PNEUMONIA DETECTION USING CONVOLUTIONAL NEURAL NETWORK ALGORITHM

A Mini Project work submitted in partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

by

B.SUMITH BABU	**17H61A04D1**
G.ARJUN KUMAR	**17H61A04D7**
J.SUJATHA	**18H65A0426**

Under the guidance of

B.SANTHOSH KUMAR

Assistant Professor

Department of ECE

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

AN0URAG GROUP OF INSTITUTIONS

AUTONOMOUS

SCHOOL OF ENGINEERING

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

Venkatapur(V), Ghatkesar (M), Medchal Dist, Telangana-500088.

2017 - 2021

ANURAG GROUP OF INSTITUTIONS

AUTONOMOUS

SCHOOL OF ENGINEERING

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

Venkatapur(V), Ghatkesar (M), Medchal Dist, Telangana-500088.

2017 - 2021

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that this project report entitled ***PNEUMONIA DETECTION USING CONVOLUTIONAL NEURAL NETWORK*** being submitted by

B.SUMITH BABU	**17H61A04D1**
G.ARJUN KUMAR	**17H61A04D7**
J.SUJATHA	**18H65A0426**

In partial fulfillment for the award of the Degree of Bachelor of Technology in Electronics & Communication Engineering to the Jawaharlal Nehru Technological University, Hyderabad is a record of bonafide work carried out under my guidance and supervision. The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

B.SANTHOSH KUMAR
Assistant Professor
Department of ECE

DR. S.SATHEES KUMARAN
Head of the Department
Department of ECE

External Examiner

ACKNOWLEDGEMENT

This project is an acknowledgement to the inspiration, drive and technical assistance contributed by many individuals. This project would have never seen light of this day without the help and guidance we have received. We would like to express our gratitude to all the people behind the screen who helped us to transform an idea into a real application.

It's our privilege and pleasure to express our profound sense of gratitude to **B.SANTOSH KUMAR, Assistant Professor**, Department of ECE for his guidance throughout this dissertation work.

We express my sincere gratitude to **Dr. S.SATHEESKUMARAN, Head of Department**, Electronics and Communication Engineering for his precious suggestions for the successful completion of this project. He is also a great source of inspiration to our work.

We would like to express my deep sense of gratitude to **Dr. K.S.RAO, Director**, Anurag Group Of Institutions for his tremendous support, encouragement and inspiration. Lastly, we thank almighty, our parents, friends for their constant encouragement without which this assignment would not be possible. We would like to thank all the other staff members, teaching and non- teaching, which have extended their timely help and eased my work.

BY

B. SUMITH BABU 17H61A04D1
G. ARJUN KUMAR 17H61A04D7
J.SUJATHA 18H65A0426

DECLARATION

We hereby declare that the result embodied in this project report entitled **"PNEUMONIA DETECTION USING CONVOLUTONAL NEURAL NETWORK"** is carried out by us during the year 2020-2021 for the partial fulfillment of the award of **Bachelor of Technology in Electronics and Communication Engineering**, from ANURAG GROUP OF INSTITUTIONS. We have not submitted this project report to any other Universities / Institute for the award of any degree.

BY

BARE SUMITH BABU	**17H61A04D1**
GANDLA ARJUN KUMAR	**17H61A04D7**
JARPULAVATH SUJATHA	**18H65A0426**

ABSTRACT

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. Lungs image pre-processing and compilation of dataset for deep learning - The project is about diagnosing pneumonia from X Ray images of lungs of a person using self laid convolutional neural network. The images were of size greater than 1000 pixels per dimension and the total dataset was tagged large and had a space of 800 MB.

