A Project Report on

PNEUMONIA DETECTION WITH MINIMUM SUPERVISION

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology

In

Computer Science and Engineering

Submitted by

THADISINA PRUDHVI REDDY

(19H55A0521)

VADLOORI RAJU

(19H55A0522)

RAVULAKORU VENKATESH

(19H55A0523)

Under the esteemed guidance of
MR.S.SIVA SKANDHA
Associate Professor

Department of Computer Science and Engineering

CMR COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution under UGC & JNTUH, Approved by AICTE, Permanently Affiliated to JNTUH, Accredited by NBA.)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2018- 2022

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This to certify that the Major Project report entitled "**PNEUMONIA DETECTION WITH MINIMUM SUPERVISION**" being submitted by **T.PRUDHVI REDDY (19H55A0521), V.RAJU (19H55A0522), R.VENKATESH (19H55A0523)** in partial fulfillment for the award of **Bachelor of Technology** in **Computer Science and Engineering** is a record of bonafide work carried out his/her under my guidance and supervision.

The results embodies in the project report have not been submitted to any other University or Institute for the award of any Degree.

Mr.S.Siva Skandha
Associate professor, Dept of CSE
Project Guide

Dr. K .Vijaya kumar
Professor and HOD
Dept. of CSE

ACKNOWLEDGMENT

With great pleasure we want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

We grateful to **Mr.S.SivaSkandha**, Associate professor Dept of Computer Science and Engineering for his valuable suggestions and guidance during the execution of this project work.

We would like to thank **Dr. K. Vijaya Kumar**, professor & Head of the Department of Computer Science and Engineering, for his moral support throughout the period of my study in CMRCET.

We highly indebted to **Major Dr. V.A. Narayana**, Principal CMRCET for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the Teaching & Non- teaching staff of Department of Computer Science and Engineering for their co-operation

Finally, we express our sincere thanks to **Mr. Ch. Gopal Reddy**, Secretary, CMR Group of Institutions, for his continuous care. I sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

T.PRUDHVI REDDY (19H55A0521)

V.RAJU (19H55A0522)

R.VENKATESH (19H55A0523)

TABLE OF CONTENTS

List of Figures

ABSTRACT

Pneumonia is a life-threatening infectious disease affecting one or both lungs in humans commonly caused by bacteria called Streptococcus pneumoniae. One in three deaths in India is caused due to pneumonia as reported by World Health Organization (WHO). Chest X-Rays which are used to diagnose pneumonia need expert radiotherapists for evaluation. Thus, developing an automatic system for detecting pneumonia would be beneficial for treating the disease without any delay particularly in remote areas. Our project wants to show a different approach in detection of diseases using deep learning algorithms. In case of our project, we take X-rays of chest as input and we are going to predict the disease whether it is present or not and if it's present how much is the severity of the disease. Normally the dataset used in building the model is Chest X-ray Dataset which is a labelled dataset but this dataset has already been used by many models which are out there and this data set is huge (which consists of nearly 1billion images) so we wanted to train our model by using chest x-ray dataset without labelled. So, by this we will first create our pre-trained model and then again, we are going to train this pre-trained model with medical image datasets without label which can be collected from other source. The learning method which we use to build this model is Self-Supervised learning.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Pneumonia is an inflammatory condition of the lung primarily affecting the small air sacs known as alveoli. Symptoms typically include some combination of productive or dry cough, chest pain, fever, and difficulty breathing. The severity of the condition is variable. Chest radiography has proven to be one of the most time- and cost-effective tools for pneumonia diagnosis. The most frequent limitation for successful pattern recognition in the biomedical imaging domain has always been a scarcity of data. To date, all previous approaches have relied on backbone networks pretrained on ImageNet. Self-supervised neural networks provide unprecedented performance in computer vision tasks.

Various approaches for self-supervised model training exist. The main paradigm has shifted towards instance discriminative models, where similar contrastive learning (SimCLR) momentum contrast for unsupervised visual representation learning (MoCo)and bootstrap your own latent architecture (BYOL) have demonstrated as-yet-untapped potential. The representations learned by these architecture sare on par with those of their supervised counterparts.

Self-supervised learning (SSL) is a method of machine learning. It learns from unlabeled sample data. It can be regarded as an intermediate form between supervised and unsupervised learning. It is based on an artificial neural network. Results shows that representations learned in a self-supervised fashion on smaller datasets with semantically closer domains are more beneficial than supervised pretraining on large but semantically very different datasets such as ImageNet. Considering that self-supervised contrastive learning for visual representations is a very new topic, this approach has huge potential. Later methodological improvements may further boost the performance.

1.2 MOTIVATION

In recent times many of the people suffered from the pneumonia to detect the pneumonia the person should take an X-Ray and that should be examine by radiologist that takes more time and expensive process instead of that building an automated detection system to save doctors precious time and patients money.

1.3 PROBLEM DEFINATION

Pneumonia is an infection that inflames the air sacs in one or both lungs. The air sacs may fill with fluid or pus (purulent material), causing cough with phlegm or pus, fever, chills, and difficulty breathing. A variety of organisms, including bacteria, viruses and fungi, can cause pneumonia. Normally to detect this disease Chest X-Rays which are used to diagnose pneumonia need expert radiotherapists for evaluation This project used to detect the pneumonia using chest X-ray. The Self-Supervised learning by using SIMCLR scheme to detect the X-ray images and make the model learn about the images and predict the disease.

1.4 OBJECTIVES

- To detect the pneumonia with minimum supervision by using self-supervised learning by using SIMCLR framework.
- By using minimum unlabeled dataset to detect pneumonia and calculate its severity.

CHAPTER 2

BACKGROUND WORK

2.1 DOMAIN INTRODUCTION

2.1.1 PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Figure 2.1 Python Logo

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this very effective.

2.1.2 GOOGLE COLAB

Google Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

Figure 2.2 Google Colab Logo

As a programmer, you can perform the following using Google Colab.

- Write and execute code in Python
- Document your code that supports mathematical equations
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets e.g., from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

2.2 LITERATURE SURVEY

2.2.1 PNEUMONIA DETECTION USING CNN

Paper title

Identifying pneumonia in chest X-rays: A deep learning approach using CNN model

Year of publication

2019

Authors

Amit Kumar Jaiswal, Prayag Tiwari, Sachin Kumar, Deepak Gupta d, Ashish Khanna, Joel J.P.C. Rodrigues.

Methodology:

In this study they have a created a model based on CNN in detecting pneumonia detection from chest X-ray images. In this model used for the multiclass classification instead of detection. The computation cost also burdens exponentially when dealing with large image.

Drawbacks

- This project is fails to detect the normal and virus x-ray images.
- The accuracy of the normal and virus is about 50% and 60% respectively.
- And it didn’t produce any output like whether the input has pneumonia or not it produce the output as confusion matrix.

2.2.2 PNEUMONIA DETECTION USING MASK R-CNN

Paper title

Identifying pneumonia in chest X-rays: A deep learning approach using MASK R-CNN model

Year of publication

2019

Authors

Amit Kumar Jaiswal, Prayag Tiwari, Sachin Kumar, Deepak Gupta d, Ashish Khanna, Joel J.P.C. Rodrigues.

Methodology:

In this study they have a created a model based on Mask-RCNN in detecting pneumonia symptoms from chest X-ray images. Mask-RCNN is a deep neural network developed to solve instance segmentation in particular. The computation cost also burdens exponentially when dealing with large image.

Drawbacks

With the usage of image augmentation, dropout and L2 regularization prevented the overfitting, but are obtained something weaker results on the training set with respect to the test.

2.2.3 PNEUMONIA DETECTION USING AN IMPROVED ALGORITHM BASED ON FASTER R-CNN

Paper title

Identifying pneumonia in chest X-rays: A deep learning approach using FASTER R-CNN model

Year of publication

2019

Authors

Kaiming He, Shaoqing Ren, Jian Sun Amit Kumar Jaiswal, Prayag Tiwari, Sachin Kumar, Deepak Gupta d, Ashish Khanna, Joel J.P.C. Rodrigues.

Methodology:

In this part, we introduce in detail our proposed DeepConv-DilatedNet method, including the data processing, the architecture of our network, and the effective enhancement effect of Soft-NMS.

Drawbacks

One drawback of Faster R-CNN is trained where all anchors in the min – batch of size 256, are extracted from a single image. Because all the samples from a single image may be correlated (i.e., Their features are similar), the network may take lot of time until reaching convergence.

2.3 EXISTING SYSTEMS

2.3.1 PNEUMONIA DETECTION USING CNN

So, the objective of this challenge is to determine whether a person suffers pneumonia or not. If yes, then determine whether it's caused by bacteria or viruses. Well, I think this project should be called **classification instead of detection.**

