1.Introduction:

1.1. Overview:

Using Deep-Learning algorithm whether a human has pneumonia or not can be determined from an X-Ray image of the human lungs. The algorithm works in such a way that when an image is uploaded it predicts whether or not the given image is an X-Ray of lungs having pneumonia or normal.This is an example of binary classification.

1.2. Purpose:

The purpose of this project is pnemonia prediction using x-ray images is to create a web application un which users can upload an image of their X-ray and find out whether they have pneumonia or normal.

2.Literature Survey:

2.1. Existing Problem:

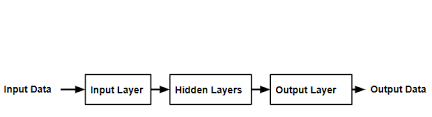
If a person thinks he is suffering from pneumonia then he should go to the hospital and there the doctors they will see the X-ray of the lungs to predict if there is pneumonia. However they cannot tell that the patient is suffering from pneumonia just by seeing the X-ray. Other tests must also be perform to determine with certainity that the patient has pneumonia.

2.2.Proposed Solution:

To simplify the process of detecting pneumonia in a patient an CNN model can be created that predicts whether or not a patient is having pneumonia by uploading the image of the patients X-Ray. It will accept the uploaded image, resize and tranform, convert it into an array and then predict whether it has normal or pneumonia.

3.Theoretical Analysis:

3.1. Block diagram:



3.2. Hardware/Software Designing:

For creating the CNN model we need keras 2.4.4 to import libraries like ImageDataGenerator for data preprocessing and sequential,dense,etc. for building the model. Numpy and tensorflow 1.14. For creating app.py python code to run the web application we require to import os, global graph, flask and WSGIServer.

4. Experimental Investigations:

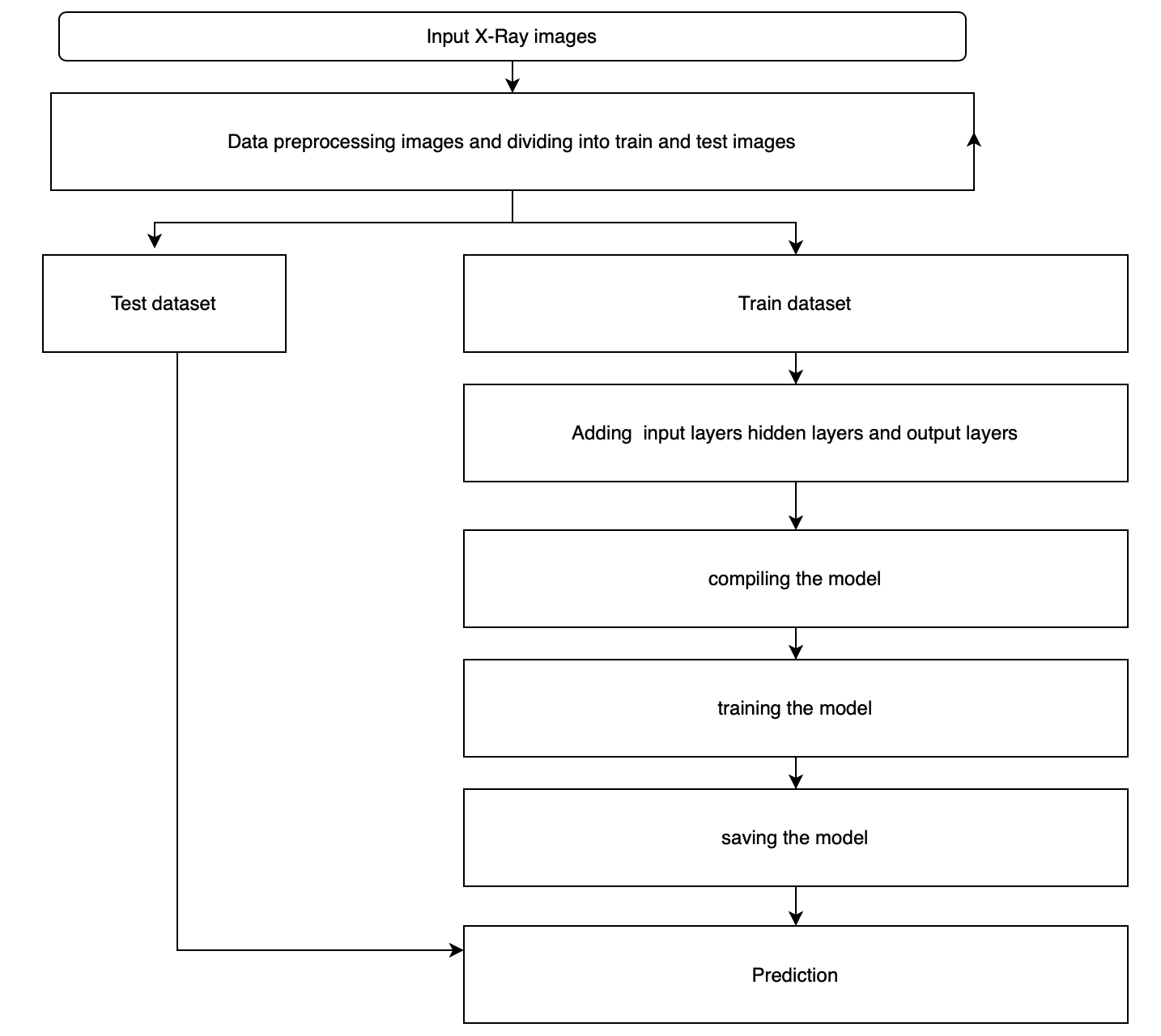
First data preprocessing rescaling and transformation was performed on the images of the given dataset. Then libraries are imported and model was initialised. After initialising the model first convolution layer is created. 3x3 feature detecter is used. Then max pooling layer is added. 2x2 matrix is used. After that flatten layer is added and last hidden and output layers. After adding all the layers of the CNN model it is compiled and trained with 10 epochs. After training model was saved.