TABLE OF CONTENTS

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

One of the major factors associated with pneumonia in children is indoor air pollution. Apart from this, under-nutrition, lack of safe water, sanitation and basic health facilities are also major factors. Pneumonia is an interstitial lung disease caused by bacteria, fungi or viruses. It accounted for approximately 16% of the 5.6 million under-five deaths, killing around 880,000 children in 2016. Affected victims were mostly less than two years old. Timely detection of pneumonia can help to prevent the deaths of children. This paper presents convolutional neural network models to accurately detect pneumonic lungs from chest X-rays, which can be utilized in the real world by medical practitioners to treat pneumonia. These models have been trained to classify chest X-ray images into normal and pneumonia in a few seconds, hence serving the purpose of early detection of pneumonia. Although transfer learning models based on convolutional neural networks like AlexNet, ResNet50, InceptionV3, VGG16 and VGG19 are some of the most successful ImageNet dataset models with pre-trained weights, they were not trained on this dataset as the size of dataset taken for our research is not as extensive compared to ones which generally employ transfer learning . A classification models were built using CNN(Convolutional neural network) to detect pneumonia from chest X-ray images to help control this deadly infection in children and other age groups. Accuracy of the model is directly correlated with the size of the dataset, that is, the use of large datasets helps improve the accuracy of the model, but there is no direct correlation between the number of convolutional layers and the accuracy of the model.

To obtain the best results, a certain number of combinations of convolution layers, dense layers, dropouts and learning rates have to be trained by evaluating the models after each execution. Initially, simple models with one convolution layer were trained on the dataset, and thereafter, the complexities were increased to get the model that not only achieved desired accuracies but also outperformed other models in terms of recall and F1 scores. The objective of the project is to develop CNN models from scratch which can classify and thus detect pneumonic patients from their chest Xrays with high validation accuracy, recall and F1 scores. Recall is often favored in medical imaging cases over other performance evaluating parameters, as it gives a measure of false negatives in the results. The number of false negatives in the result is very crucial

in determining the real-world performance of models. If a model achieves high accuracy but low recall values, it is termed as underperforming, inefficacious and even unsafe as higher false-negative values imply higher number of instances where the model is predicting a patient as normal, but in reality, the person is diseased. Hence, it would risk the patient's life. To prevent this, the focus would be only models with great recall values, decent accuracies and F1 scores.

CHAPTER 2

LITERATURE REVIEW

Several new frameworks and designs using various learning models have been developed along with infinite datasets have helped counts to beat restorative work power in different remedial imaging assignments, for instance, skin threatening development portrayal , channel unmistakable confirmation , arrhythmia disclosure , and macular diabetic retinopathy distinguishing proof . Robotized examine using chest radiographs have gotten creating interests. These counts are dynamically being used for driving lung handle recognizable proof and pneumonic tuberculosis course of action .e execution of a couple convolutional models on different varieties from the standard relying upon the uninhibitedly available Open I dataset found that a comparative significant convolutional mastermind configuration doesn't perform well over all peculiarities , outfit models basically improved gathering precision when differentiated and individual model, therefore, other types of limited learning method will help to improve the accuracy when stood out from rule based procedures. Truthful dependence occurring in names has been gathered to reach at dynamically definite conjectures; thusly beating various strategies on given 13 pictures looked over 14 distinct classes. Estimations for separating and foreseeing names radiating from different radiology pictures similarly as reports have been inspected, yet the image names were generally obliged to illness names, thusly missing sensible information. Area of diseases from X-shaft pictures was reviewed in , portrayals on picture sees from chest X-bar were finished in, and body parts division from chest X-bar pictures and figured tomography operations were implemented in. Then again, taking in picture features from substance and making picture portrayals relative with what a human would depict are yet to be mishandled. In PC vision, profound learning has just indicated its capacity for picture arrangement with superhuman precision. What's more, the therapeutic picture handling field is strikingly investigating profound learning. Be that as it may, one significant issue in the medicinal space is the accessibility of huge datasets with solid ground-truth explanation. In this manner, move learning draws near, as proposed by Bar et al.6, were frequently considered to defeat such issues. In various types of learning techniques, CNN proved to be one of the best algorithms for image classification, analysis, segmentation and other tasks. One of the latest supercomputer discovered is NVidia DGX2 which has enhanced the

performance of several CNN classification methods. But there is still problem when the CNN architectures are consuming large resources for computation purpose and also overhead. Various types of hardware accelerators and their architectures are discussed in for reducing the power consumption and large overheads. An example of such accelerator is FGPA discussed in for minimizing power consumption.