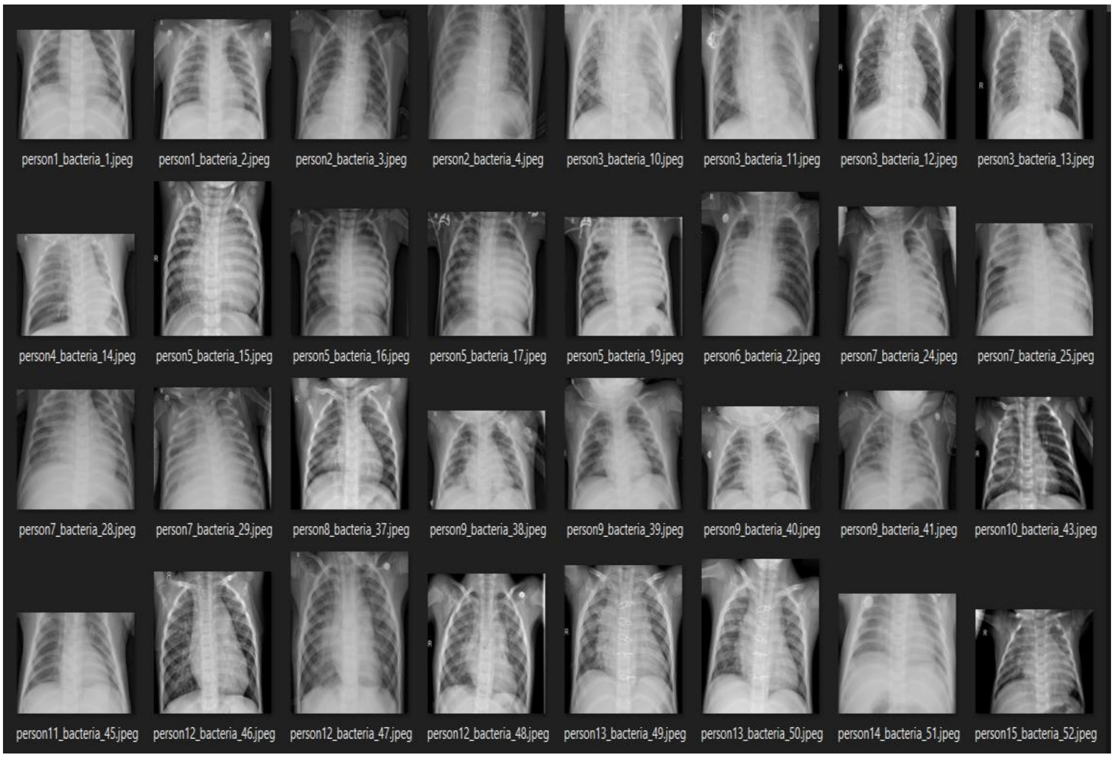


Figure 2.3 Chest X-ray images

Several x-ray images in the dataset used in this project. In other words, this task is going to be a multiclass classification problem where the label names are: *normal*, *virus*, and *bacteria*. In order to solve this problem. I will use CNN (Convolutional Neural Network), thanks to its excellent ability to perform image classification. Not only that, but here I also implement the image augmentation technique as an approach to improve model performance. By the way, here I obtained 80% accuracy on test data which is pretty impressive to me. The dataset used in this project can be downloaded from this Kaggle link. The size of the entire dataset itself is around 1 GB, so it might take a while to download. Or, we can also directly create a Kaggle Notebook and code the entire project there, so we don't even need to download anything. Next, if you explore the dataset folder, you will see that there are 3 subfolders, namely *train*, *test* and *val*. Well, I think those folder names are self-explanatory. In addition, the data in the *train* folder consists of 1341, 1345, and 2530 samples for *normal*, *virus* and *bacteria* class respectively.

Convolutional Neural Network (CNN):

Convolutional neural networks (CNN) are one of the most popular models used today. This neural network computational model uses a variation of multilayer perceptron's and contains one or more convolutional layers that can be either entirely connected or pooled. These convolutional layers create feature maps that record a region of image which is ultimately broken into rectangles and sent out for nonlinear processing.

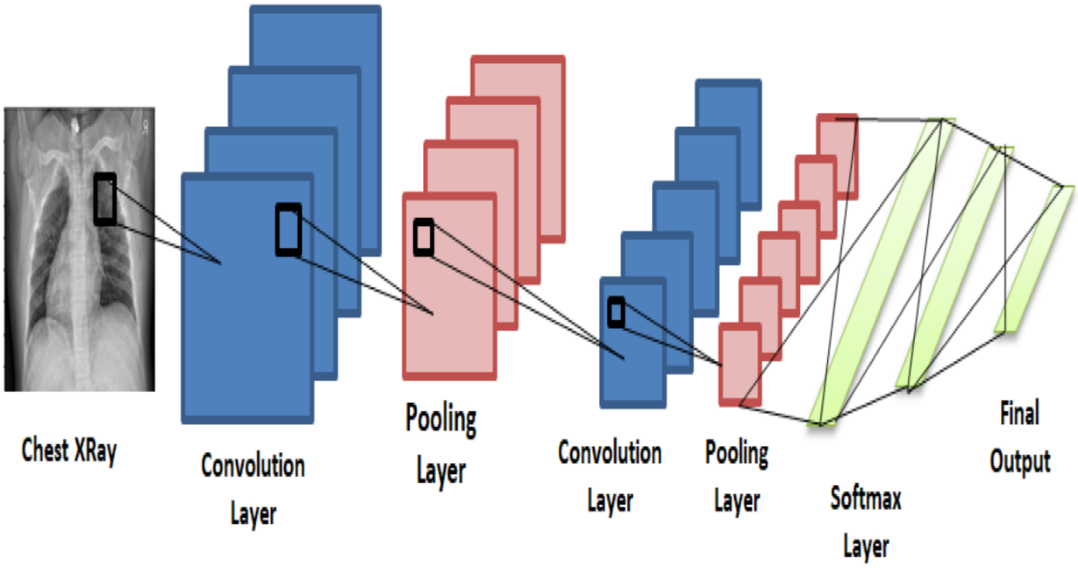


Figure 2.4 Convolution Neural Network Architecture

Advantages

- Very High accuracy in image recognition problems.
- Automatically detects the important features without any human supervision.
- Weight sharing.

Disadvantages

- CNN do not encode the position and orientation of object.
- Lack of ability to be spatially invariant to the input data.
- Lots of training data is required.

Drawbacks

- This project is fails to detect the normal and virus x-ray images.
- The accuracy of the normal and virus is about 50% and 60% respectively.
- And it didn't produce any output like whether the input has pneumonia or not it produce the output as confusion matrix.

Confusion matrix

The size of the entire dataset used in this project is 1 GB.

- This project is more accuracy in detecting lungs which is affected by the bacteria.
- Pneumonia detection on chest X-ray Accuracy - 90%

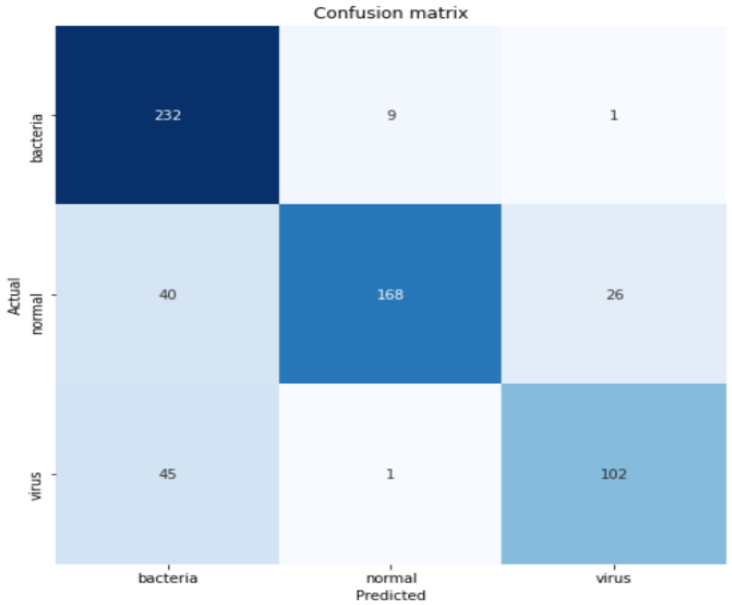


Figure 2.5 Confusion matrix

2.3.2 IDENTIFYING PNEUMONIA IN CHEST X-RAYS USING MASK-RCNN MODEL.

In this study they have a created a model based on Mask-RCNN in detecting pneumonia symptoms from chest x-ray images. Mask-RCNN is a deep neural network developed to solve instance segmentation in particular. The computation cost also burdens exponentially when dealing with large image.

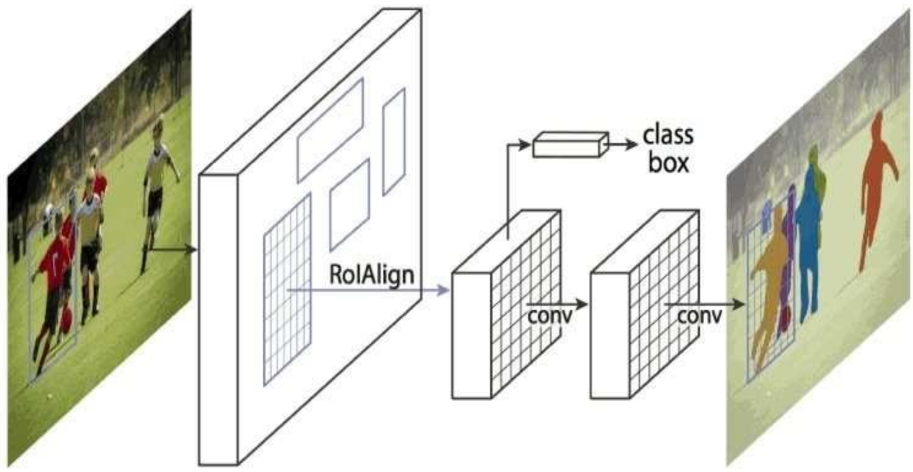


Figure 2.6 Mask R-CNN Architecture

Mask-RCNN approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. The method, called mask R-CNN. It extends faster R-CNN by adding a branch for predicting segmentation masks on each region of interest (ROI), in parallel with the existing branch for classification and bounding box regression. The mask branch is a small FCN applied to each ROI, predicting a segmentation mask in a pixel-to pixel manner. Mask R-CNN is simple to implement and train given the faster R-CNN framework. In this model based on Mask-RCNN is detecting pneumonia symptoms from chest x-ray images.

- The size of the entire dataset used in this is 25684 images.
- In this model based on mask-RCNN is detecting pneumonia symptoms from chest x-ray images.

MASK R-CNN:

Faster R-CNN and YOLO are good at detecting the objects in the input image. They also have very low detection time and can be used in real-time systems. However, there is a challenge that can't be dealt with object detection, the bounding box generated by YOLO and Faster R-CNN does not give any indication about the shape of the object.

Instance Segmentation:

This segmentation identifies each instance (occurrence of each object present in the image and colour them with different pixel). It basically works to classify each pixel location and generate the segmentation mask for each of the objects in the image. This approach gives more idea about the objects in the image because it preserves the safety of those objects while recognizing it.

