PNEUMONIA DETECTION USING DEEP LEARNING

A PROJECT REPORT

Submitted By

MEGHA C

TVE20MCA2039

to

the APJ Abdul Kalam Technological University

in partial fulfilment of the requirements for the award of the degree

of

Master of Computer Applications

Department of Computer Applications

College of Engineering

Trivandrum-695016

MARCH 2022

Declaration

I undersigned hereby declare that the project report PNEUMONIA DETECTION USING DEEP LEARNING, submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University Kerala is a bonafide work done by me under supervision of Prof. Deepa S S. This submission represents my ideas in my words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity as directed in the ethics policy of the college and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title.

Place : Trivandrum **MEGHA C**

Date : 5/2/2022

DEPARTMENT OF COMPUTER APPLICATIONS

COLLEGE OF ENGINEERING
TRIVANDRUM

CERTIFICATE

This is to certify that the report entitled **PNEUMONIA DETECTION USING DEEP LEARNING** submitted by **MEGHA C** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications is a bonafide record of the project work carried out by him under my guidance and supervision. This report in any form has not been submitted to any University or Institute for any purpose.

Prof. Deepa S S
Project Guide

Prof. Deepa S S
Head of the Department

Acknowledgement

If words are considered as symbols of approval and tokens of acknowledgement, then let words play the heralding role of expressing our gratitude.

First of all we would like to thank **God** almighty for bestowing with wis- dom,courage and perseverance which had helped to complete this project Pneumonia Detection Using Deep Learning. This project has been a reality as a result of the help given by a large number of personalities.

I would like to thank **Dr Jiji C V**, Principal, College of Engineering Trivandrum, who helped me during the entire process of work.

I am extremely grateful to **Prof. Deepa S S**, HOD, Dept of Computer Applications, for providing me with best facilities and atmosphere for the creative work guidance and encouragement.And also for the valuable guidance, support and advices that aided in the successful completion of my project.

I profusely thank other teachers in the department and all other staffs of CET, for their guidance and inspirations throughout my course of study.

I owe my thanks to my friends and all others who have directly or indirectly helped me in the successful completion of this project. No words can express my humble gratitude to my beloved parents and relatives who have been guiding me in all walks of my journey.

MEGHA C

ABSTRACT

The coronavirus is a deadly virus that has claimed hundreds of thousands of lives in countries around the world. Older adults and people who have severe underlying medical conditions or prior cases of pneumonia seem to be at higher risk for developing more serious complications from the virus. With rising deaths and limited medical resources, doctors and medical professionals around the world are working around the clock to treat patients and prevent the spread of the virus. Severe forms of the virus can cause pneumonia leading to greater risk of death.

With the growing technological advancement, we have in this day and age, it is possible to use tools based on deep learning frameworks to detect pneumonia based on chest x-ray images. The challenge here would be to aid the diagnosis process which allows for expedited treatment and better clinical outcomes. It is crucial to have quick and accurate detection of pneumonia so patients can receive treatment in a timely manner especially in impoverished regions.

Contents

List of Figures

Chapter 1

Introduction

Artificial intelligence has found its use in various fields during the course of its development, especially in recent years with the enormous increase in available data. Its main task is to assist making better, faster and more reliable decisions. Artificial intelligence and machine learning are increasingly finding their application in medicine. This is especially true for medical fields that utilize various types of biomedical images and where diagnostic procedures rely on collecting and processing a large number of digital images. The application of machine learning in processing of medical images helps with consistency and boosts accuracy in reporting.

This project describes the use of deep learning algorithms to process chest X-ray images in order to support the decision-making process in determining the correct diagnosis. Specifically, the project is focused on the use of deep learning algorithm based on convolutional neural network in order to build a processing model. This model has the task to help with a classification problem that is detecting whether a chest X-ray shows changes consistent with pneumonia or not, and classifying the X-ray images in two groups depending on the detection results. Index Terms-artificial intelligence; convolutional neural network; deep learning; image processing, machine learning; pneumonia detection.

Chapter 2

Literature Review

Various references, articles, papers were checked on the convolutional neural network classification algorithm and their applications. They gave us ideas on which framework is better and why this algorithm will suit the problem for our project A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. Many journals regarding deep learning algorithms helped us understanding how we should figure out the huddles Infront of us. The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. Also,a CNN convolves learned features with input data, and uses 2D convolutional layers, making this architecture well suited to processing 2D data, such as images. Since CNNs eliminate the need for manual feature extraction, one doesn't need to select features required to classify the images. We are now aware of larger possibilities in the area of machine learning which we attained through referring various reading materials

Chapter 3

Requirement Analysis

3.1 Purpose

In order to achieve a working project in the python environment some major requirements were done. Software Development require some important steps to be taken. In developing our web application was also done with similar steps

3.2 Problem Definition

Our proposed system can predict whether the input chest x-ray scan provided by the user is pneumonic or not. We use CNN algorithm in order to predict the resulting outcome.

