and Loss

(b) Confusion Matrix

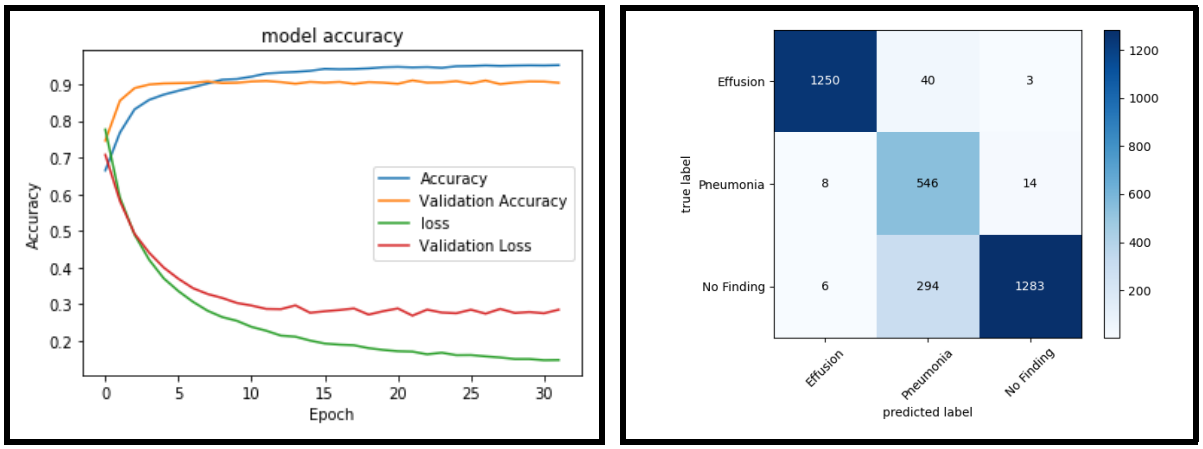
Figure 6.5: Training Accuracy, Loss and Confusion Matrix of Inception V3

From the above fig. 6.5, It is evident that this Inception V3 pre-trained model is performing well. Here, the training and validation accuracy and loss is shown in fig. 6.5a and confusion matrix of Inception V3 in fig. 6.5b. This Inception V3 is trained without preprocess. Here, we have used early stopping. If the validation accuracy of a setup didn't increase after consecutive ten epochs then the training has stopped. Moreover, Inception V3 converges after 10 epochs and achieves a training accuracy of 89.92% and validation accuracy 88.72%. After the Inception V3 has been trained, it is tested using 3444 images. The result after testing the pretrained model is presented above through a confusion matrix. From the fig, Confusion matrix, Precision, Recall, and the F1 score is calculated for each class and also the accuracy, macro average precision, recall and F1 score of the entire setup which is stated in the table 6.5 below.

Table 6.5: Inception V3 without Preprocess (Normal)

	Precision	**Recall**	**F1 Score**	**Support**
Effusion	0.9779	0.9597	0.9687	1293
Pneumonia	0.6102	0.9701	0.7492	568
No Finding	0.9914	0.7966	0.8834	1583
Macro avg.	0.8598	0.9088	0.8671	
Accuracy	88.65%			

6.2.3 Xception



(a) Training Accuracy and Loss (b) Confusion Matrix

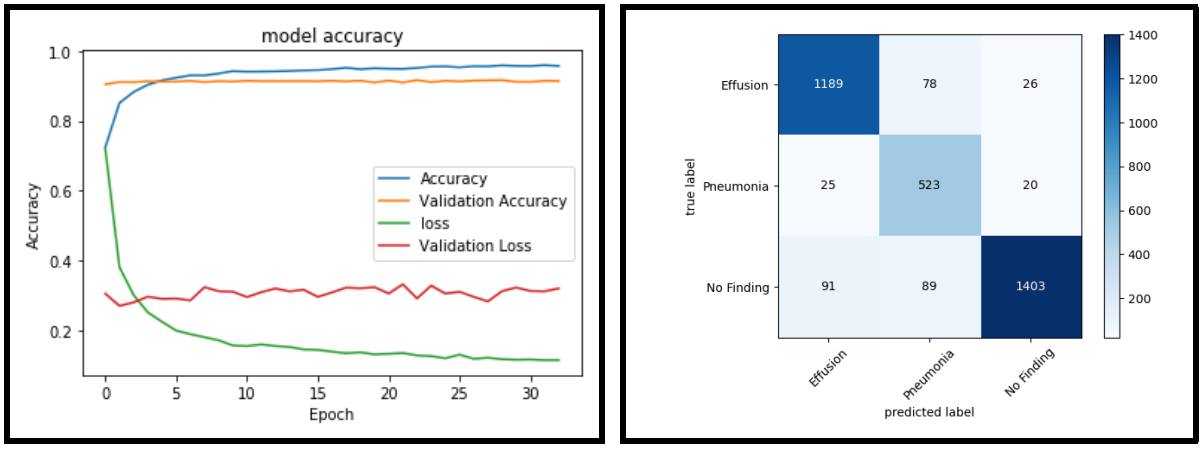
Figure 6.6: Training Accuracy, Loss and Confusion Matrix of Xception

From the above fig. 6.6, It is evident that this Xception pre-trained model is performing well. Here, the training and validation accuracy and loss is shown in fig. 6.6a and confusion matrix of Xception in fig. 6.6b. This Xception is trained without preprocess. Here, we have used early stopping. If the validation accuracy of a setup didn't increase after consecutive ten epochs then the training has stopped. Moreover, Xception converges after 22 epochs and achieves a training accuracy of 95.75% and validation accuracy 89.68%. After the Xception has been trained, it is tested using 3444 images. The result after testing the pretrained model is presented above through a confusion matrix. From the fig, Confusion matrix, Precision, Recall, and the F1 score is calculated for each class and also the accuracy, macro average precision, recall and F1 score of the entire setup which is stated in the table 6.6 below.

Table 6.6: Xception without Preprocess (Normal)

	Precision	Recall	F1 Score	Support
Effusion	0.9889	0.9667	0.9776	1293
Pneumonia	0.6205	0.9613	0.7542	568
No Finding	0.9869	0.8105	0.8900	1583
Macro avg.	0.8654	0.9128	0.8739	
Accuracy	89.40%			

6.2.4 ResNet50



(a) Training Accuracy and Loss

(b) Confusion Matrix

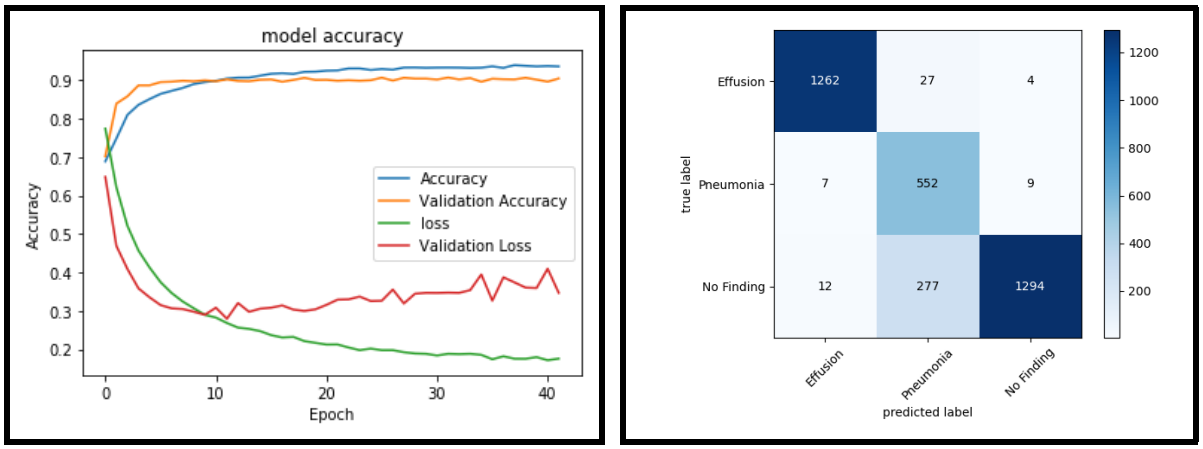
Figure 6.7: Training Accuracy, Loss and Confusion Matrix of ResNet50

From the above fig. 6.7, It is evident that this ResNet50 pre-trained model is performing well. Here, the training and validation accuracy and loss is shown in fig. 6.7a and confusion matrix of ResNet50 in fig. 6.7b. This ResNet50 is trained without preprocess. Here, we have used early stopping. If the validation accuracy of a setup didn't increase after consecutive ten epochs then the training has stopped. Moreover, ResNet50 converges after 23 epochs and achieves a training accuracy of 97.23% and validation accuracy 92.35%. After the ResNet50 has been trained, it is tested using 3444 images. The result after testing the pretrained model is presented above through a confusion matrix. From the fig, Confusion matrix, Precision, Recall, and the F1 score is calculated for each class and also the accuracy, macro average precision, recall and F1 score of the entire setup which is stated in the table 6.7 below.