Mask R-CNN architecture:

Mask R-CNN was proposed by *Kaiming He et al.* in 2017. It is very similar to Faster R-CNN except there is another layer to predict segmented. The stage of region proposal generation is same in both the architecture the second stage which works in parallel predict class, generate bounding box as well as outputs a binary mask for each RoI.

It comprises of –

- Backbone Network
- Region Proposal Network
- Mask Representation
- RoI Align

Advantages of Mask R-CNN

- **Simplicity:** Mask R-CNN is simple to train.

- **Performance:** Mask R-CNN outperforms all existing, single-model entries on every task.
- **Efficiency:** The method is very efficient and adds only a small overhead to Faster R-CNN.
- **Flexibility:** Mask R-CNN is easy to generalize to other tasks. For example, it is possible to use Mask R-CNN for human pose estimation in the same framework.

Drawbacks

With the usage of image augmentation, obtained something weaker results on the training set with respect to the test.

2.3.3 PNEUMONIA DETECTION USING AN IMPROVED ALGORITHM BASED ON FASTER R-CNN

ln this project they used a method based on DeepCon_DilatedNet of idefntifying and localizing pneumonia in chest X-ray (CXR) images. It is a two-stage detector R-CNN is adopted as the structure of a network Feature Pyramid Network FPN is integrated into residual neural network of dilated bottleneck so that deep features are expanded to preserve the deep feature and position information of the object. The images in training sets are augmented by flipping horizontally and vertically then it calculates the true positive rate and false positive rate then it forms ROC curve if the curve is to the upper left corner, the higher the true-positive rate obtained by the classifier in comparison to its false-positive rate, indicating that the classifier performs well.

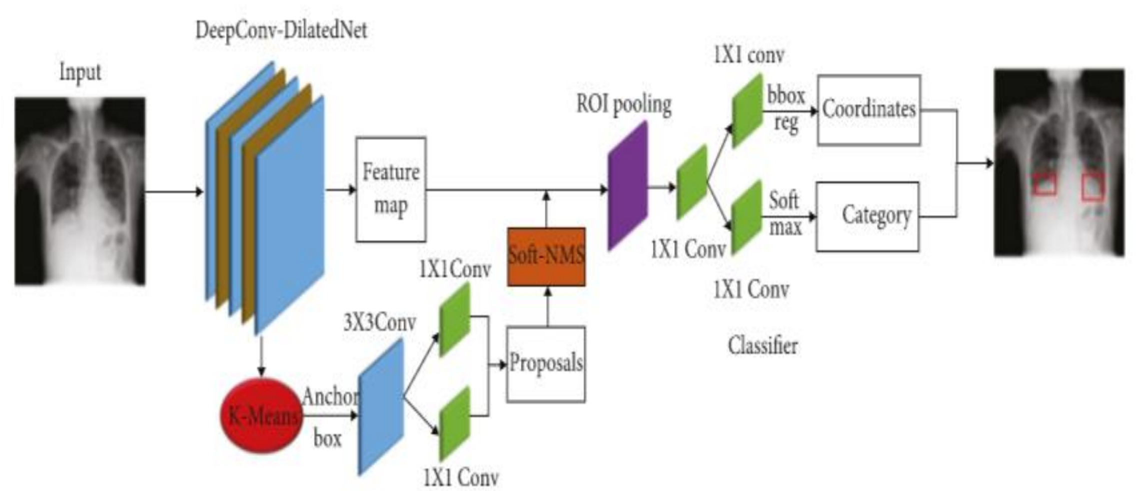


Figure 2.7 Faster R-CNN architecture

That began at the end of 2019 had a major impact on the world. It is still raging in many countries and has caused great losses to people's lives and property. In this paper, we present a method based on DeepConv-DilatedNet of identifying and localizing pneumonia in chest X-ray (CXR) images. Two-stage detector Faster R-CNN is adopted as the structure of a network. Feature Pyramid Network (FPN) is integrated into the residual neural network of a dilated bottleneck so that the deep features are expanded to preserve the deep feature and position information of the object. In the case of DeepConv-DilatedNet, the deconvolution network is used to restore high-level feature maps into its original size, and the target information is further retained. On the other hand, DeepConv-DilatedNet uses a popular fully convolution architecture with computation shared on the entire image. Then, Soft-NMS is used to screen boxes and ensure sample quality. Also, K-Means++ is used to generate anchor boxes to improve the localization accuracy. The algorithm obtained 39.23% Mean Average Precision (mAP) on the X-ray image dataset from the Radiological Society of North America (RSNA) and got 38.02% Mean Average Precision (mAP) on the ChestX-ray14 dataset, surpassing other detection algorithms. So, in this paper, an improved algorithm that can provide doctors with location information of pneumonia lesions is proposed.

Faster R-CNN:

Faster R-CNN was introduced in 2015 by *k He et al.* After the Fast R-CNN, the bottleneck of the architecture is selective search. Since it needs to generate *2000* proposals per image. It constitutes a major part of the training time of the whole architecture. In Faster R-CNN, it was replaced by the region proposal network. First of all, in this network, we passed the image into the backbone network. This backbone network generates a convolution feature map. These feature maps are then passed into the region proposal network. The region proposal network takes a feature map and generates the anchors (the centre of the sliding window with a unique size and scale). These anchors are then passed into the classification layer (which classifies that there is an object or not) and the regression layer (which localize the bounding box associated with an object).

Drawbacks:

One drawback of Faster R-CNN is trained where all anchors in the min – batch of size 256, are extracted from a single image. Because all the samples from a single image may be correlated (i.e., Their features are similar), the network may take lot of time until reaching convergence.

2.3.4 Comparison between CNN, Mask R-CNN and Faster R-CNN

CNN	FASTER R-CNN	MASK R-CNN
1)the first step of this model is data preprocessing (i.e., data resizing and labelling the data with multi class) 2)the accuracy of normal class is very less 3)this model is more accurate in detecting pneumonia effected by bacteria 4)it works well on below 1235 images 5)the accuracy of class bacteria is 92 and virus and normal is 60 and 50 respectively and the overall accuracy of this project is 66% 6)the output of this model is confusion matrix 7)this model is faster than the faster R-CNN and Mask R-CNN because it takes less size dataset and uses only CNN algorithm	1)the first step of this model is data preprocessing (i.e., image reshaping and flipping) 2)the next is contrast enhancement and brightness. This will increase the accuracy 3)the accuracy of this model will low when image contrast and brightness is very low 4)it is mainly used for large dataset 5)the output of this model is roc curve and image with pneumonia affected area 6)the accuracy of this model is 80% 7)it is slow because it takes large dataset as input and it uses more than 3 algorithms	1)the first step of this model is data preprocessing (i.e., image reshaping and labeling each image with three classes they are "lung opacity, normal and Not normal) 2)then next step will strong those class in one csv file. 3)the overall accuracy of this model is 63% 4)output of this model is csv file

TABLE 2.1 Comparison between CNN, Mask R-CNN and Faster R-CNN

CHAPTER 3

PROPOSED SYSTEM

3.1 PROSPOSED SYSTEM:

- In this system we use ImageNet Dataset which is unlabeled as first step to create a pre-trained model
- Using this pre-trained model, we again train the model with unlabeled medical image dataset by using SIMCLR framework.
- SIMCLR is a framework for contrastive learning of visual representations. It learns representation by augmented views of the same data which is a way of Self-Supervised Learning.
- After obtaining this model we test the model using limited medical image labeled dataset to check the accuracy of the model