This project explains How a web application powered with deep learning algorithm can help public in early detection of pnuemonia.

3.3 Objective

The following are the main objectives of the System:

- The user can input their chest x-ray scan as image.
- The trained model in the web application will process the input.
- The web application then shows the resulting predicted value to the user.

3.4 Overall Description

3.4.1 Software requirement

- Python3
- Matplotlib
- Numpy
- Sklearn
- Tensorflow
- cv2
- Streamlit

3.4.2 Hardware requirement

- Windows or Linux system
- A computer with working peripherals

3.5 Technologies Used

3.5.1 Python

Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant white space. Its language constructs and object-oriented approach aims to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

3.5.2 Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

3.5.3 Numpy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

3.5.4 Scikit-learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

3.5.5 Tensorflow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

3.5.6 cv2

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java,

etc.cv2 is the module import name for opencv-python, "Unofficial pre-built CPU-only OpenCV packages for Python"

3.5.7 Streamlit

Streamlit is an open source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.

3.6 Functional requirements

Functional requirements represent the intended behaviour of the system. This behaviour may be expressed as services, tasks or functions that the specified system is required to perform. The following functional requirements have been identified for this project.

3.7 Non-Functional requirements

Non-Functional requirements define the general qualities of the software product. Non-functional requirement is in effect a constraint placed on the system or the development process. They are usually associated with the product descriptions such as maintainability, usability, portability, etc. It mainly limits the solutions for the problem. The solution should be good enough to meet the non- functional requirements.

Chapter 4

Methodology and System Design

4.1 System Design

Design of a system can be defined as the process of applying various techniques and principles for defining a device, a process or a system in sufficient detail to permit its physical realization. Thus, system design is a solution, a "how to" approach to the creation of a new system. The system is web based. The input is taken from the user through a web page and the input is passed to the model that is trained. The model perform tasks such as pre processing and feature extraction on the input data. The results of these processes are used to evaluate the input using the pre trained model.

4.2 Convolutional Neural Network

A Convolutional Neural Network (Con- vNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. Also,a CNN convolves learned features with input data, and uses 2D convolutional layers, making this

architecture well suited to processing 2D data, such as images. Since CNNs eliminate the need for manual feature ex- traction, one doesn't need to select features required to classify the images.

The CNN is a class of deep learning neural networks. CNNs represent a huge breakthrough in image recognition and classification, where they are most commonly used. They consist of few layers, such as: input, output and between them there are hidden layers. These hidden layers do the most work in terms of calculation. Convolutional layers can be found inside of the hidden layers . Hidden layers can also consist of pooling layers and fully connected layers. The most important building block of a CNN is,

convolutional Layer: each neuron in the convolutional layer is only connect to a small number of neurons (receptive field) in the next convolutional layer . Such a design allows the network structure to focus on a small low-level feature in the first hidden layer, and then aggregate them into higher-level features in the next hidden layer, and so on . This design is one the main reasons why CNNs are successfully used in image recognition.

Pooling Layer : The main use of pooling layers is to reduce the size of the input image without losing any important information. This is done to reduce the computational cost, and reduce memory usage. This process also reduces the number of parameters, which reduces the risk of overfitting.A max pooling layer, which is the most common type of pooling layer has been used in this project.

Fully Connected Layer : After several convolutional and max pooling layers, the final classification is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular (non-convolutional) artificial neural networks. Their activations can thus be computed as an affine transformation, with matrix multiplication followed by a bias offset (vector addition of a learned or fixed bias term).