Table 6.7: ResNet50 without Preprocess (Normal)

	Precision	**Recall**	**F1 Score**	**Support**
Effusion	0.9111	0.9196	0.9153	1293
Pneumonia	0.7580	0.9208	0.8315	568
No Finding	0.9683	0.8863	0.9255	1583
Macro avg.	0.8791	0.9089	0.8908	
Accuracy	90.45%			

6.2.5 Inception-ResNet-V2



(a) Training Accuracy and Loss

(b) Confusion Matrix

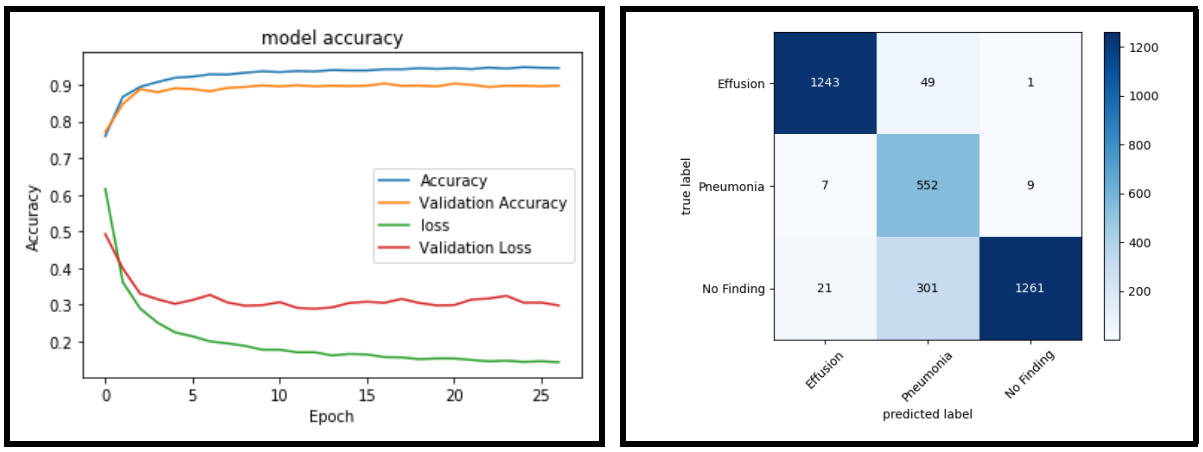
Figure 6.8: Training Accuracy, Loss and Confusion Matrix of Inception-ResNet-V2

From the above fig. 6.8, It is evident that this Inception-ResNet-V2 pre-trained model is performing well. Here, the training and validation accuracy and loss is shown in fig. 6.8a and confusion matrix of Inception-ResNet-V2 in fig. 6.8b. This Inception-ResNet-V2 is trained without preprocess. Here, we have used early stopping. If the validation accuracy of a setup didn't increase after consecutive ten epochs then the training has stopped. Moreover, Inception-ResNet-V2 converges after 31 epochs and achieves a training accuracy of 96.27% and validation accuracy 90.92%. After the Inception-ResNet-V2 has been trained, it is tested using 3444 images. The result after testing the pretrained model is presented above through a confusion matrix. From the fig, Confusion matrix, Precision, Recall, and the F1 score is calculated for each class and also the accuracy, macro average precision, recall and F1 score of the entire setup which is stated in the table 6.8 below.

Table 6.8: Inception-ResNet-V2 without Preprocess (Normal)

	Precision	**Recall**	**F1 Score**	**Support**
Effusion	0.9852	0.9760	0.9806	1293
Pneumonia	0.6449	0.9718	0.7753	568
No Finding	0.9901	0.8174	0.8955	1583
Macro avg.	0.8734	0.9217	0.8838	
Accuracy	90.24%			

6.2.6 DenseNet121



(a) Training Accuracy and Loss

(b) Confusion Matrix

Figure 6.9: Training Accuracy, Loss and Confusion Matrix of DenseNet121

From the above fig. 6.9, It is evident that this DenseNet121 pre-trained model is performing well. Here, the training and validation accuracy and loss is shown in fig. 6.9a and confusion matrix of DenseNet121 in fig. 6.9b. This DenseNet121 is trained without preprocess. Here, we have used early stopping. If the validation accuracy of a setup didn't increase after consecutive ten epochs then the training has stopped. Moreover, DenseNet121 converges after 17 epochs and achieves a training accuracy of 93.51% and validation accuracy 89.27%. After the DenseNet121 has been trained, it is tested using 3444 images. The result after testing the pretrained model is presented above through a confusion matrix. From the fig, Confusion matrix, Precision, Recall, and the F1 score is calculated for each class and also the accuracy, macro average precision, recall and F1 score of the entire setup which is stated in the table 6.9 below.

Table 6.9: DenseNet121 without Preprocess (Normal)

	Precision	**Recall**	**F1 Score**	**Support**
Effusion	0.9780	0.9613	0.9696	1293
Pneumonia	0.6120	0.9718	0.7510	568
No Finding	0.9921	0.7966	0.8837	1583
Macro avg.	0.8607	0.9099	0.8681	
Accuracy	88.73%			

6.3 Results

In this section, the training results have shown in the table using the ablation experiment on Custom CNN and different architectures of transfer learning.

6.3.1 Ablation Experiments of CNN

6.3.1.1 Training Evaluation-1

Table 6.10: Training Evaluation-1

Setup-1		**Without Preprocess (Normal)**	**With CLAHE**	**With Power law**
Layers	5	Loss: 0.4683	Loss: 0.427	Loss: 0.4432
Optimizer	Adam			
Learning Rate	0.001			
Activation Function	ReLu	Accuracy: 0.7771	Accuracy: 0.7957	Accuracy: 0.8125
Kernel Size	3x3			
Loss Function	Categorical_ Crossentropy	Validation Loss: 0.6068	Validation Loss: 0.3210	Validation Loss: 0.6175
Batch Size	32			
Max Pooling	2x2	Validation Accuracy: 0.7830	Validation Accuracy: 0.7929	Validation Accuracy: 0.7829
Epoch	2 (Normal) 2 (CLAHE) 4 (Power Law)			

In table 6.10, Setup-1 is five layers of the custom architecture of Neural Network with 3-way softmax activation. In this setup, we have used Adam as an Optimizer with a learning rate of 0.001 and ReLU as an activation function for Convolutional Layer. The kernel size is set to 3x3. Apart from these, there are some more hyperparameters in the table above which we have used in this setup. From the table above, we can see that if we trained the model in the above configuration without any preprocess on the dataset, then we get training accuracy 0.7771 and validation accuracy 0.7830. On the contrary, we get training accuracy 0.7957 and validation accuracy 0.7929 by trained with CLAHE on the dataset. Similarly, we get training accuracy 0.8125 and validation accuracy 0.7829 by trained with Power-law transform with sharpness and contrast on the dataset. Comparing these three approaches, we can see that the validation accuracy has improved by trained without preprocessing on the dataset in setup-1.