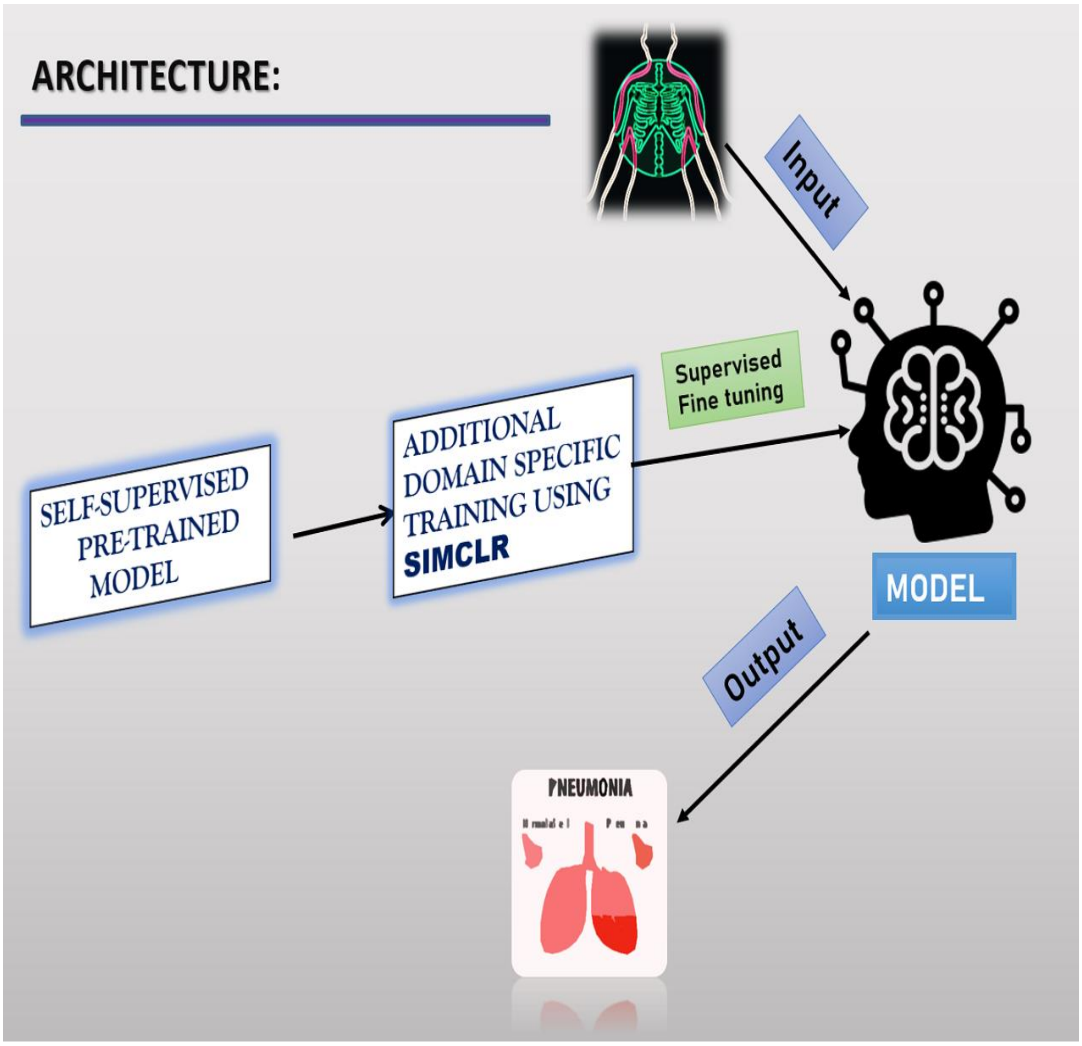


Figure 3.1 Proposed System Architecture

3.1.1 SELF-SUPERVISED LEARNING

Self-supervised learning (SSL) is a method of machine learning. It learns from unlabelled sample data. It can be regarded as an intermediate form between supervised and unsupervised learning. It is based on an artificial neural network. The neural network learns in two steps. First, the task is solved based on pseudo-labels which help to initialize the network weights. Second, the actual task is performed with supervised or unsupervised learning. Self-supervised learning has produced promising results in recent years and has found practical application in audio processing and is being used by Facebook and others for speech recognition The primary appeal of SSL is that training can occur with data of lower quality, rather than improving ultimate outcomes. Self-supervised learning more closely imitates the way humans learn to classify objects.

How Useful is Self-Supervised Learning?

The concept of self-supervised learning aims to address challenges in supervised learning when it comes to collecting, handling, cleaning, labelling, and analysing data. Developers who want to create an image classification algorithm, therefore, create supervised learning-capable systems to collect comprehensive data to get a representative sample. Apart from feeding the computer image datasets, developers need to classify the images before they can be used for training. The process is arduous and time-consuming compared with how humans approach learning. The human learning process is multifaceted. It involves both supervised and unsupervised learning processes. While we learn via experiments and curiosity, we also acquire knowledge better using fewer and simplified data. Even now, this remains a challenge for deep learning systems. While we have seen advances in learning-based AI systems that can break down speech, images, and text, performing complex tasks remains a challenge for these. That is what self-supervised learning is trying to address. In short, self-supervised learning allows AI systems to break down complex tasks into simple ones to arrive at a desired output despite the lack of labelled datasets.

How does Self-Supervision Work?

The basic concept of self-supervision relies on encoding an object successfully. A computer capable of self-supervision must know the different parts of any object so it can recognize it from any angle. Only then can it classify the thing correctly and provide context for analysis to come up with the desired output.

Real-World Applications of Self-Supervised Learning

According to Yann Lecun, a computer scientist known for his impressive work in the ML field, the closest we have to self-supervised learning systems are the so-called "Transformers." These are ML models that successfully use natural language processing (NLP) without the need for labelled datasets. They are capable of processing massive amounts of unstructured data and "transform" them into usable information for various purposes. The Transformers are behind Google's BERT and Meena, Open AI's GPT2, and Facebook's Roberta. But while they are better than their predecessors at answering questions, they still require much work to hone their understanding of human linguistics. Aside from processing unstructured data, the Transformers can also solve problems that involve manipulating symbols, which makes them useful in developing neural networks that carry out pattern recognition and statistical estimation. To date, the Transformers are vital in processing words and mathematical symbols easily. However, translating them into visual representations remains a challenge. Self-supervised learning is proving to be a significant component of AI and ML that would help experts resolve today's pressing challenges.

Types

Training data can be divided into positive examples and negative examples. Positive examples are those that match the target. For example, if you're learning to identify birds, the positive training data are those pictures that contain birds. Negative examples are those that do not.

Contrastive SSL

Contrastive SSL uses both positive and negative examples. Contrastive learning's loss function minimizes the distance between positive samples while maximizing the distance between negative samples

Non-contrastive SSL

Non-contrastive SSL uses only positive examples. Counterintuitively, NCSSL converges on a useful local minimum rather than reaching the expected identify function with zero loss. Effective NCSSL requires an extra predictor on the online side that does not back-propagate on the target side.

3.1.2 SIMCLR FRAMEWORK

SimCLR is a framework for contrastive learning of visual representations. It learns representations by maximizing agreement between differently augmented views of the same data example via a contrastive loss in the latent space. It consists of:

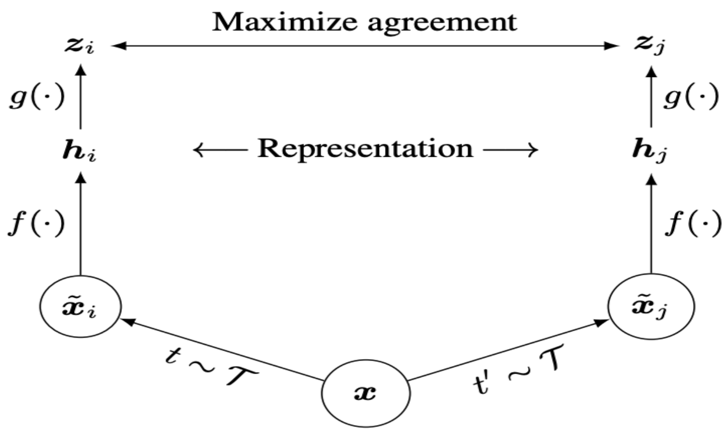


Figure 3.2 simCLR framework

A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are simple from the same family of argumentations (t ~ T and t' ~ T) applied to each data example to obtain two correlated views. A base encoder network f(.) and a projection head g(.) are trained to maximise agreement using a contrastive loss.

A stochastic data augmentation module that transforms any given data example randomly resulting in two correlated views of the same example, denoted x~i and x~j, which is considered a positive pair. SimCLR sequentially applies three simple augmentations: random cropping followed by resize back to the original size, random color distortions, and random Gaussian blur. The authors find random crop and color distortion is crucial to achieve good performance.

- A neural network base encoder f(·) that extracts representation vectors from augmented data examples. The framework allows various choices of the network architecture without any constraints. The authors opt for simplicity and adopt ResNet to obtain hi=f(sx~i)=ResNet(x~i) where hi∈Rd is the output after the average pooling layer.
- A small neural network projection head g(·) that maps representations to the space where contrastive loss is applied. Authors use a MLP with one hidden layer to obtain zi=g(hi)=W(2)σ(W(1)hi) where σ is a ReLU nonlinearity. The authors find it beneficial to define the contrastive loss on zi's rather than hi's.

- A contrastive loss function defined for a contrastive prediction task. Given a set {x~k} including a positive pair of examples x~i and x~j , the contrastive prediction task aims to identify x~j in {x~k}k≠i for a given x~i.

A minibatch of N examples is randomly sampled and the contrastive prediction task is defined on pairs of augmented examples derived from the minibatch, resulting in 2N data points. Negative examples are not sampled explicitly. Instead, given a positive pair, the other 2(N−1) augmented examples within a minibatch are treated as negative examples. A NT-Xent (the normalized temperature-scaled cross entropy loss) loss function is used (see components).

CHAPTER 4
DESIGNING

4.1 STEP WISE PROCEDURE IMPLEMENTATION

1. Data Set Collection.
2. Importing Libraries and dataset.
3. Pre-processing of Data.
4. Splitting the Data Set.
5. Object detection and instance segmentation.
6. Training the model with SIMCLR.

4.1.1 Data Set Collection

A dataset from Kaggle was used. It contained 6072 Chest X-Ray Images in jpeg format and the images were categorized into a) Train b) Test and c) Validate categories. These were further broken into 2 categories, Pneumonia and Normal.

The dataset covers both Normal X-Ray images and X-Ray images with Pneumonia. As one can notice, X-Ray of a healthy human (left) would show a clear, translucent image of the lungs. Meanwhile, X-Ray of the lungs of a person suffering from Pneumonia (right) would be opacified in a focal manner as shown in figure.

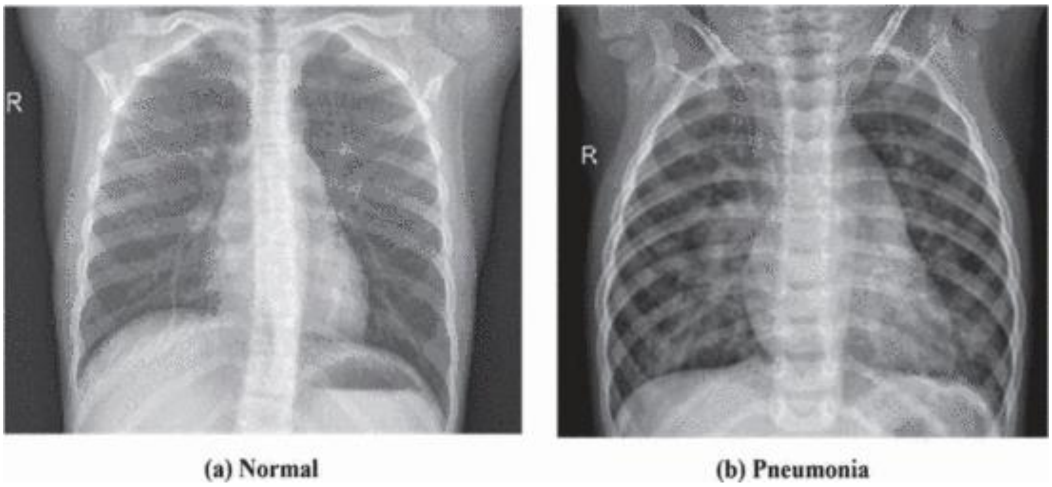


Figure 4.1 Normal and Pneumonia Chest X-ray images

Size of Dataset:

- No of samples: 6072
- No of pneumonia images: 4032
- No of normal images: 2030

Dataset Source: Kaggle website

https://www.kaggle.com/therealcyberlord/pneumonia-detection-using-deep-learning/data

4.1.2 Importing Libraries and dataset

To import libraries such as Pandas, NumPy, SimCLR, OpenCV, TensorFlow and Matplotlib are used and Chest X-Ray Dataset is imported into model which contains normal and pneumonia Chest X-Ray images.