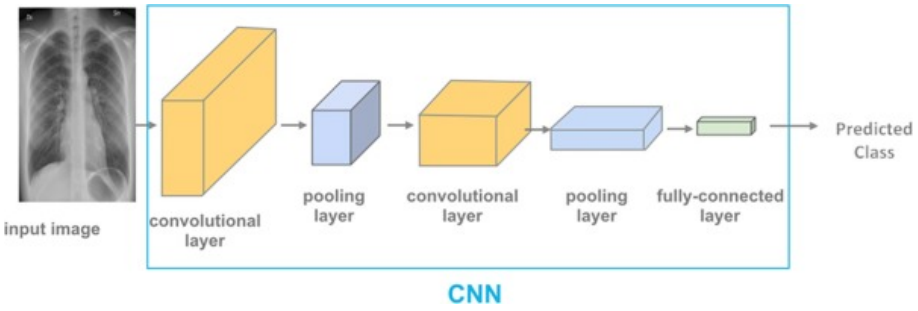

Figure 4.1: CNN in pnuemonia detection

4.3 Dataset

The dataset used for this project is the Chest X-Ray Images (Pneumonia) from Kaggle. The dataset consists of training data, validation data, and testing data. The training data consists of 5,216 chest x-ray images with 3,875 images shown to have pneumonia and 1,341 images shown to be normal. The validation data is relatively small with only 16 images with 8 cases of pneumonia and 8 normal cases. The testing data consists of 624 images split between 390 pneumonia cases and 234 normal cases.

In the training dataset there appears to be almost three times more pneumonia cases than normal cases which makes sense because this is medical data. Since the validation data is a small sample, it is combined with the training and testing data and shuffle the new dataset before applying the dataset split.

The new shuffled dataset is split into 60:15:25 ratio for training, validation and testing datasets.

4.3.1 Sample images in dataset

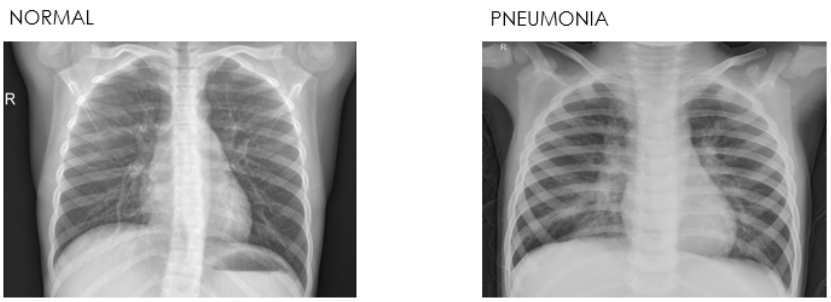

Figure 4.2: sample images

4.4 The Model

The model is a simple 3-layer convolutional neural network with max pooling and a ReLU activation function. The last 2 layers is a dense layer with the final dense layer consisting a sigmoid activation function.

4.4.1 ReLU Activation Function

The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.ReLU behaves very good in deep neural networks, mostly because it does not saturate for positive values and because it is fast to compute. There are some variations of the ReLU activation function such as leaky ReLU, parametric leaky ReLU and SELU.

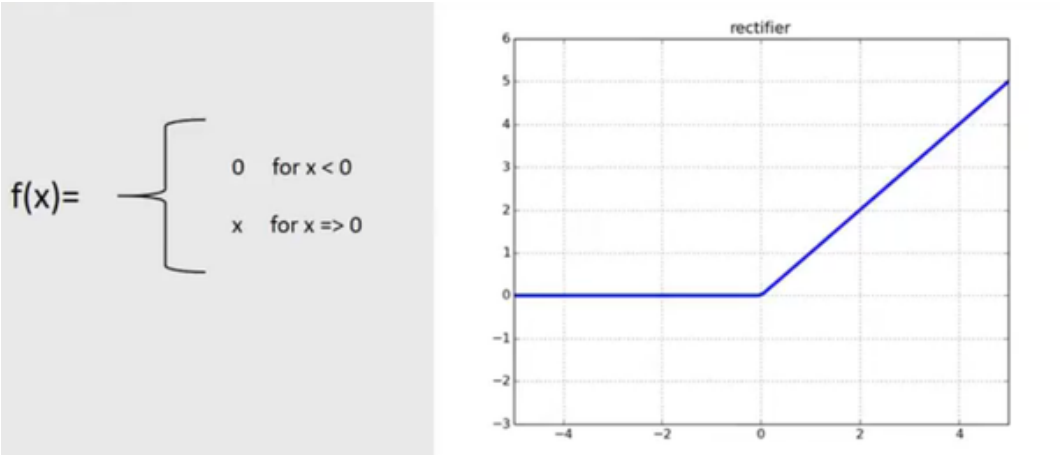

Figure 4.3: ReLU Activation Function

4.4.2 Sigmoid Activation Function

Sigmoid as an activation function in neural networks, A weighted sum of inputs is passed through an activation function and this output serves as an input to the next layer. When the activation function for a neuron is a sigmoid function it is a guarantee that the output of this unit will always be between 0 and 1.