For prediction saved model is loaded, transformations takes place and the given image is predicted whether normal or pneumonia. X-ray image given here was normal and it predicted accurately as normal.

After saving cnn.h5 and deep learning algorithm file html file and corresponding styles.css and file.js are created. Css is for the layout of the web application and .js is neccesary for previewing the image uploaded in the web application.

Last python code is created app.py. This is to run the model in the web application. After importing the libraries model is loaded. Html file is rendered and uploaded image is saved to uploads folder. The image is resized, converted into array form then interpreted by the machine when running the application. Finally prediction is displayed.

5. Flowchart:



6. Result:

When app.py is running in the terminal in the web application image can be uploaded and the model will predict whether it is pnemonia or normal. The X-Ray image will also be previewed.

7. Advantages and disadvantages:

The advantage of this cnn is it gives accurate prediction even when it is trained with less number of epochs. Generally accuarcy will be high only when more than 100 epochs are initialised.

Disadvantage is spatial invarience is not possible.

8. Applications:

Pneumonia prediction using x-ray images can be applied in medical field for identifying whether a human has pneumonia or normal.

9. Conclusion:

In this project deep learning cnn model and web application to predict pneumonia from an x-ray image has been created. An x-ray can be uploaded and the algorithm will detect pneumonia.

10. Future scope:

The web application can be used for identifying pneumonis in a patient. Currently it is a time consuming process to diagnos a patient with pneumonia but with this application it can identify in very less time and patients suffering from severe pneumonia can get treatment immediately.

11. Bibilography appendix:

Source code:

Deep learning algorithm:

**from** keras.preprocessing.image **import** ImageDataGenerator

train\_datagen=ImageDataGenerator(rescale=1/255,shear\_range=0.2,zoom\_range=0.2,horizontal\_flip=**True**)

test\_datagen=ImageDataGenerator(rescale=1/255)

x\_train=train\_datagen.flow\_from\_directory(r'/users/apurvaaddula/Desktop/chest\_xray/train',target\_size=(64,64),batch\_size=32,class\_mode='binary')

x\_test=test\_datagen.flow\_from\_directory(r"/users/apurvaaddula/Desktop/chest\_xray/test",target\_size=(64,64),batch\_size=32,class\_mode='binary')

print(x\_train.class\_indices)

**from** keras.models **import** Sequential

**from** keras.layers **import** Dense

**from** keras.layers **import** Conv2D

**from** keras.layers **import** MaxPooling2D

**from** keras.layers **import** Flatten

model=Sequential()

model.add(Conv2D(32,(3,3),input\_shape=(64,64,3),activation='relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Flatten())

model.add(Dense(output\_dim=128,init='random\_uniform',activation='relu'))

model.add(Dense(output\_dim=1,init='random\_uniform',activation='sigmoid'))

model.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy'])

**import** tensorflow **as** tf

tf.compat.v1.global\_variables

model.fit\_generator(x\_train,steps\_per\_epoch=163,epochs=10,validation\_data=x\_test,validation\_steps=20)

model.save('cnn.h5')

**from** keras.models **import** load\_model

**from** keras.preprocessing **import** image

**import** numpy **as** np

model=load\_model("cnn.h5")

img=image.load\_img(r"/users/apurvaaddula/Desktop/IM-0001-0001 copy.jpeg",target\_size=(64,64))

x=image.img\_to\_array(img)

x=np.expand\_dims(x,axis=0)

pred=model.predict\_classes(x)

app.py:

**from** \_\_future\_\_ **import** division, print\_function

**import** sys

**import** os

**import** glob

**import** numpy **as** np

**from** keras.preprocessing **import** image

**from** keras.applications.imagenet\_utils **import** preprocess\_input, decode\_predictions

**from** keras.models **import** load\_model

**from** keras **import** backend

**from** tensorflow.keras **import** backend

**import** tensorflow **as** tf

**global** graph

graph=tf.compat.v1.get\_default\_graph()

**from** skimage.transform **import** resize

**from** flask **import** Flask, redirect, url\_for, request, render\_template

**from** werkzeug.utils **import** secure\_filename

**from** gevent.pywsgi **import** WSGIServer

app=Flask(\_\_name\_\_)

model=load\_model('cnn.h5')

@app.route('/')

**def** home():

**return** render\_template('index.html')

@app.route('/predict',methods=['GET','POST'])

**def** upload():

**if** request.method=='POST':

f=request.files['image']

basepath=os.path.dirname(\_\_file\_\_)

file\_path=os.path.join(basepath,'uploads',secure\_filename(f.filename))

f.save(file\_path)

img=image.load\_img(file\_path, target\_size=(64,64))

x=image.img\_to\_array(img)

x=np.expand\_dims(x,axis=0)

**with** graph.as\_default():

preds=model.predict\_classes(x)

index=['Normal','Pneumonia']

text='prediction:'+str(index[preds[0][0]])

**return** text

**if** \_\_name\_\_=='\_\_main\_\_':

app.run(debug=**True**,threaded=**False**)