4.1.3 Pre-processing of data

What are the data that are collected from Chest X-Ray dataset the images are taken and perform the operations like reshaping, resizing and removing unwanted and blank images from the dataset to get a predicted output? Which helpful to get accurate results.

4.1.4 Splitting the Data Set

We took Dataset called Chest X-Ray Dataset present in Kaggle website. The chest X-Ray Dataset contains two types of images called normal Chest X-Ray images and pneumonia Chest X-Ray images by using this chest X-ray images the model can predict the pneumonia and its severity.

4.1.5 Object detection and image segmentation

- **Object detection** is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images
- **Instance segmentation** is the task of detecting and delineating each distinct object of interest appearing in an image

4.1.6 Training the model with SIMCLR

- Self-supervised Formulation [Data Augmentation]
- Getting Representations [Base Encoder]
- Projection Head
- Tuning Model: [Bringing similar closer]

Data augmentation module: This module transforms any given data example stochastically generating two correlated views of the same example, denoted by *xi* and *xj*. Here the authors used three simple augmentations: *random cropping* followed by reseizing to the original size, *random color distortions*, and *random Gaussian blur*.

Base Encoder: ResNet-50 is used as the base neural network encoder for extracting representation vectors from the augmented data examples. The output of the last *average pooling layer* used for extracting representations.

Projection Head: A small neural network, MLP with one hidden layer, is used to map the representations from the base encoder to 128-dimensional latent space where contrastive loss is applied. *ReLU* is the activation function used in this projection head.

Contrastive Loss Function: Given a set of examples including a positive pair of examples (*xi* and *xj*), the contrastive prediction task aims to identify *xj* in the given set for a given *xi*.

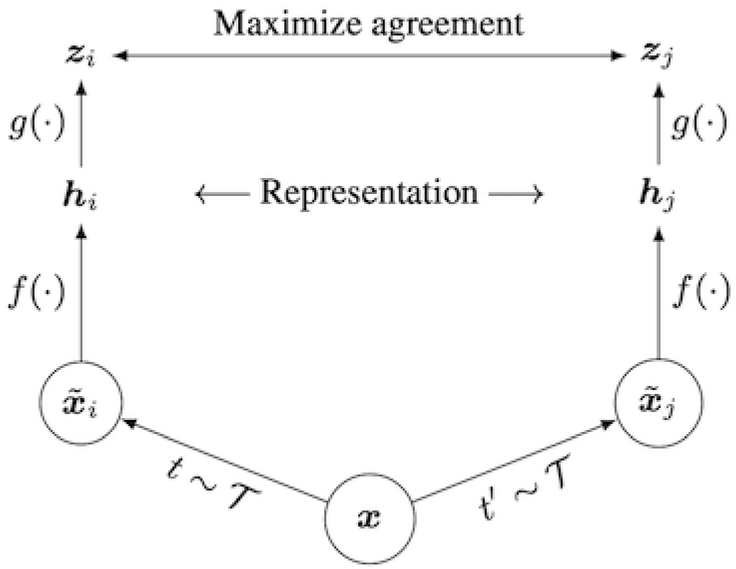


Figure 4.2 A simple framework for contrastive learning of visual representations.

Two separate data augmentation operators are simple from the same family of argumentations *(t ~ T and t' ~ T)* applied to each data example to obtain two correlated views. A base encoder network *f(.)* and a projection head *g(.)* are trained to maximise agreement using a contrastive loss. After training is completed, we through away the projection head *g(.)* and use encoder *f(.)* and representation **h** for downstream tasks.

CHAPTER 5

RESULTS AND PERFORMANCE ANALYSIS

5.1 PERFORMANCE ANALYSIS

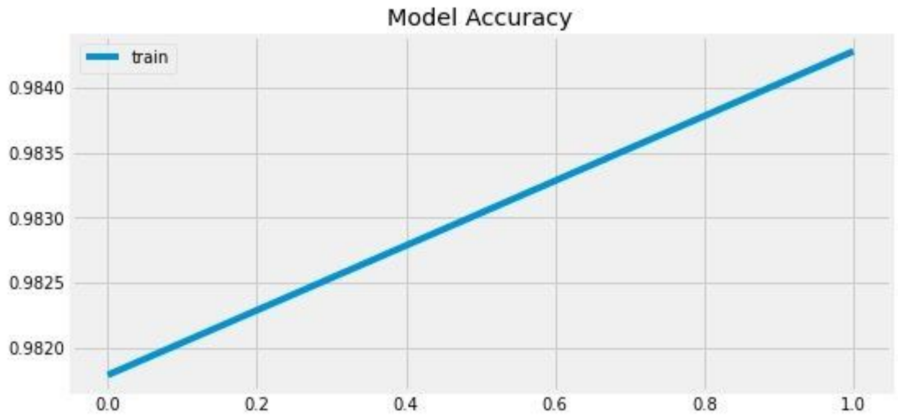
Prediction models are applied and the performance is determined using the Confusion Matrix (CM) method. The Confusion Matrix (CM) is used to analyse and determine the performance of the proposed loan prediction model (Shown in Table I and II). Figure 1 shows the CM parameters summarized from [13-14]. The interpretation in the CM is as follows:

- True Positive (TP), when both the actual and predicted values are positive (1)
- True Negative (TN), when both the actual and predicted values are negative (0)
- False Positive (FP), when the actual value is negative and the predicted value is positive(1)
- False Negative (FN), when the actual value is positive (1) and the predicted value is negative (0).

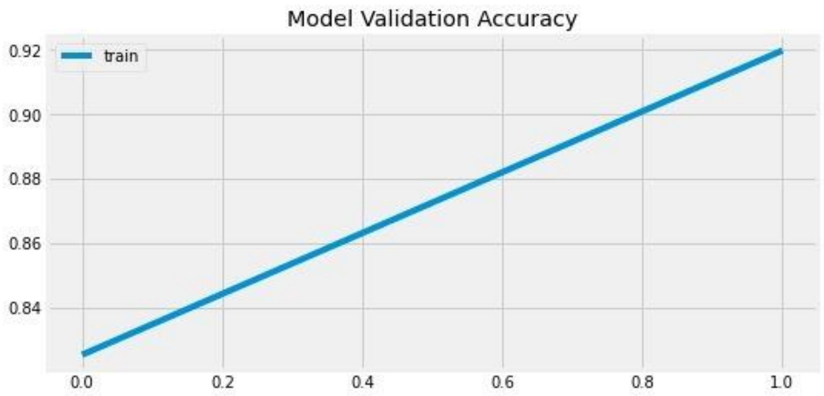
		Predicted Values	
Actual values		Neagative(0)	
	Neagative(0)	TN	FP
	Positive(1)	FN	TP

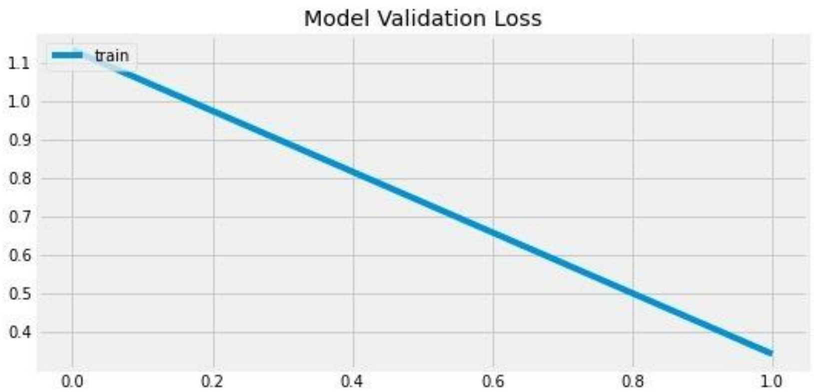
TABLE 5.1 CONFUSION MATRIX

```
plt.show()
```



+ Code + Text



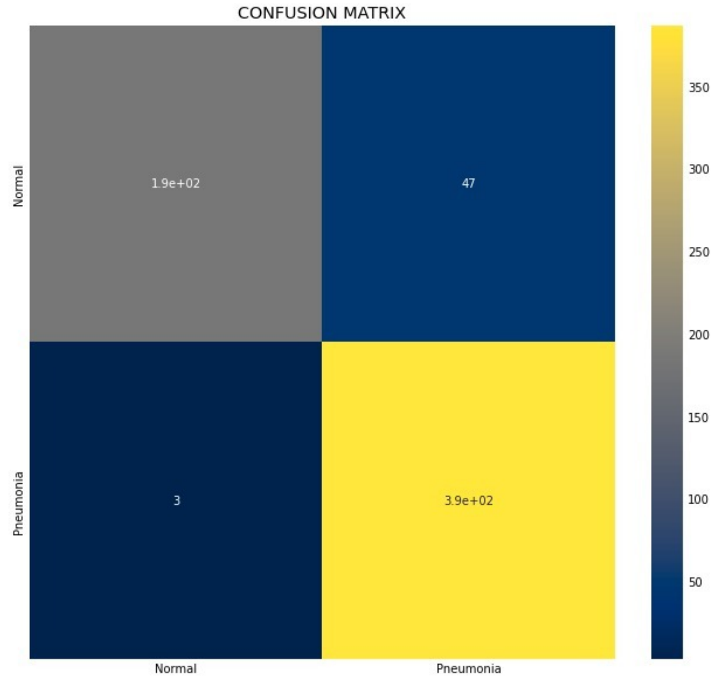


```
# classification report
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(test_data['target'], pred.flatten()))
```

```
              precision    recall  f1-score   support

           0       0.98      0.80      0.88       234
           1       0.89      0.99      0.94       390

    accuracy                           0.92       624
   macro avg       0.94      0.90      0.91       624
weighted avg       0.93      0.92      0.92       624
```