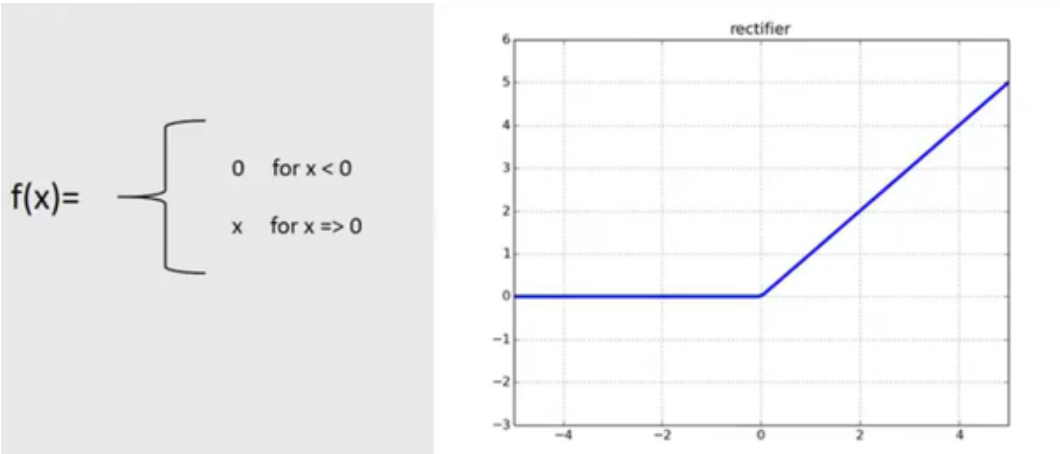

Figure 4.4: Sigmoid Activation Function

Chapter 5

Coding

5.1 Algorithm

- Step 1: Start
- Step 2: Collecting the dataset for the model. The overall result and quality of the project is heavily dependent on the quality of the dataset that is used to train the system
- Step 3: Pre processing the dataset
- Step 4: Training the model based on the dataset , The model is trained with training and validation dataset
- Step 6: Obtaining the model, The model after training and aquiring maximum accuracy
- Step 7: Implementing the model in the Web Application
- Step 8: Providing input data to the pre trained model to predict result
- Step 9: Stop

5.2 Source Code

Importing Libraries Unzipping dataset :

```
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
from zipfile import ZipFile
import pickle
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
```

Figure 5.1: Importing Libraries

```
def unzip():
  zip = ZipFile('/content/drive/MyDrive/Colab Notebooks/Pneumonia Detection/archive.zip', 'r')
  zip.extractall('Unzip_File')
  zip.close()

# unzip()
```

```
# # path name
train_dir = '/content/Unzip_File/chest_xray/train'
val_dir = '/content/Unzip_File/chest_xray/val'
test_dir = '/content/Unzip_File/chest_xray/test'

# categories
categories = ['NORMAL', 'PNEUMONIA']
```

Figure 5.2: Unzipping dataset

Creating Model :

```
image_width = 224
image_height = 224
image_size = (image_width,image_height)

def make_data():
  nor_data=[]
  pne_data=[]

  for dir in [train_dir, val_dir, test_dir]:
    for category in categories:
      path = os.path.join(dir, category)
      label = categories.index(category)
```

Figure 5.3: loading data(1)

```
    for img in os.listdir(path):
      img_path = os.path.join(path,img)

      try:
        image = cv2.imread(img_path)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image = cv2.resize(image, image_size)

        image = np.array(image, dtype=np.float64)
        if label==0:
          nor_data.append([image,label])
        else:
          pne_data.append([image,label])

      except:
        pass
pik = open('/content/drive/MyDrive/Colab Notebooks/Chest XRay/normal.pickle', 'wb')
pickle.dump(nor_data, pik)
pik.close()
pik = open('/content/drive/MyDrive/Colab Notebooks/Chest XRay/pneumonia.pickle', 'wb')
pickle.dump(pne_data, pik)
pik.close()
```

Figure 5.4: loading data(2)

```python
def load_data(dir):
  nor_count=0
  pne_count = 0
  pick = open(dir, 'rb')
  data = pickle.load(pick)
  pick.close()

  np.random.shuffle(data)

  feature = []
  labels = []

  for img,label in data:
    feature.append(img)
    labels.append(label)
    if(label==0):
      nor_count+=1
    else:
      pne_count+=1

  feature = np.array(feature, dtype=np.float64)
  labels = np.array(labels)

  feature /= 255.0

  return [feature, labels]
```