6.3.1.2 Training Evaluation-2

In table 6.11, Setup-2 is seven layers of the custom architecture of Neural Network with 3-way softmax activation. In this setup, we have used Adam as an Optimizer with a learning

Table 6.11: Training Evaluation-2

Setup-2		Without Preprocess (Normal)	With CLAHE	With Power law
Layers	7	Loss: 0.4013	Loss: 0.4609	Loss: 0.3547
Optimizer	Adam			
Learning Rate	0.001			
Activation Function	ReLu	Accuracy: 0.8122	Accuracy: 0.7812	Accuracy: 0.8177
Kernel Size	3x3			
Loss Function	Categorical_ Crossentropy	Validation Loss: 0.3472	Validation Loss: 0.5303	Validation Loss: 0.2373
Batch Size	32			
Max Pooling	2x2	Validation Accuracy: 0.7999	Validation Accuracy: 0.8028	Validation Accuracy: 0.7833
Epoch	4 (Normal) 17 (CLAHE) 5 (Power Law)			

rate of 0.001 and ReLU as an activation function for Convolutional Layer. The kernel size is set to 3x3. Apart from these, there are some more hyperparameters in the table above which we have used in this setup. From the table above, we can see that if we trained the model in the above configuration without any preprocess on the dataset, then we get training accuracy 0.8122 and validation accuracy 0.7999. On the contrary, we get training accuracy 0.7812 and validation accuracy 0.8028 by trained with CLAHE on the dataset. Similarly, we get training accuracy 0.8177 and validation accuracy 0.7833 by trained with Power-law transform with sharpness and contrast on the dataset. Comparing these three approaches, we can see that the validation accuracy has improved by trained without preprocessing on the dataset in setup-2 as before.

6.3.1.3 Training Evaluation-3

Table 6.12: Training Evaluation-3

Setup-3		Without Preprocess (Normal)	With CLAHE	With Power law
Layers	9	Loss: 0.4045	Loss: 0.4008	Loss: 0.9871
Optimizer	Adam			
Learning Rate	0.001			
Activation Function	ReLu	Accuracy: 0.8126	Accuracy: 0.8085	Accuracy: 0.8198
Kernel Size	3x3			
Loss Function	Categorical_ Crossentropy	Validation Loss: 0.3768	Validation Loss: 0.5480	Validation Loss: 0.4470
Batch Size	32			
Max Pooling	2x2	Validation Accuracy: 0.8007	Validation Accuracy: 0.8125	Validation Accuracy: 0.7999
Epoch	4 (Normal) 3 (CLAHE) 5 (Power Law)			

In table 6.12, Setup-3 is nine layers of the custom architecture of Neural Network with 3-

way softmax activation. In this setup, we have used Adam as an Optimizer with a learning rate of 0.001 and ReLU as an activation function for Convolutional Layer. The kernel size is set to 3x3. Apart from these, there are some more hyperparameters in the table above which we have used in this setup. From the table above, we can see that if we trained the model in the above configuration without any preprocess on the dataset, then we get training accuracy 0.8126 and validation accuracy 0.8007. On the contrary, we get training accuracy 0.8085 and validation accuracy 0.8125 by trained with CLAHE on the dataset. Similarly, we get training accuracy 0.8198 and validation accuracy 0.7999 by trained with Power-law transform with sharpness and contrast on the dataset. Comparing these three approaches, we can see that the validation accuracy has improved by trained without preprocessing on the dataset in setup-3 as before.

6.3.1.4 Training Evaluation-4

Table 6.13: Training Evaluation-4

<table>
<tr><th colspan="2">Setup-4</th><th>Without Preprocess (Normal)</th><th>With CLAHE</th><th>With Power law</th></tr>
<tr><td>Layers</td><td>10</td><td rowspan="3">Loss: 0.3838</td><td rowspan="3">Loss: 0.3905</td><td rowspan="3">Loss: 0.3866</td></tr>
<tr><td>Optimizer</td><td>Adam</td></tr>
<tr><td>Learning Rate</td><td>0.001</td></tr>
<tr><td>Activation Function</td><td>ReLu</td><td rowspan="2">Accuracy: 0.8245</td><td rowspan="2">Accuracy: 0.8187</td><td rowspan="2">Accuracy: 0.8205</td></tr>
<tr><td>Kernel Size</td><td>3x3</td></tr>
<tr><td>Loss Function</td><td>Categorical_ Crossentropy</td><td rowspan="2">Validation Loss: 0.484</td><td rowspan="2">Validation Loss: 0.3794</td><td rowspan="2">Validation Loss: 0.6122</td></tr>
<tr><td>Batch Size</td><td>32</td></tr>
<tr><td>Max Pooling</td><td>2x2</td><td rowspan="2">Validation Accuracy: 0.8116</td><td rowspan="2">Validation Accuracy: 0.8160</td><td rowspan="2">Validation Accuracy: 0.8054</td></tr>
<tr><td>Epoch</td><td>4 (Normal)
4 (CLAHE)
5 (Power Law)</td></tr>
</table>

In table 6.13, Setup-4 is ten layers of the custom architecture of Neural Network with 3-way softmax activation. In this setup, we have used Adam as an Optimizer with a learning rate of 0.001 and ReLU as an activation function for Convolutional Layer. The kernel size is set to 3x3. Apart from these, there are some more hyperparameters in the table above which we have used in this setup. From the table above, we can see that if we trained the model in the above configuration without any preprocess on the dataset, then we get training accuracy 0.8245 and validation accuracy 0.8116. On the contrary, we get training accuracy 0.8187 and validation accuracy 0.8160 by trained with CLAHE on the dataset. Similarly, we get training accuracy 0.8205 and validation accuracy 0.8054 by trained with Power-law transform with sharpness and contrast on the dataset. Comparing these three approaches, we can see that the validation accuracy has improved by trained without preprocessing on the dataset in setup-4 as before.

6.3.1.5 Training Evaluation-5

Table 6.14: Training Evaluation-5

Setup-5		Without Preprocess (Normal)	With CLAHE	With Power law
Layers	12	Loss: 0.3954	Loss: 0.3739	Loss: 0.3003
Optimizer	Adam			
Learning Rate	0.001			
Activation Function	ReLu	Accuracy: 0.8151	Accuracy: 0.8311	Accuracy: 0.8648
Kernel Size	3x3			
Loss Function	Categorical_ Crossentropy	Validation Loss: 0.1918	Validation Loss: 0.3634	Validation Loss: 0.5041
Batch Size	32			
Max Pooling	2x2	Validation Accuracy: 0.8167	Validation Accuracy: 0.8282	Validation Accuracy: 0.8096
Epoch	5 (Normal) 4 (CLAHE) 7 (Power Law)			

In table 6.14, Setup-5 is twelve layers of the custom architecture of Neural Network with 3-way softmax activation. In this setup, we have used Adam as an Optimizer with a learning rate of 0.001 and ReLU as an activation function for Convolutional Layer. The kernel size is set to 3x3. Apart from these, there are some more hyperparameters in the table above which we have used in this setup. From the table above, we can see that if we trained the model in the above configuration without any preprocess on the dataset, then we get training accuracy 0.8151 and validation accuracy 0.8167. On the contrary, we get training accuracy 0.8311 and validation accuracy 0.8282 by trained with CLAHE on the dataset. Similarly, we get training accuracy 0.8648 and validation accuracy 0.8096 by trained with Power-law transform with sharpness and contrast on the dataset. Comparing these three approaches, we can see that the validation accuracy has improved by trained without preprocessing on the dataset in setup-5 as before. Furthermore, It can be seen that validation accuracy is increasing as the layer of architecture grows from setup-1 to setup-5.

6.3.1.6 Training Evaluation-6

In table 6.15, Setup-6 is fifteen layers of the custom architecture of Neural Network with 3-way softmax activation. In this setup, we have used Adam as an Optimizer with a learning rate of 0.001 and ReLU as an activation function for Convolutional Layer. The kernel size is set to 3x3. Apart from these, there are some more hyperparameters in the table above which we have used in this setup. From the table above, we can see that if we trained the model in the above configuration without any preprocess on the dataset, then we get training accuracy 0.8148 and validation accuracy 0.8190. On the contrary, we get training accuracy 0.8260 and validation accuracy 0.8218 by trained with CLAHE on the dataset. Similarly, we get training accuracy 0.8262 and validation accuracy 0.8239 by trained with Power-law

Table 6.15: Training Evaluation-6

Setup-6		Without Preprocess (Normal)	With CLAHE	With Power law
Layers	15	Loss: 0.4088	Loss: 0.3840	Loss: 0.3798
Optimizer	Adam			
Learning Rate	0.001			
Activation Function	ReLu	Accuracy: 0.8148	Accuracy: 0.8260	Accuracy: 0.8262
Kernel Size	3x3			
Loss Function	Categorical_ Crossentropy	Validation Loss: 0.3395	Validation Loss: 0.3262	Validation Loss: 0.4714
Batch Size	32			
Max Pooling	2x2	Validation Accuracy: 0.8190	Validation Accuracy: 0.8218	Validation Accuracy: 0.8239
Epoch	4 (Normal) 4 (CLAHE) 5 (Power Law)			

transform with sharpness and contrast on the dataset. Comparing these three approaches, we can see that the validation accuracy has decreased by trained without preprocessing on the dataset in setup-6. Comparing Setup-5 and Setup-6, it can be seen that the validation accuracy has decreased due to the increase in the number of layers of architecture. So twelve layers is appropriate for this experiment.