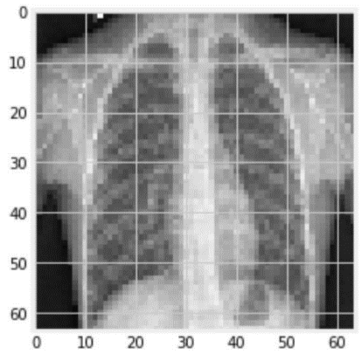
Accuracy:92%

+ Code + Text



Choose files No file chosen Upload widget is only available
Please rerun this cell to enable.

```
Saving IM-0001-0001.jpeg to IM-0001-0001 (1).jpeg
Normal
```



5.2 CODE FOR PNEUMONIA DETECTION

PNEMONIA DETECTION USING SELF SUPERVISED LEARNING

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import numpy as np
import pandas as pd
sns.set()
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
from tensorflow.keras.applications import DenseNet121, VGG19, ResNet50
from sklearn.metrics import precision_recall_curve, roc_curve, accuracy_score, confusion_matrix, precision_score, recall_score


import PIL.Image
import matplotlib.pyplot as mpimg
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array
from tensorflow.keras.preprocessing import image

from tqdm import tqdm
import warnings
warnings.filterwarnings("ignore")

from sklearn.utils import shuffle
```

```
train_df = pd.read_csv('./Chest-XRay-Dataset/Chest_xray_Metadata.csv')
train_df.shape
train_df.head(5)
```

		X_ray_image_name	Label	Dataset_type	Label_2_Virus_category	Label_1_Virus_category
0	0	IM-0128-0001.jpeg	Normal	TRAIN	NaN	NaN
1	1	IM-0127-0001.jpeg	Normal	TRAIN	NaN	NaN
2	2	IM-0125-0001.jpeg	Normal	TRAIN	NaN	NaN
3	3	IM-0122-0001.jpeg	Normal	TRAIN	NaN	NaN
4	4	IM-0119-0001.jpeg	Normal	TRAIN	NaN	NaN

```
train_df.dropna(how = 'all')
train_df.isnull().sum()
```

```
                          0
X_ray_image_name          0
Label                     0
Dataset_type              0
Label_2_Virus_category    5852
Label_1_Virus_category    1576
dtype: int64
```

```
train_df.fillna('unknown', inplace=True)
train_df.isnull().sum()
```

```
                          0
X_ray_image_name          0
Label                     0
Dataset_type              0
Label_2_Virus_category    0
Label_1_Virus_category    0
dtype: int64
```

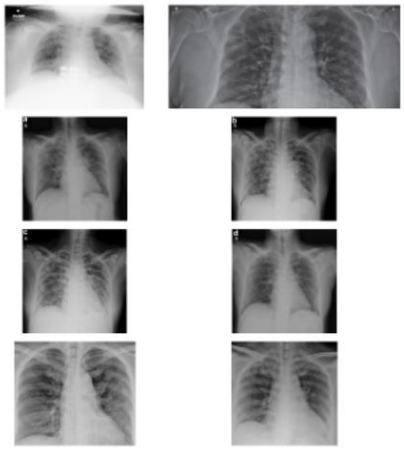
```
train_data = train_df[train_df['Dataset_type'] == 'TRAIN']
test_data = train_df[train_df['Dataset_type'] == 'TEST']
assert train_data.shape[0] + test_data.shape[0] == train_df.shape[0]
print(f"Shape of train data : {train_data.shape}")
print(f"Shape of test data : {test_data.shape}")
test_data.sample(10)
```

```
Shape of train data : (5286, 6)
Shape of test data : (624, 6)
```

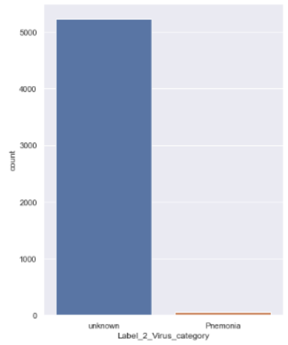
		X_ray_image_name	Label	Dataset_type	Label_2_Virus_category	Label_1_Virus_category
5717	5740	person96_bacteria_466.jpeg	Pnemonia	TEST	unknown	bacteria
5355	5378	IM-0050-0001.jpeg	Normal	TEST	unknown	unknown
5531	5554	person161_bacteria_759.jpeg	Pnemonia	TEST	unknown	bacteria
5647	5670	person122_bacteria_582.jpeg	Pnemonia	TEST	unknown	bacteria
5499	5522	NORMAL2-IM-0362-0001.jpeg	Normal	TEST	unknown	unknown
5501	5524	NORMAL2-IM-0360-0001.jpeg	Normal	TEST	unknown	unknown
5373	5396	IM-0025-0001.jpeg	Normal	TEST	unknown	unknown
5380	5403	NORMAL2-IM-0331-0001.jpeg	Normal	TEST	unknown	unknown
5428	5451	NORMAL2-IM-0267-0001.jpeg	Normal	TEST	unknown	unknown
5599	5622	person138_bacteria_658.jpeg	Pnemonia	TEST	unknown	bacteria

```
test_img_dir = './Chest-XRay-Dataset/Chest-XRay-Dataset/test/'
train_img_dir = './Chest-XRay-Dataset/Chest-XRay-Dataset/train/'

sample_train_images = list(os.walk(train_img_dir))[0][2][:8]
sample_train_images = list(map(lambda x: os.path.join(train_img_dir, x), sample_train_images))
sample_test_images = list(os.walk(test_img_dir))[0][2][:8]
sample_test_images = list(map(lambda x: os.path.join(test_img_dir, x), sample_test_images))
```

```
plt.figure(figsize = (10,10))
for iterator, filename in enumerate(sample_train_images):
    image = PIL.Image.open(filename)
    plt.subplot(4,2,iterator+1)
    plt.axis("off")
    plt.imshow(image, cmap=plt.cm.bone)

plt.tight_layout()
```



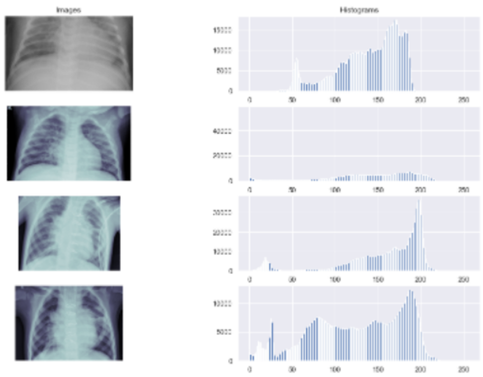
```
plt.figure(figsize=(6,8))
sns.countplot(train_data['Label_2_Virus_category']);
```




```
fig, ax = plt.subplots(4, 2, figsize=(15, 10))


pnemonia_path = train_data[train_data['Label_2_Virus_category']=='Pnemonia']['X_ray_image_name'].values

sample_pnemonia_path = pnemonia_path[6:10]
sample_pnemonia_path = list(map(lambda x: os.path.join(train_img_dir, x), sample_pnemonia_path))

for row, file in enumerate(sample_pnemonia_path):
    image = plt.imread(file)
    ax[row, 0].imshow(image, cmap=plt.cm.bone)
    ax[row, 1].hist(image.ravel(), 256, [0,256])
    ax[row, 0].axis('off')
    if row == 0:
        ax[row, 0].set_title('Images')
        ax[row, 1].set_title('Histograms')
fig.suptitle('Label_2_Virus_category = Pnemonia', size=16)
plt.show()
```

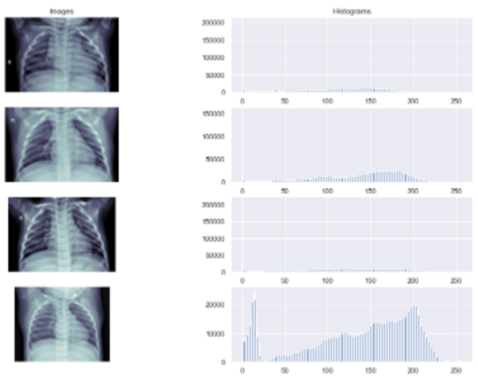
Label_2_Virus_category = Pnemonia

Label_2_Virus_category = Pnemonia




```
fig, ax = plt.subplots(4, 2, figsize=(15, 10))


normal_path = train_data[train_data['Label']=='Normal']['X_ray_image_name'].values

sample_normal_path = normal_path[:4]
sample_normal_path = list(map(lambda x: os.path.join(train_img_dir, x), sample_normal_path))

for row, file in enumerate(sample_normal_path):
    image = plt.imread(file)
    ax[row, 0].imshow(image, cmap=plt.cm.bone)
    ax[row, 1].hist(image.ravel(), 256, [0,256])
    ax[row, 0].axis('off')
    if row == 0:
        ax[row, 0].set_title('Images')
        ax[row, 1].set_title('Histograms')
fig.suptitle('Label = NORMAL', size=16)
plt.show()
```

Label = NORMAL




```
final_train_data = train_data[(train_data['Label'] == 'Normal') |
                              ((train_data['Label'] == 'Pnemonia') &
                               (train_data['Label_2_Virus_category'] == 'Pnemonia'))]
```