Figure 5.5: loading data(3)

```python
def split_data():
  train_nor = normal[0][:int(len(normal[0])*0.6)]                                    # 60% normal data
  train_nor_label = normal[1][:int(len(normal[1])*0.6)]
  val_nor = normal[0][int(len(normal[0])*0.6):int(len(normal[0])*0.75)]               # 15% normal data
  val_nor_label = normal[1][int(len(normal[1])*0.6):int(len(normal[1])*0.75)]
  test_nor = normal[0][int(len(normal[0])*0.75):]                                    # 25% normal data
  test_nor_label = normal[1][int(len(normal[1])*0.75):]

  train_pneu = pneumonia[0][:int(len(pneumonia[0])*0.6)]                             # 60% pneumonia data
  train_pneu_label = pneumonia[1][:int(len(pneumonia[1])*0.6)]
  val_pneu = pneumonia[0][int(len(pneumonia[0])*0.6):int(len(pneumonia[0])*0.75)]    # 15% pneumonia data
  val_pneu_label = pneumonia[1][int(len(pneumonia[1])*0.6):int(len(pneumonia[1])*0.75)]
  test_pneu = pneumonia[0][int(len(pneumonia[0])*0.75):]                             # 25% pneumonia data
  test_pneu_label = pneumonia[1][int(len(pneumonia[1])*0.75):]

  train_feature = np.concatenate((train_nor, train_pneu))
  train_label = np.concatenate((train_nor_label, train_pneu_label))

  val_feature = np.concatenate((val_nor, val_pneu))
  val_label = np.concatenate((val_nor_label, val_pneu_label))

  test_feature = np.concatenate((test_nor, test_pneu))
  test_label = np.concatenate((test_nor_label, test_pneu_label))

  return train_feature, train_label, val_feature, val_label, test_feature, test_label
```

Figure 5.6: splitting data

```
[ ]  datagen = ImageDataGenerator(
            # rotation_range = 45,
            # shear_range = 0.2,
            rescale=1/255.0,
            zoom_range = 0.2,
            height_shift_range = 0.2,
            width_shift_range = 0.2,
            fill_mode = "constant")
            # horizontal_flip = True,
            # vertical_flip = True)
            # brightness_range = (0.5, 1.5))

[ ]  it = datagen.flow(x_train.reshape(x_train.shape[0], image_height, image_width, 1), y_train, batch_size=64)
```

Figure 5.7: augmenting data

```
[ ]  input_layer = tf.keras.Input([image_width, image_height,1])

     conv1 = tf.keras.layers.Conv2D(filters=32, kernel_size=(5,5),padding='Same',
                                    activation='relu')(input_layer)

     pool1 = tf.keras.layers.MaxPooling2D(pool_size=(2,2))(conv1)

     # drop1 = tf.keras.layers.Dropout(0.25)(conv1)

     conv2 = tf.keras.layers.Conv2D(filters=64, kernel_size=(3,3), padding='Same',
                                    activation='relu')(pool1)

     # drop2 = tf.keras.layers.Dropout(0.25)(conv2)

     pool2 = tf.keras.layers.MaxPooling2D(pool_size=(2,2), strides=(2,2))(conv2)

     conv3 = tf.keras.layers.Conv2D(filters=96, kernel_size=(3,3), padding='Same',
                                    activation='relu')(pool2)

     pool3 = tf.keras.layers.MaxPooling2D(pool_size=(2,2), strides=(2,2))(conv3)

     flt1 =tf.keras.layers.Flatten()(pool3)

     dn1 = tf.keras.layers.Dense(128, activation='relu')(flt1)

     out = tf.keras.layers.Dense(1, activation='sigmoid')(dn1)
```

Figure 5.8: layers

```
[ ]  #@title
     model = tf.keras.Model(input_layer, out)
```

Figure 5.9: model

```
[ ]  opt = tf.keras.optimizers.Adam(learning_rate=0.001)

     model.compile(optimizer=opt, loss='binary_crossentropy',
                   metrics=['accuracy'])

     earlystopping = EarlyStopping(
         monitor = 'val_accuracy',
         patience = 5,
         restore_best_weights=False)

     # mcp_save = ModelCheckpoint(
     #     '/content/mdl_wts.hdf5',
     #     save_best_only=True,
     #     monitor='loss',
     #     mode='min')

     rlrop = ReduceLROnPlateau(
         monitor='val_accuracy',
         factor=0.2,
         patience=2)
```