6.3.1.7 Training Evaluation-7

Table 6.16: Training Evaluation-7

Setup-7		Without Preprocess (Normal)	With CLAHE	With Power law
Layers	12	Loss: 0.3412	Loss: 0.3800	Loss: 0.3516
Optimizer	Adam			
Learning Rate	0.001			
Activation Function	ReLu	Accuracy: 0.8476	Accuracy: 0.8283	Accuracy: 0.8424
Kernel Size	5x5			
Loss Function	Categorical_ Crossentropy	Validation Loss: 0.6614	Validation Loss: 0.3509	Validation Loss: 0.2077
Batch Size	32			
Max Pooling	2x2	Validation Accuracy: 0.8249	Validation Accuracy: 0.8247	Validation Accuracy: 0.8174
Epoch	9 (Normal) 4 (CLAHE) 10 (Power Law)			

In table 6.16, Setup-7 is twelve layers of the custom architecture of Neural Network with 3-way softmax activation. In this setup, we have used Adam as an Optimizer with a learning rate of 0.001 and ReLU as an activation function for Convolutional Layer. The kernel size is set to 5x5. Apart from these, there are some more hyperparameters in the table above which we have used in this setup. From the table above, we can see that if we trained the model in the above configuration without any preprocess on the dataset, then we get

training accuracy 0.8476 and validation accuracy 0.8249. On the contrary, we get training accuracy 0.8283 and validation accuracy 0.8247 by trained with CLAHE on the dataset. Similarly, we get training accuracy 0.8424 and validation accuracy 0.8174 by trained with Power-law transform with sharpness and contrast on the dataset. Comparing these three approaches, now we can see that the validation accuracy has improved by trained without preprocessing on the dataset in setup-7. Comparing setup-6 and setup-7, it can be seen that the validation accuracy has improved due to the number of layers of architecture being 12 and kernel size being 5x5.

6.3.1.8 Training Evaluation-8

Table 6.17: Training Evaluation-8

Setup-8		**Without Preprocess (Normal)**	**With CLAHE**	**With Power law**
Layers	12	Loss: 0.3215	Loss: 0.4154	Loss: 0.3968
Optimizer	Adam			
Learning Rate	0.001			
Activation Function	ReLu	Accuracy: 0.8267	Accuracy: 0.8126	Accuracy: 0.8198
Kernel Size	7x7			
Loss Function	Categorical_ Crossentropy	Validation Loss: 0.3694	Validation Loss: 0.3648	Validation Loss: 0.8736
Batch Size	32			
Max Pooling	2x2	Validation Accuracy: 0.7982	Validation Accuracy: 0.7855	Validation Accuracy: 0.7136
Epoch	5 (Normal) 4 (CLAHE) 7 (Power Law)			

In table 6.17, Setup-8 is twelve layers of the custom architecture of Neural Network with 3-way softmax activation. In this setup, we have used Adam as an Optimizer with a learning rate of 0.001 and ReLU as an activation function for Convolutional Layer. The kernel size is set to 7x7. Apart from these, there are some more hyperparameters in the table above which we have used in this setup. From the table above, we can see that if we trained the model in the above configuration without any preprocess on the dataset, then we get training accuracy 0.8267 and validation accuracy 0.7982. On the contrary, we get training accuracy 0.8126 and validation accuracy 0.7855 by trained with CLAHE on the dataset. Similarly, we get training accuracy 0.8198 and validation accuracy 0.7136 by trained with Power-law transform with sharpness and contrast on the dataset. Comparing these three approaches, we can see that the validation accuracy has improved by trained without preprocessing on the dataset in setup-8 as before. Comparing setup-7 and setup-8, it can be seen that the validation accuracy has decreased due to the kernel size being 7x7. So, the appropriate kernel size is 5x5 for this experiment.

6.3.1.9 Training Evaluation-9

Table 6.18: Training Evaluation-9

Setup-9 (Best)		Without Preprocess (Normal)	With CLAHE	With Power law
Layers	12	Loss: 0.3921	Loss: 0.131	Loss: 0.3589
Optimizer	Adam			
Learning Rate	0.01			
Activation Function	ReLu	Accuracy: 0.8389	Accuracy: 0.8305	Accuracy: 0.8406
Kernel Size	5x5			
Loss Function	Categorical_ Crossentropy	Validation Loss: 0.5256	Validation Loss: 0.4281	Validation Loss: 0.4622
Batch Size	32			
Max Pooling	2x2	Validation Accuracy: 0.8372	Validation Accuracy: 0.8281	Validation Accuracy: 0.8187
Epoch	6 (Normal) 5 (CLAHE) 7 (Power Law)			

In table 6.18, Setup-9 is twelve layers of the custom architecture of Neural Network with 3-way softmax activation. In this setup, we have used Adam as an Optimizer with a learning rate of 0.01 and ReLU as an activation function for Convolutional Layer. The kernel size is set to 5x5. Apart from these, there are some more hyperparameters in the table above which we have used in this setup. From the table above, we can see that if we trained the model in the above configuration without any preprocess on the dataset, then we get training accuracy 0.8389 and validation accuracy 0.8372. On the contrary, we get training accuracy 0.8305 and validation accuracy 0.8281 by trained with CLAHE on the dataset. Similarly, we get training accuracy 0.8406 and validation accuracy 0.8187 by trained with Power-law transform with sharpness and contrast on the dataset. Comparing these three approaches, we can see that the validation accuracy has improved by trained without preprocessing on the dataset in setup-9 as before. Comparing setup-8 and setup-9, it can be seen that the validation accuracy has improved due to the kernel size being 5x5 and the learning rate being 0.01. This is the best setup in our remaining experiment.

6.3.1.10 Training Evaluation-10

In table 6.19, Setup-10 is twelve layers of the custom architecture of Neural Network with 3-way softmax activation. In this setup, we have used SGD as an Optimizer with a learning rate of 0.01 and ReLU as an activation function for Convolutional Layer. The kernel size is set to 5x5. Apart from these, there are some more hyperparameters in the table above which we have used in this setup. From the table above, we can see that if we trained the model in the above configuration without any preprocess on the dataset, then we get training accuracy 0.8167 and validation accuracy 0.8140. On the contrary, we get training accuracy 0.8307 and validation accuracy 0.8227 by trained with CLAHE on the dataset.

Table 6.19: Training Evaluation-10

Setup-10		Without Preprocess (Normal)	With CLAHE	With Power law
Layers	12	Loss: 0.4015	Loss: 0.3758	Loss: 0.3750
Optimizer	SGD			
Learning Rate	0.01			
Activation Function	ReLu	Accuracy: 0.8167	Accuracy: 0.8307	Accuracy: 0.8333
Kernel Size	5x5			
Loss Function	Categorical_ Crossentropy	Validation Loss: 0.5191	Validation Loss: 0.2766	Validation Loss: 0.3304
Batch Size	32			
Max Pooling	2x2	Validation Accuracy: 0.8140	Validation Accuracy: 0.8227	Validation Accuracy: 0.8142
Epoch	9 (Normal) 8 (CLAHE) 13 (Power Law)			

Similarly, we get training accuracy 0.8333 and validation accuracy 0.8142 by trained with Power-law transform with sharpness and contrast on the dataset. Comparing these three approaches, now we can see that the validation accuracy has decreased by trained without preprocessing on the dataset in setup-10. Comparing setup-9 and setup-10, it can be seen that the validation accuracy has decreased due to the optimizer being SGD. So, this setup is not good.