```
final_train_data['class'] = final_train_data.Label.apply(lambda x: 'negative' if x=='Normal' else 'positive')
test_data['class'] = test_data.Label.apply(lambda x: 'negative' if x=='Normal' else 'positive')

final_train_data['target'] = final_train_data.Label.apply(lambda x: 0 if x=='Normal' else 1)
test_data['target'] = test_data.Label.apply(lambda x: 0 if x=='Normal' else 1)
```

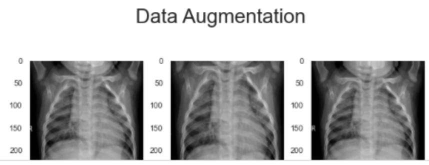
```
final_train_data = final_train_data[['X_ray_image_name', 'class', 'target', 'Label_2_Virus_category']]
final_test_data = test_data[['X_ray_image_name', 'class', 'target']]
```

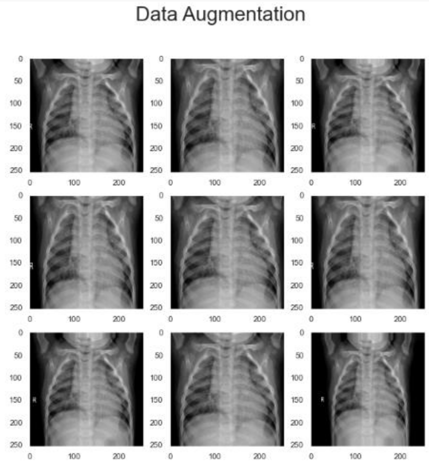
```
test_data['Label'].value_counts()
```

```
Pnemonia    390
Normal      234
Name: Label, dtype: int64
```

```
datagen =  ImageDataGenerator(
  shear_range=0.2,
  zoom_range=0.2,
)

def read_img(filename, size, path):
    #img = image.load_img(os.path.join(path, filename), target_size=size
    img = tf.keras.preprocessing.image.load_img(os.path.join(path, filename), target_size=size)
    #convert image to array
    input_arr = img_to_array(tf.keras.preprocessing.image.load_img(os.path.join(path, filename), target_size=size)) / 255
    #input_arr = tf.keras.preprocessing.image.img_to_array(image)
    input_arr = np.array(input_arr)
    return input_arr
```

```
samp_img = read_img(final_train_data['X_ray_image_name'][0],
                                 (255,255),
                                 train_img_dir)

plt.figure(figsize=(10,10))
plt.suptitle('Data Augmentation', fontsize=28)

i = 0


for batch in datagen.flow(tf.expand_dims(samp_img,0), batch_size=16):
    plt.subplot(3, 3, i+1)
    plt.grid(False)
#     print(batch.shape)
#     print(batch.ndim)
    plt.imshow(batch.reshape(255, 255, 3));

    if i == 8:
        break
    i += 1

plt.show();
```






```
pnemonia_df = final_train_data[final_train_data['Label_2_Virus_category'] == 'Pnemonia']
with_pnemonia_augmented = []

def augment(name):
    img = read_img(name, (255,255), train_img_dir)
    i = 0
    for batch in tqdm(datagen.flow(tf.expand_dims(img, 0), batch_size=32)):
        with_pnemonia_augmented.append(tf.squeeze(batch).numpy())
        if i == 20:
            break
        i =i+1


pnemonia_df['X_ray_image_name'].apply(augment)
```

```
5288    None
5289    None
5290    None
        ...
5905    None
5906    None
5907    None
5908    None
5909    None
Name: X_ray_image_name, Length: 624, dtype: object
```

```
print(len(train_arrays))
print(len(test_arrays))
```

```
1400
624
```

```
y_train = np.concatenate((np.int64(final_train_data['target'].values), np.ones(len(with_pnemonia_augmented), dtype=np.int64)))
```

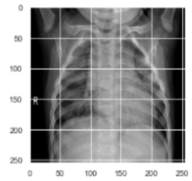
```
dummy = np.concatenate((np.array(train_arrays), np.array(with_pnemonia_augmented)))
```

```
train_tensors = tf.convert_to_tensor(dummy)
test_tensors  = tf.convert_to_tensor(np.array(test_arrays))
y_train_tensor = tf.convert_to_tensor(y_train)
y_test_tensor = tf.convert_to_tensor(final_test_data['target'].values)
```

```
train_dataset = tf.data.Dataset.from_tensor_slices((train_tensors, y_train_tensor))
test_dataset = tf.data.Dataset.from_tensor_slices((test_tensors, y_test_tensor))
```

```
for i,l in train_dataset.take(1):
    plt.imshow(i);
    print(l)
```

```
tf.Tensor(0, shape=(), dtype=int64)
```



```
BATCH_SIZE = 16
BUFFER = 1000
#BUFFER = int(1e4)
#BATCH_SIZE = 500
train_batches = train_dataset.shuffle(BUFFER).batch(BATCH_SIZE)
test_batches = test_dataset.batch(BATCH_SIZE)

for i,l in train_batches.take(1):
    print('Train Shape per Batch: ',i.shape);
for i,l in test_batches.take(1):
    print('Test Shape per Batch: ',i.shape);
```

```
Train Shape per Batch:  (16, 255, 255, 3)
Test Shape per Batch:  (16, 255, 255, 3)
```

```
INPUT_SHAPE = (255,255,3)

base_model = tf.keras.applications.ResNet50(input_shape= INPUT_SHAPE,
                                            include_top=False,
                                            weights='imagenet')

base_model.trainable = False
```

```
for i,l in train_batches.take(1):
    pass
base_model(i).shape
```

```
TensorShape([16, 8, 8, 2048])
```

```
model = Sequential()
model.add(base_model)
model.add(keras.layers.GlobalAveragePooling2D())
model.add(keras.layers.Dense(128))
#model.add(keras.layers.Dense(128,activation='elu'))
model.add(keras.layers.Dropout(0.2))
model.add(keras.layers.Dense(1, activation = 'sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 8, 8, 2048)        23587712

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 128)               262272

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 23,850,113
Trainable params: 262,401
Non-trainable params: 23,587,712
```

+ Code + Text

```
                loss = 'binary_crossentropy',
                metrics=['accuracy'])
```

```
history = model.fit(train_batches, epochs=10, validation_data=test_batches, callbacks=[callbacks])
```

```
Epoch 1/10
164/164 [==============================] - 487s 2s/step - loss: 0.6847 - accuracy: 0.7357 - val_loss: 0.5756 - val_accuracy: 0.6651
Epoch 2/10
164/164 [==============================] - 405s 2s/step - loss: 0.6445 - accuracy: 0.7678 - val_loss: 0.6574 - val_accuracy: 0.6410
Epoch 3/10
164/164 [==============================] - 337s 2s/step - loss: 0.5637 - accuracy: 0.7815 - val_loss: 0.5245 - val_accuracy: 0.7276
Epoch 4/10
164/164 [==============================] - 2695s 16s/step - loss: 0.4990 - accuracy: 0.7972 - val_loss: 0.6173 - val_accuracy: 0.6651
Epoch 5/10
124/164 [=====================>........] - ETA: 1:00 - loss: 0.5451 - accuracy: 0.7954
```

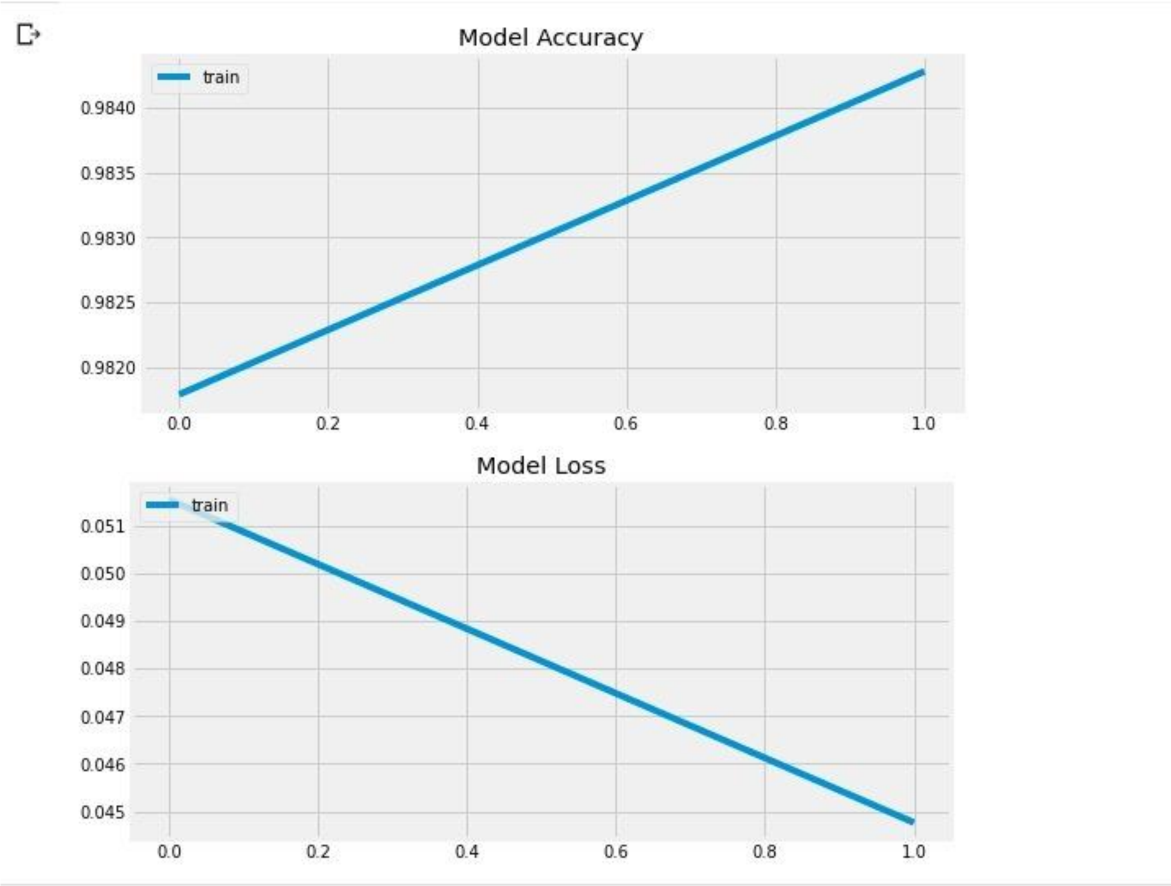
```
history = model.fit(train_batches, epochs=10, validation_data=test_batches, callbacks=[callbacks])
```

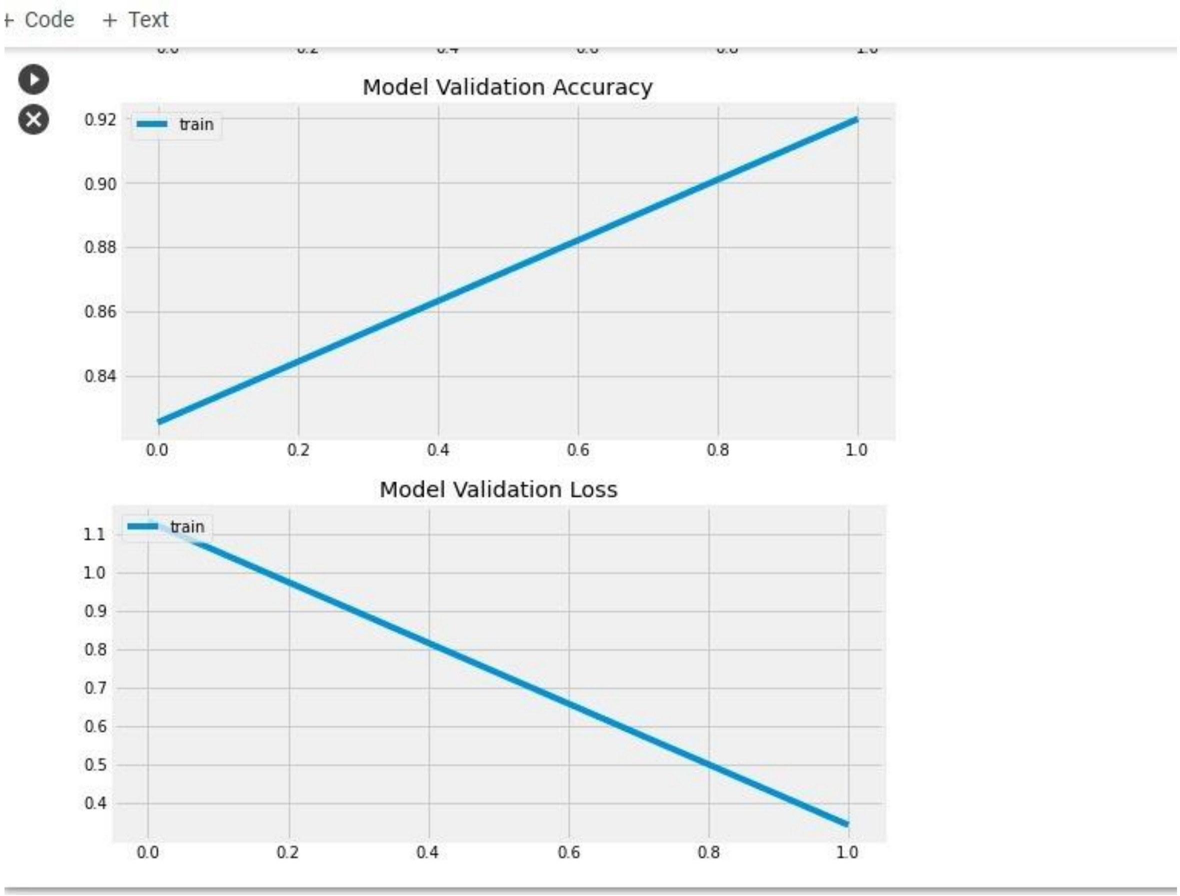
```
#predictions = model.predict(np.array(test_arrays))
pred =(model.predict(np.array(test_arrays)) > 0.5).astype("int32")
```