Figure 5.10: Optimizers and early stopping

```
[ ]  model.fit(#it,
             x_train,y_train,
             # steps_per_epoch=len(x_train) // 64,
             validation_data=(x_val,y_val),
             epochs=100,
             callbacks=[earlystopping, rlrop])

     Epoch 1/100
     110/110 [==============================] - 46s 407ms/step - loss: 0.5837 - accuracy: 0.7235 - val_loss: 0.5274 - val_accuracy: 0.7292 - lr: 0.0010
     Epoch 2/100
     110/110 [==============================] - 44s 403ms/step - loss: 0.3666 - accuracy: 0.8220 - val_loss: 0.2335 - val_accuracy: 0.9101 - lr: 0.0010
     Epoch 3/100
     110/110 [==============================] - 44s 403ms/step - loss: 0.2402 - accuracy: 0.9026 - val_loss: 0.2164 - val_accuracy: 0.9033 - lr: 0.0010
     Epoch 4/100
     110/110 [==============================] - 44s 403ms/step - loss: 0.2031 - accuracy: 0.9243 - val_loss: 0.3011 - val_accuracy: 0.8737 - lr: 0.0010
     Epoch 5/100
     110/110 [==============================] - 44s 403ms/step - loss: 0.1725 - accuracy: 0.9396 - val_loss: 0.1557 - val_accuracy: 0.9397 - lr: 2.0000e-04
     Epoch 6/100
     110/110 [==============================] - 44s 403ms/step - loss: 0.1635 - accuracy: 0.9411 - val_loss: 0.1596 - val_accuracy: 0.9317 - lr: 2.0000e-04
     Epoch 7/100
     110/110 [==============================] - 44s 403ms/step - loss: 0.1619 - accuracy: 0.9419 - val_loss: 0.1512 - val_accuracy: 0.9408 - lr: 2.0000e-04
     Epoch 8/100
     110/110 [==============================] - 44s 403ms/step - loss: 0.1651 - accuracy: 0.9391 - val_loss: 0.1483 - val_accuracy: 0.9420 - lr: 2.0000e-04
```

Figure 5.11: model fit

```
[ ]  model.evaluate(x_test, y_test,verbose=1)

     46/46 [==============================] - 18s 392ms/step - loss: 0.1460 - accuracy: 0.9413
     [0.14600317180156708, 0.9412969350814819]

[ ]  model.save('/content/drive/MyDrive/Colab Notebooks/Pneumonia Detection/model_alpha1.h5')
```

Figure 5.12: evaluate model

```
import streamlit as st
import tensorflow as tf
import numpy as np
from PIL import Image, ImageOps
st.write("""
          # Pneumonia Prediction
          """
          )
upload_file = st.sidebar.file_uploader("Upload X-ray Images")
Generate_pred=st.sidebar.button("Predict")
model=tf.keras.models.load_model('model_alpha1.h5')
def import_n_pred(image_data, model):
    size = (224,224)
    image = ImageOps.fit(image_data, size)
    image = ImageOps.grayscale(image)
    img = np.asarray(image)
    reshape=img[np.newaxis,...]
    pred = model.predict(reshape)
    return pred
if Generate_pred:
    image=Image.open(upload_file)
    with st.expander('X_ray Image', expanded = True):
        st.image(image, width=350)#, use_column_width=True
    pred=import_n_pred(image, model)
    labels = ['NORMAL','PNEUMONIA']
    st.title("Prediction:{0}".format(labels[int(round(pred[0][0]))]))
```

Figure 5.13: UI code

Chapter 6

Testing and Implementation

System testing is the process in which the system undergoes experimental testing so as to check that the system does not fail i.e., to check whether the required system is running according to specification and user expectation. System testing also tests to find discrepancies between the system and its original objective, current specification and systems documentation. Hence most useful and practical approach is with the understanding that testing is the process of executing a program with the explicit intention of finding errors that is making the program fail. Testing Is considered to be the least creative phase of the whole cycle of system design. In the real sense it is the phase, which helps to bring out the creativity of the other phases make it shine.

6.1 Unit testing

Unit testing focuses verification error on the smallest unit of software design the module. Using the procedural design description as a guide, important control paths are tested to uncover errors with the boundary of module. The relative complexity of test and uncovered errors is limited by the constrained scope established for unit testing. The unit test is normally white box oriented and the step can be conducted in parallel for multiple modules. The module interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing. All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once. And finally, all handling paths are tested.