6.3.1.11 Training Evaluation-11

Table 6.20: Training Evaluation-11

Setup-11		Without Preprocess (Normal)	With CLAHE	With Power law
Layers	12	Loss: 0.4015	Loss: 0.4127	Loss: 0.4128
Optimizer	Adam			
Learning Rate	0.01			
Activation Function	tanh	Accuracy: 0.8217	Accuracy: 0.8379	Accuracy: 0.8229
Kernel Size	5x5			
Loss Function	Categorical_ Crossentropy	Validation Loss: 0.6121	Validation Loss: 0.3968	Validation Loss: 0.4016
Batch Size	32			
Max Pooling	2x2	Validation Accuracy: 0.7812	Validation Accuracy: 0.7829	Validation Accuracy: 0.7892
Epoch	11 (Normal) 9 (CLAHE) 15 (Power Law)			

In table 6.20, Setup-11 is twelve layers of the custom architecture of Neural Network with 3-way softmax activation. In this setup, we have used Adam as an Optimizer with a learning rate of 0.01 and tanh as an activation function for Convolutional Layer. The kernel size is set to 5x5. Apart from these, there are some more hyperparameters in the table above which we have used in this setup. From the table above, we can see that if we trained the model in the above configuration without any preprocess on the dataset, then we get training

accuracy 0.8217 and validation accuracy 0.7812. On the contrary, we get training accuracy 0.8379 and validation accuracy 0.7829 by trained with CLAHE on the dataset. Similarly, we get training accuracy 0.8229 and validation accuracy 0.7892 by trained with Power-law transform with sharpness and contrast on the dataset. Comparing these three approaches, we can see that the validation accuracy has decreased by trained without preprocessing on the dataset in setup-11. Comparing setup-10 and setup-11, it can be seen that the validation accuracy has decreased due to the activation function being tanh. So, obviously setup-11 may not be the best choice and setup-9 is a more efficient alternative among other setups in our training evaluation of custom CNN.

6.3.2 Training Experiment of VGG16

Table 6.21: Training experiment of VGG16

VGG16		**Without Preprocess**
Error Function	Categorical Cross-Entropy	Loss: 0.1821
Pre-trained Weights	ImageNet	
Optimizer	Adam	Accuracy: 0.9698
Batch Size	32	
Epoch	10	Validation Loss: 0.1819
Input Size	224x224	
Learning Rate	0.01	Validation Accuracy: 0.9683
Layer	16	

In this pretrained model, VGG16 on ImageNet dataset have used as the classifier. Most of the layers have untouched. A new prediction layer replaces only the final layer with three-way softmax activation. In this training experiment of table 6.21, we have used Adam as an Optimizer with a learning rate of 0.01 and 'Categorical Crossentropy' as loss function. VGG16 has 16 layers and the input size is 224x224. Here, the epoch is 10 and the batch size is 32. From the table 6.21 above, we can see that if we trained VGG16 in the above configuration without preprocessing on the dataset, then we get training accuracy 0.9698 and validation accuracy 0.9683.

6.3.3 Training Experiment of Inception V3

In this pretrained model, Inception V3 on ImageNet dataset have used as the classifier. Most of the layers have untouched. A new prediction layer replaces only the final layer with three-way softmax activation. In this training experiment of table 6.22, we have used Adam as an Optimizer with a learning rate of 0.01 and 'Categorical Crossentropy' as loss function.

Table 6.22: Training experiment of Inception V3

Inception V3		**Without Preprocess**
Error Function	Categorical Cross-Entropy	Loss: 0.3129
Pre-trained Weights	ImageNet	
Optimizer	Adam	Accuracy: 0.8992
Batch Size	32	
Epoch	18	Validation Loss: 0.4213
Input Size	299x299	
Learning Rate	0.01	Validation Accuracy: 0.8872
Layer	48	

Inception V3 has 48 layers and the input size is 299x299. Here, the epoch is 18 and the batch size is 32. From the table 6.22 above, we can see that if we trained Inception V3 in the above configuration without preprocessing on the dataset, then we get training accuracy 0.8992 and validation accuracy 0.8872.

6.3.4 Training Experiment of Xception

Table 6.23: Training experiment of Xception

Xception		**Without Preprocess**
Error Function	Categorical Cross-Entropy	Loss: 0.1923
Pre-trained Weights	ImageNet	
Optimizer	Adam	Accuracy: 0.9575
Batch Size	32	
Epoch	22	Validation Loss: 0.3824
Input Size	299x299	
Learning Rate	0.01	Validation Accuracy: 0.8968
Layer	71	

In this pretrained model, Xception on ImageNet dataset have used as the classifier. Most of the layers have untouched. A new prediction layer replaces only the final layer with three-way softmax activation. In this training experiment of table 6.23, we have used Adam as an Optimizer with a learning rate of 0.01 and 'Categorical Crossentropy' as loss function. Xception has 71 layers and the input size is 299x299. Here, the epoch is 22 and the batch size is 32. From the table 6.23 above, we can see that if we trained Xception in the above configuration without preprocessing on the dataset, then we get training accuracy 0.9575 and validation accuracy 0.8968.

6.3.5 Training Experiment of ResNet50

Table 6.24: Training experiment of ResNet50

ResNet50		Without Preprocess
Error Function	Categorical Cross-Entropy	Loss: 0.1827
Pre-trained Weights	ImageNet	
Optimizer	Adam	Accuracy: 0.9723
Batch Size	32	
Epoch	23	Validation Loss: 0.3728
Input Size	224x224	
Learning Rate	0.01	Validation Accuracy: 0.9235
Layer	50	

In this pretrained model, ResNet50 on ImageNet dataset have used as the classifier. Most of the layers have untouched. A new prediction layer replaces only the final layer with three-way softmax activation. In this training experiment of table 6.24, we have used Adam as an Optimizer with a learning rate of 0.01 and 'Categorical Crossentropy' as loss function. ResNet50 has 50 layers and the input size is 224x224. Here, the epoch is 23 and the batch size is 32. From the table 6.24 above, we can see that if we trained ResNet50 in the above configuration without preprocessing on the dataset, then we get training accuracy 0.9723 and validation accuracy 0.9235.

6.3.6 Training Experiment of Inception-ResNet-V2

Table 6.25: Training experiment of Inception-ResNet-V2

Inception-ResNet-V2		Without Preprocess
Error Function	Categorical Cross-Entropy	Loss: 0.2351
Pre-trained Weights	ImageNet	
Optimizer	Adam	Accuracy: 0.9627
Batch Size	32	
Epoch	31	Validation Loss: 0.3528
Input Size	299x299	
Learning Rate	0.01	Validation Accuracy: 0.9092
Layer	164	

In this pretrained model, Inception-ResNet-V2 on ImageNet dataset have used as the classifier. Most of the layers have untouched. A new prediction layer replaces only the final layer with three-way softmax activation. In this training experiment of table 6.25, we have used Adam as an Optimizer with a learning rate of 0.01 and 'Categorical Crossentropy' as

loss function. Inception-ResNet-V2 has 164 layers and the input size is 299x299. Here, the epoch is 31 and the batch size is 32. From the table 6.25 above, we can see that if we trained Inception-ResNet-V2 in the above configuration without preprocessing on the dataset, then we get training accuracy 0.9627 and validation accuracy 0.9092.

6.3.7 Training Experiment of DenseNet121

Table 6.26: Training experiment of DenseNet121

DenseNet121		**Without Preprocess**
Error Function	Categorical Cross-Entropy	Loss: 0.1929
Pre-trained Weights	ImageNet	
Optimizer	Adam	Accuracy: 0.9351
Batch Size	32	
Epoch	17	Validation Loss: 0.3923
Input Size	224x224	
Learning Rate	0.01	Validation Accuracy: 0.8927
Layer	121	

In this pretrained model, DenseNet121 on ImageNet dataset have used as the classifier. Most of the layers have untouched. A new prediction layer replaces only the final layer with three-way softmax activation. In this training experiment of table 6.26, we have used Adam as an Optimizer with a learning rate of 0.01 and 'Categorical Crossentropy' as loss function. DenseNet121 has 121 layers and the input size is 224x224. Here, the epoch is 17 and the batch size is 32. From the table 6.26 above, we can see that if we trained DenseNet121 in the above configuration without preprocessing on the dataset, then we get training accuracy 0.9351 and validation accuracy 0.8927.

In the training experiment of different architectures of transfer learning, it can be seen that the VGG16 is more efficient and computationally less costly and less time consuming than other pre-trained architectures.

6.3.8 Comparison between Different Approaches of the Dataset of CNN

An analysis of the table and graph above reveals several essential aspects which is given below-

- In this project, a total of eleven CNN-centric ablation experiments have been performed, of which CNN's Setup-9 has given better results in three approaches based on the validation accuracy than other setups.