```
#predictions = model.predict(np.array(test_arrays))
preds =model.predict(np.array(test_arrays))
```

```
plt.figure(figsize=(8, 4))
plt.plot(history.epoch, history.history['accuracy'])
plt.title('Model Accuracy')
plt.legend(['train'], loc='upper left')
plt.show()

plt.figure(figsize=(8, 4))
plt.plot(history.epoch, history.history['loss'])
plt.title('Model Loss')
plt.legend(['train'], loc='upper left')
plt.show()

plt.figure(figsize=(8, 4))
plt.plot(history.epoch, history.history['val_accuracy'])
plt.title('Model Validation Accuracy')
plt.legend(['train'], loc='upper left')
plt.show()

plt.figure(figsize=(8,4))
plt.plot(history.epoch, history.history['val_loss'])
plt.title('Model Validation Loss')
plt.legend(['train'], loc='upper left')
plt.show()
```

5.3 RESULT

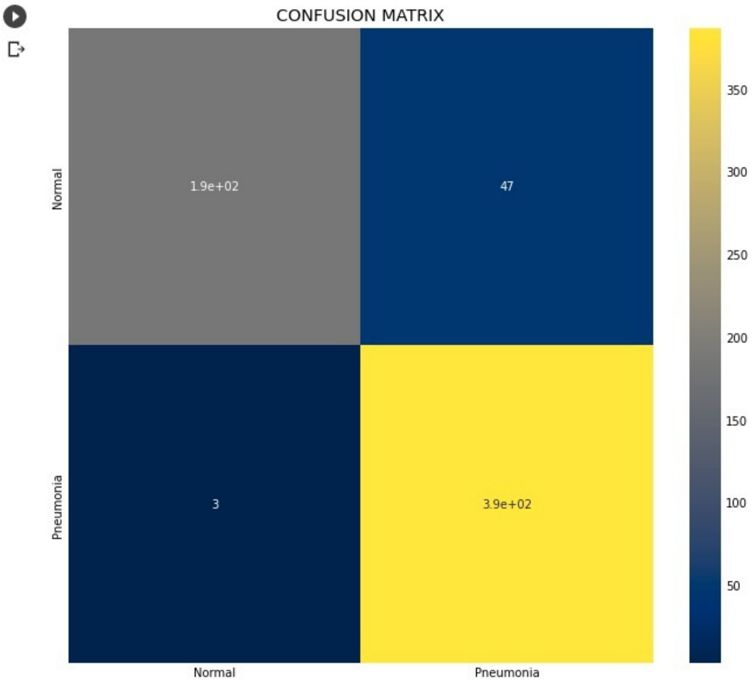

Model Accuracy
train
0.9840
0.9835
0.9830
0.9825
0.9820
0.0
0.2
0.4
0.6
0.8
1.0
Model Loss
train
0.051
0.050
0.049
0.048
0.047
0.046
0.045
0.0
0.2
0.4
0.6
0.8
1.0


+ Code
+ Text
Model Validation Accuracy
train
0.92
0.90
0.88
0.86
0.84
0.0
0.2
0.4
0.6
0.8
1.0
Model Validation Loss
train
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8
1.0

```
# classification report
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(test_data['target'], pred.flatten()))
```

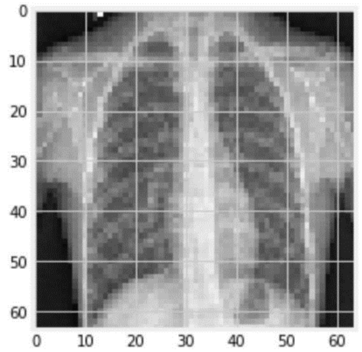
```
              precision    recall  f1-score   support

           0       0.98      0.80      0.88       234
           1       0.89      0.99      0.94       390

    accuracy                           0.92       624
   macro avg       0.94      0.90      0.91       624
weighted avg       0.93      0.92      0.92       624
```

+ Code + Text



Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving IM-0001-0001.jpeg to IM-0001-0001 (1).jpeg
Normal
```



CHAPTER 6

CONCLUSION AND FUTURE WORK

CONCLUSION AND FUTURE WORK

In the Phase one of major project, we went through the existing models of Pneumonia Detection using chest x-rays they are Pneumonia detection using CNN, Pneumonia Detection chest X-RAY Using Mask R-CNN and Pneumonia Detection Using Improved Faster R-CNN

These projects use to preprocess and labels the dataset as the first step that take more time label each image in dataset and the next step, they train the model with the labeled data to test the dataset and accuracy some of the models fails in detection of normal x-ray images and bacteria infected is Pneumonia.

In the future this work could be extended to detect and classify X-ray images consisting of lung cancer and pneumonia. Distinguishing X-ray images that contain lung cancer and pneumonia has been a big issue in recent times, and our next approach should be to tackle this problem.

REFERENCES

REFERENCES

1. https://www.analyticsvidhya.com/blog/2020/09/pneumonia-detection-using-cnn-with-implementation-in-python/
2. https://albertvillanova.github.io/blog/2020/11/17/pytorch-lightning-transfer-learning.html
3. https://www.hindawi.com/journals/cmmm/2021/8854892/
4. https://towardsdatascience.com/pneumonia-detection-using-deep-learning-5dc66468eb9
5. https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
6. https://www.mdpi.com/2079-9292/10/13/1512/pdf
7. https://iopscience.iop.org/article/10.1088/1757-899X/1022/1/012066/pdf
8. C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, Boston, America, 2015.View at: Google Scholar
9. T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, Hawaii, America, 2017.View at: Google Scholar
10. N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-NMS–improving object detection with one line of code," in *Proceedings of the IEEE international conference on computer vision*, pp. 5561–5569, Venice, Italy, 2017.View at: Google Scholar
11. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, Las Vegas, America, 2016.View at: Google Scholar
12. https://albertvillanova.github.io/blog/2020/11/17/pytorch-lightning-transfer-learning.html
13. https://www.hindawi.com/journals/cmmm/2021/8854892/
14. https://towardsdatascience.com/pneumonia-detection-using-deep-learning-5dc66468eb9
15. https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
16. https://www.mdpi.com/2079-9292/10/13/1512/pdf