SLNO	PROCEDURE	EXPECTED RESULT	ACTUAL RESULT	PASS OR FAIL
1	Feeding user data	Model should access the inputted values	Same as expected	pass
2	Predicting the destination	Model should predict the output	Same as expected	pass
3	Should keep accuracy	Model should be accurate	Same as expected	pass

6.2 Integration Testing

Also known as integration and testing, is a step in software testing in which individual modules are combined and tested as a single block.Integration testing is conducted to evaluate its possibilities with specified functional requirements.

SLNO	PROCEDURE	EXPECTED RESULT	ACTUAL RESULT	PASS OR FAIL
1	The training model integrated to project	Predictions should me made	Same as expected	pass

6.3 System testing

System testing is the process in which the system undergoes experimental testing so as to check that the system does not fail i.e., to check whether the required system is running according to specification and user expectation

SLNO	PROCEDURE	EXPECTED RESULT	ACTUAL RESULT	PASS OR FAIL
1	Correct output is given for input	Correct output is obtained	Same as expected	pass

Chapter 7

Results and Discussion

The system performs all the functions as intended. The system predicts the output result with respect to the input given by the user. The aim behind this pneumonia detection project is to provide the applications of deep learning to the public, where they can use the power of machine learning and deep learning.

7.1 Accuracy

Deep learning model accuracy is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data. In our project the accuracy score obtained is around 95 percent.

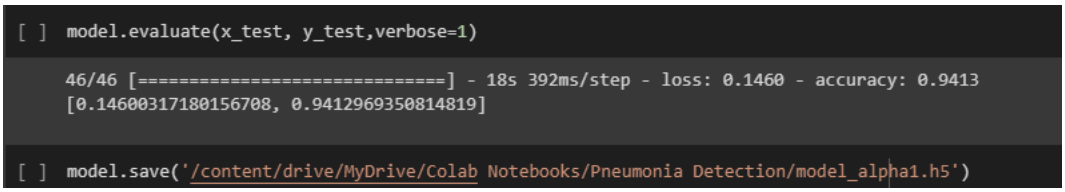

Figure 7.1: accuracy score

7.2 Advantages and Limitations

7.2.1 Advantages

- Timely detection of pneumonia

- Time saving
- Cost saving
- Can be used by medical practitioners to treat pneumonia
- Accurately detect pneumonic lungs from chest X-rays

7.2.2 Limitations

- Quality of the x-ray scan may effect the prediction by the model
- The system needs a x-ray scan to predict the output, which may be costly

Chapter 8

Conclusion

This paper presents a web application for public in order to identify a pneumonic lungs. Users can access the web app and make predictions. This application is implemented using Python and Streamlit. The system has reached a steady state where all bugs have been eliminated. The system is operated at a high level of efficiency. Accuracy of the model is above 0.9 which is good for trained models.

Chapter 9

Future Enhancement

Since there will be many users on system heavy traffic need to be addressed, Mobile version of the application also can be the future of the applications. The model can be updated to include doctor references to the user whose input scan image was found to be pneumonic. Rapid changes to the trained models can be done faster.

Chapter 10

Appendix

10.1 Screenshot

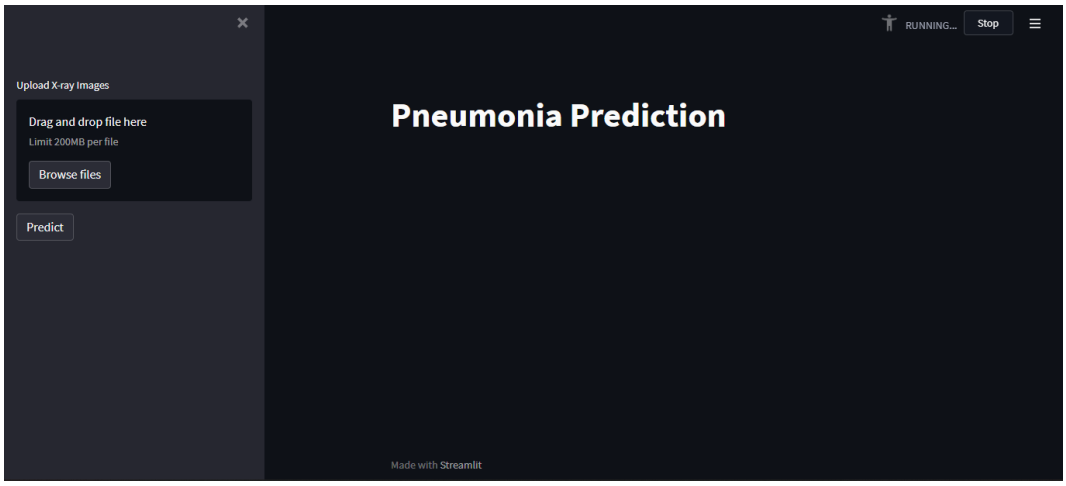

Figure 10.1: screen snap(1)