Table 6.27: Comparison between different transforms of the dataset of CNN

Architecture's Name	Dataset	Accuracy	Precision	F1 Score	Recall
CNN (Setup-9)	Without Preprocess (normal)	82.17%	0.8046	0.8040	0.8508
	With CLAHE	79.30%	0.7778	0.7748	0.8191
	With Power Law	79.04%	0.7732	0.7697	0.8106

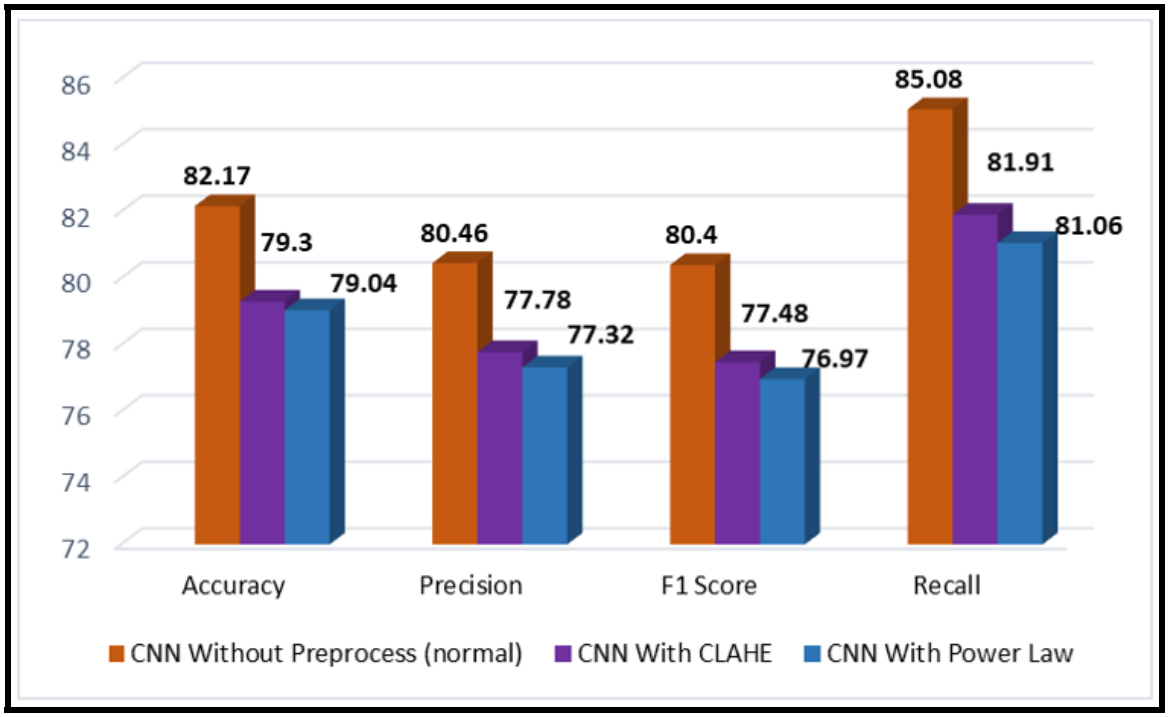


Figure 6.10: Comparison between different transforms of the dataset of CNN

- But we have picked the setup-9 of custom CNN trained without preprocessing of the dataset where it gives higher validation accuracy than other approaches. Finally, we have evaluated the testing results of setup-9 by the evaluation metrics.
- In the case of deep learning, it is not necessary to preprocess the dataset. We have good accuracy in custom CNN trained without any preprocessing of the dataset.
- It can be seen from the table 6.27 above that not only the result of testing accuracy is good by trained CNN (setup-9), but also the result of F1 score, precision, Recall is better than CLAHE and Power-law transform.

6.3.9 Comparison between Different Architectures of Transfer Learning

Table 6.28: Comparison between different Architectures of Transfer Learning

Architecture's Name	Dataset	Accuracy	Precision	F1 Score	Recall
VGG16	**Without Preprocess (normal)**	94.45%	0.9271	0.9345	0.9449
Inception V3	**Without Preprocess (normal)**	88.65%	0.8598	0.8671	0.9088
Xception	**Without Preprocess (normal)**	89.40%	0.8654	0.8739	0.9128
ResNet50	**Without Preprocess (normal)**	90.45%	0.8791	0.8908	0.9089
Inception-ResNet-V2	**Without Preprocess (normal)**	90.24%	0.8734	0.8838	0.9217
DenseNet121	**Without Preprocess (normal)**	88.73%	0.8607	0.8681	0.9099

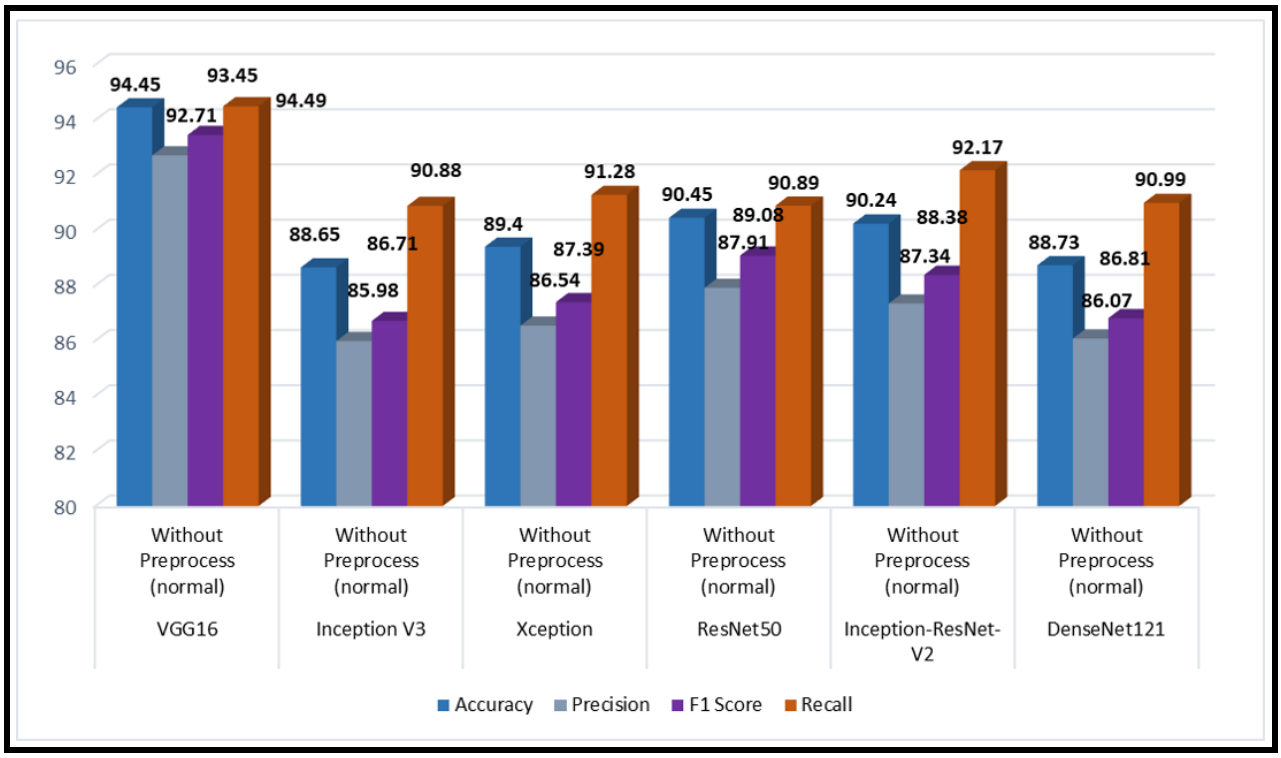


Figure 6.11: Comparison between different Architectures of Transfer Learning

Analyzing the graph and table 6.28 above, it can be seen that VGG16 gave better results than other architectures in the testing evaluation. Every architecture has been trained without preprocessing. Not only the result of testing accuracy is good by trained VGG16, but also the result of F1 score, precision, Recall is better than other architectures.