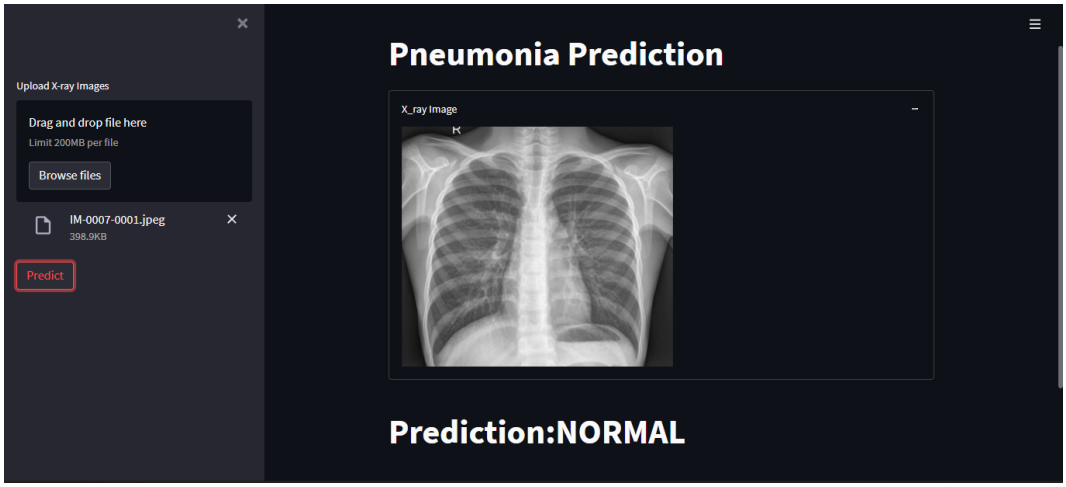

Figure 10.2: screen snap(2)

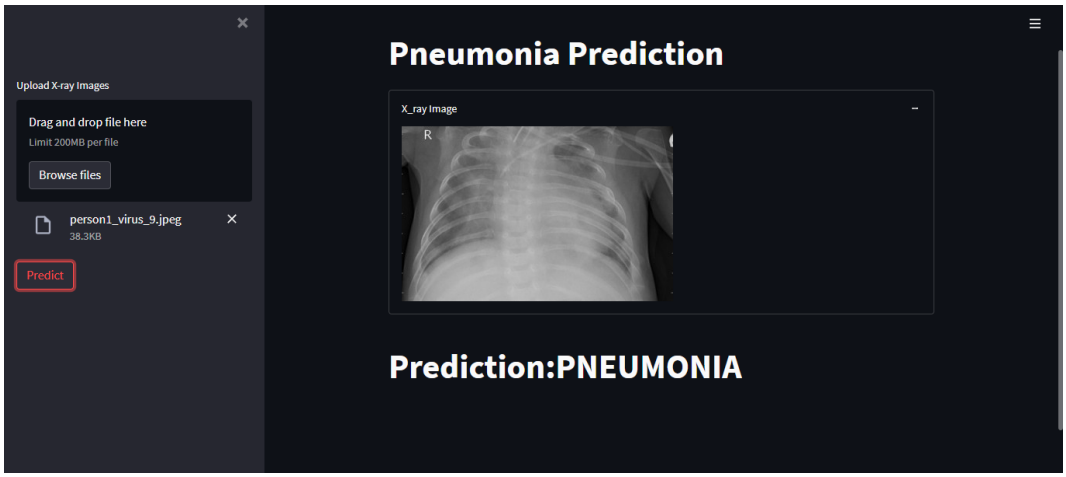

Figure 10.3: screen snap(3)

Bibliography

[1] Pneumonia Detection Using Convolutional Neural Networks (CNNs) by V. Sirish Kaushik, Anand Nayyar, Gaurav Kataria and Rachna Jain

[2] Pneumonia Detection Using Deep Learning Based on Convolutional Neural Network by Luka Račić, Tomo Popović, Senior Member, IEEE, Stevan Čakić, Stevan Šandi