6.3.10 Comparison between CNN (setup-9) and Transfer Learning (VGG16)

Table 6.29: Comparison between CNN (setup-9) and Transfer Learning (VGG16)

Architecture's Name	Dataset	Accuracy	Precision	F1 Score	Recall
CNN (Setup-9)	**Without Preprocess (normal)**	82.17	80.46	80.40	85.08
VGG16	**Without Preprocess (normal)**	94.45	92.71	93.45	94.49

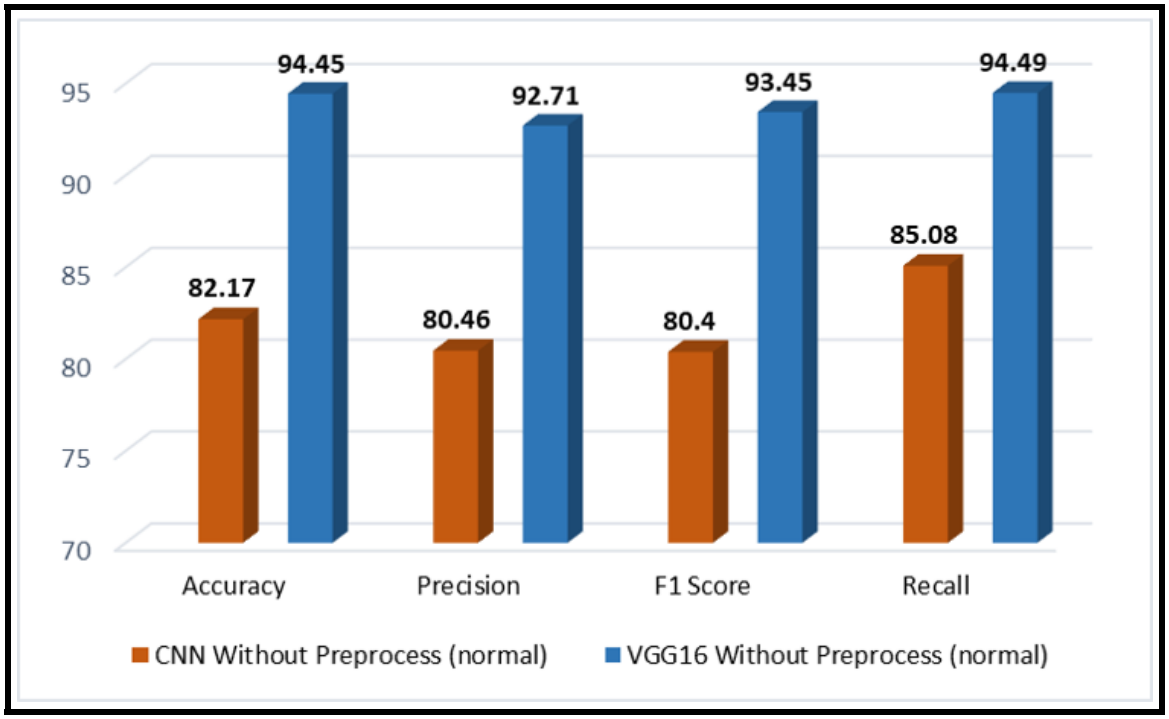


Figure 6.12: Comparison between CNN (setup-9) and Transfer Learning (VGG16)

An analysis of the table 6.29 and graph above reveals several essential aspects which is given below-

- In the case of deep learning, it is not necessary to preprocess the dataset. We have good accuracy in both CNN and VGG16 without preprocessing.
- Here, it can be seen from the table above that not only the result of testing accuracy is good by trained pre-trained model (VGG16) but also the result of F1 score, precision, Recall is better than CNN.

We can explore the above table and graph through the ablation experiment and say that we have got the satisfactory testing accuracy using VGG16 pre-trained model.

Chapter 7

Conclusion and Future Work

7.1 Limitation

Deep learning approach keeps the high acceleration because it affects the acceleration of GPU based computing power and non-linearity which allow deeper networks for better utilization. Nevertheless, this method has some disadvantages. From this system, we could get better accuracy or more efficient architecture, but due to some limitations, it was not possible. Some of the limitations are discussed below.

1. The first thing to notice when training a model with deep learning is that it is very efficient but also very time-consuming and computationally expensive.

2. Deep Learning is very hardware dependent and takes a lot of time to prepare in a larger dataset as it is not possible to get a good result without tuning hyperparameter a reasonable amount of time. Therefore, a powerful Quadro GPU is needed to reduce this time-consuming problem, which is economically very expensive.

3. In our dataset, there are lots of images with different high resolution. However, due to a lack of hardware configuration, we could not train the original x-ray image of the chest diseases. Otherwise, we would have got better accuracy.

4. Due to poor GPU configuration, we were unable to evaluate our testing data at once. Otherwise, the accuracy would have been further improved.

5. Accuracy would have been better if the containing image sample of the dataset was larger. So these are the main limitations of our system.

7.2 Future Work

The general architectures of the 'chest disease detection' system are proposed to resolve the system limitations and give satisfactory results of chest diseases in a pretty precise time. Thus, there are numerous suggested aspects to the system for future work that can be improved by adding or modifying them, which are discussed below.

1. The system can be improved to make a Cloud-Based Automated Clinical Decision Support System that mainly consists of two parts that the server-side and the client-side, and both will be linked together over a shared network. The server-side contains a deep learning model, which implements the training and classification task. The client-side is an application where users can upload their chest x-ray images and passes them to the server. The server will train and classify those images and tell them whether there is chest disease in the image or not. If so, what kind of chest disease is there?

2. This system can be developed to create a multi-label detection system that can classify multiple diseases at a time.

3. Increasing the training dataset used for training the model can make the system more robust, and then the system will be capable to detect chest diseases more accurately.

4. Chest disease datasets can be collected from several reputed hospitals in Bangladesh and trained by various deep learning algorithms. So, the automated system of the detection process is further improved in Bangladesh.

5. The system can be enhanced by training with unsupervised learning, which has no specific data label in the testing data set. In that system, users will be able to know the corresponding output using any chest x-ray image. Although there is no specific testing data, so the system will be quite complex to build. However, if the system is developed in this way, it will be much more realistic and efficient to detect any chest x-ray diseases.

7.3 Conclusion

The diagnosis of chest diseases in X-Ray images is one of the critical research challenges from the past decades. Several machine learning, feature learning, and pattern analysis techniques are introduced to address this issue. But most of the techniques have failed to provide good results in practical aspects. Therefore, there is a need for a system for the accurate interpretation of radiographic images. In this paper, a deep learning-based approach is proposed to classify diseases from chest X-ray images using Deep Convolutional Neural Networks (DCNN) and transfer learning (VGG16, DenseNet121, Inception V3, Inception-ResNet-V2, and ResNet50) using without preprocessing and with preprocessing (CLAHE, Power Law). Here, We have done ablation experiments with CNN through eleven setups and adopted the transfer learning approach by used the pre-trained architectures trained on the ImageNet dataset to extract features. These features were passed to the classifiers of respective models, and the output was collected from individual architectures. The experiments were conducted using a chest x-ray 14 dataset, and the CXI dataset, which contains 34439 images, and performances were evaluated using various performance metrics. Furthermore, the obtained results show that the CNN (Setup-9) gave high performance (accuracy more than 82%) and VGG16 gave the testing accuracy of 94.45%. On the contrary, none of the proposed setups (CNN) has been susceptible to give satisfactory results with CLAHE and Power law transform. In this context, we can say that image processing is not necessary for deep learning, and CNN doesn't work well for any setup. It works excellently for specific settings (Setup-9) of CNN and we got the satisfactory result from VGG16.

References

[1] N. Ansari, A. R. Faizabadi, S. Motakabber, and M. I. Ibrahimy, "Effective pneumonia detection using res net based transfer learning," 2020.

[2] "The-global-impact-of-respiratory-disease." https://www.who.int/gard/publications/The_Global_Impact_of_Respiratory_Disease.pdf. Accessed: 2020-5-28.

[3] K. E. Asnaoui, Y. Chawki, and A. Idri, "Automated methods for detection and classification pneumonia based on x-ray images using deep learning," *arXiv preprint arXiv:2003.14363*, 2020.

[4] A. Chaudhary, A. Hazra, and P. Chaudhary, "Diagnosis of chest diseases in x-ray images using deep convolutional neural network," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–6, IEEE, 2019.

[5] "What is a chest x-ray (chest radiography)?." https://www.radiologyinfo.org/en/info.cfm?pg=chestrad. Accessed: 2020-5-28.

[6] "Lung diseases overview." https://www.webmd.com/lung/lung-diseases-overview. Accessed: 2020-5-28.

[7] V. Chouhan, S. K. Singh, A. Khamparia, D. Gupta, P. Tiwari, C. Moreira, R. Damaševičius, and V. H. C. de Albuquerque, "A novel transfer learning based approach for pneumonia detection in chest x-ray images," *Applied Sciences*, vol. 10, no. 2, p. 559, 2020.

[8] S. Bharati and P. Podder, "Disease detection from lung x-ray images based on hybrid deep learning," *arXiv preprint arXiv:2003.00682*, 2020.

[9] A. Rasheed, M. S. Younis, M. Bilal, and M. Rasheed, "Classification of chest diseases using wavelet transforms and transfer learning," *arXiv preprint arXiv:2002.00625*, 2020.

[10] S. Rajaraman, S. Candemir, G. Thoma, and S. Antani, "Visualizing and explaining deep learning predictions for pneumonia detection in pediatric chest radiographs," in *Medical Imaging 2019: Computer-Aided Diagnosis*, vol. 10950, p. 109500S, International Society for Optics and Photonics, 2019.

[11] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2097–2106, 2017.

[12] O. Stephen, M. Sain, U. J. Maduh, and D.-U. Jeong, "An efficient deep learning approach to pneumonia classification in healthcare," *Journal of healthcare engineering*, vol. 2019, 2019.

[13] S. R. Islam, S. P. Maity, A. K. Ray, and M. Mandal, "Automatic detection of pneumonia on compressed sensing images using deep learning," in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pp. 1–4, IEEE, 2019.

[14] R. Soni, R. Singh, S. Shah, and S. Malik, "Chest disease detection from human x-ray scans using deep learning," tech. rep., EasyChair, 2020.

[15] I. M. Baltruschat, H. Nickisch, M. Grass, T. Knopp, and A. Saalbach, "Comparison of deep learning approaches for multi-label chest x-ray classification," *Scientific reports*, vol. 9, no. 1, pp. 1–10, 2019.

[16] R. H. Abiyev and M. K. S. Ma'aitah, "Deep convolutional neural networks for chest diseases detection," *Journal of healthcare engineering*, vol. 2018, 2018.

[17] D. Varshni, K. Thakral, L. Agarwal, R. Nijhawan, and A. Mittal, "Pneumonia detection using cnn based feature extraction," in *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pp. 1–7, IEEE, 2019.

[18] H. Zhu, W. Sun, M. Wu, G. Guan, and Y. Guan, "Pre-processing of x-ray medical image based on improved temporal recursive self-adaptive filter," in *2008 The 9th International Conference for Young Computer Scientists*, pp. 758–763, IEEE, 2008.

[19] M. T. Islam, M. A. Aowal, A. T. Minhaz, and K. Ashraf, "Abnormality detection and localization in chest x-rays using deep convolutional neural networks," *arXiv preprint arXiv:1705.09850*, 2017.

[20] Y. Sun, Y. Liu, G. Wang, and H. Zhang, "Deep learning for plant identification in natural environment," *Computational intelligence and neuroscience*, vol. 2017, 2017.

[21] Y. Bar, I. Diamant, L. Wolf, S. Lieberman, E. Konen, and H. Greenspan, "Chest pathology detection using deep learning with non-medical training," in *2015 IEEE 12th international symposium on biomedical imaging (ISBI)*, pp. 294–297, IEEE, 2015.

[22] Y. Bar, I. Diamant, L. Wolf, and H. Greenspan, "Deep learning with non-medical training used for chest pathology identification," in *Medical Imaging 2015: Computer-Aided Diagnosis*, vol. 9414, p. 94140V, International Society for Optics and Photonics, 2015.

[23] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, *et al.*, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," *arXiv preprint arXiv:1711.05225*, 2017.

[24] L. C. Ebert, J. Heimer, W. Schweitzer, T. Sieberth, A. Leipner, M. Thali, and G. Ampanozi, "Automatic detection of hemorrhagic pericardial effusion on pmct using deep learning-a feasibility study," *Forensic Science, Medicine and Pathology*, vol. 13, no. 4, pp. 426–431, 2017.

[25] Y. Wei, W. Xia, M. Lin, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan, "Hcp: A flexible cnn framework for multi-label image classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 9, pp. 1901–1907, 2015.

[26] R. Mor, I. Kushnir, J.-J. Meyer, J. Ekstein, and I. Ben-Dov, "Breath sound distribution images of patients with pneumonia and pleural effusion," *Respiratory care*, vol. 52, no. 12, pp. 1753–1760, 2007.

[27] A. Bashir, Z. A. Mustafa, I. Abdelhameid, and R. Ibrahem, "Detection of malaria parasites using digital image processing," in *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*, pp. 1–5, IEEE, 2017.

[28] H. A. Nugroho, S. A. Akbar, and E. E. H. Murhandarwati, "Feature extraction and classification for detection malaria parasites in thin blood smear," in *2015 2nd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, pp. 197–201, IEEE, 2015.

[29] S. N. Chavan and A. M. Sutkar, "Malaria disease identification and analysis using image processing," *International Journal of Computing and Technology (IJCAT)*, vol. 1, no. 6, 2014.

[30] Z. Liang, A. Powell, I. Ersoy, M. Poostchi, K. Silamut, K. Palaniappan, P. Guo, M. A. Hossain, A. Sameer, R. J. Maude, *et al.*, "Cnn-based image analysis for malaria diagnosis," in *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 493–496, IEEE, 2016.

[31] W. D. Pan, Y. Dong, and D. Wu, "Classification of malaria-infected cells using deep convolutional neural networks," *Machine Learning: Advanced Techniques and Emerging Applications*, p. 159, 2018.

[32] A. Rahman, H. Zunair, M. S. Rahman, J. Q. Yuki, S. Biswas, M. A. Alam, N. B. Alam, and M. Mahdy, "Improving malaria parasite detection from red blood cell using deep convolutional neural networks," *arXiv preprint arXiv:1907.10418*, 2019.

[33] "Pneumonia." https://www.mayoclinic.org/diseases-conditions/pneumonia/symptoms-causes/syc-20354204. Accessed: 2020-5-28.

[34] "What are the symptoms of pneumonia?." https://www.hopkinsmedicine.org/health/conditions-and-diseases/pneumonia. Accessed: 2020-5-28.

[35] "Pleural effusion (fluid in the pleural space)." https://www.medicinenet.com/pleural_effusion_fluid_in_the_chest_or_on_lung/article.htm. Accessed: 2020-5-28.

[36] "What is clahe?." https://imagemagick.org/script/clahe.php. Accessed: 2020-6-1.

[37] "Power law (gamma) transformations." https://theailearner.com/2019/01/26/power-law-gamma-transformations/. Accessed: 2020-6-1.

[38] "Deep learning." https://www.investopedia.com/terms/d/deep-learning.asp. Accessed: 2020-6-1.

[39] "Supervised vs unsupervised learning: Key differences." https://www.guru99.com/supervised-vs-unsupervised-learning.html. Accessed: 2020-6-1.

[40] "Understanding of convolutional neural network (cnn)." t.ly/Bd9e. Accessed: 2020-6-1.

[41] "A gentle introduction to pooling layers for convolutional neural networks." https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/. Accessed: 2020-6-1.

[42] "Understanding neural networks. from neuron to rnn, cnn, and deep learning." t.ly/yDFK. Accessed: 2020-6-1.

[43] "Adam — latest trends in deep learning optimization." t.ly/guvK. Accessed: 2020-6-1.

[44] "Loss functions explained." t.ly/kbPw. Accessed: 2020-6-1.

[45] "logistic-regression-as-a-neural-network." https://snaildove.github.io/2018/02/02/02_neural-networks-basics/. Accessed: 2020-6-1.

[46] "Why initialize weights." t.ly/dYzM. Accessed: 2020-6-1.

[47] "What is transfer learning?." t.ly/isNB. Accessed: 2020-6-1.

[48] "What is feature extraction?." https://deepai.org/machine-learning-glossary-and-terms/feature-extraction. Accessed: 2020-6-1.

[49] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, *et al.*, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.

Generated using Undegraduate Thesis LaTeX Template, Version 1.4. Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh.

This thesis was generated on Friday 22nd April, 2022 at 6:24am.