Further, using the modern techniques, some hardware related models were introduced. The modern research also relies on the various optimization techniques. Genetic Algorithms are one of the most advance optimization techniques for the researchers having large hyper parameters discussed. Recently, a study says that Google introduced GPipe which is a machine learning library for training the data in a parallel way. ID-CNNs are one of the recent advance techniques which is able to perform better feature extraction in an efficient way but it is mostly suitable for the sequential data. Recently, many data scientists have proved that using CNNs in Deep Learning will improve the performance of the algorithms and theses scientists have used energy physics for the particle collision analysis in energy physics which has shown great results. Therefore, CNNs have proved very efficient in classification tasks used in Deep Learning.

CHAPTER 3

METHODOLOGY

3.1 PROPOSED METHOD:

The method proposed in the project is convolutional neural network using 4 hidden layers with ReLU (Rectified linear unit) activation function upto one fully connected layer then a dropout layer with key 0.5. The model used in project is compiled using adam optimizer learning rate reduced to 0.0001, binary cross entropy function and sigmoid activation function as shown in Fig 4.1

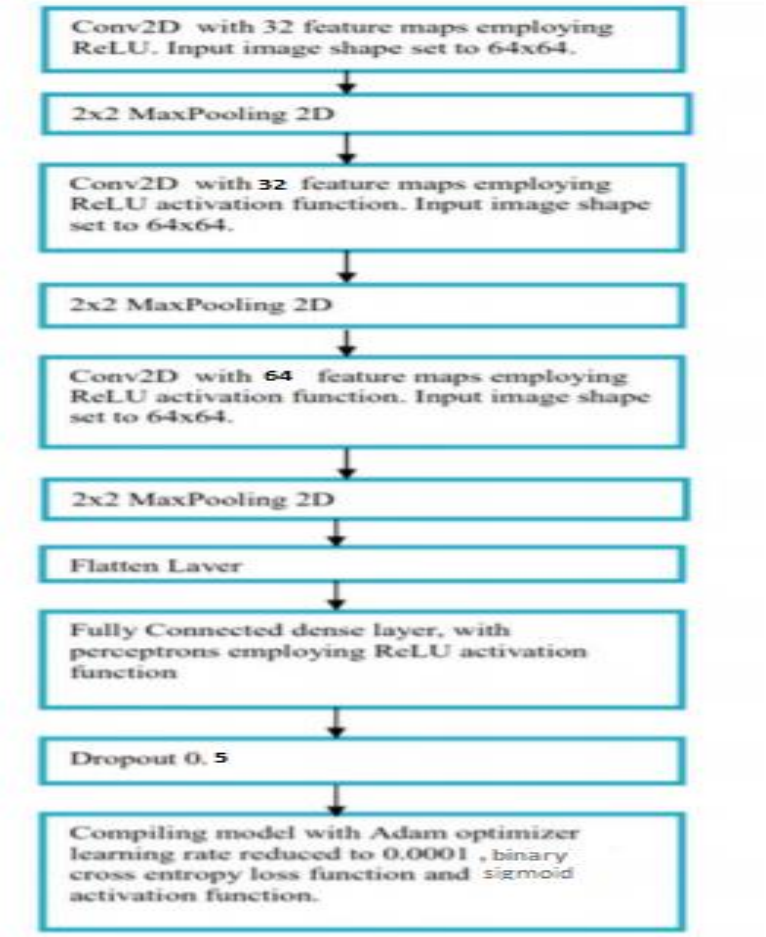

Fig 3.1 Flow chart of the model

3.2 CONVOLUTIONAL NEURAL NETWORK

CNN models have been created from scratch and trained on Chest X-Ray Images (Pneumonia) dataset on Kaggle. Keras neural network library with TensorFlow backend has been used to implement the models. Dataset consists of 2916 training images, 624 testing images and 16 validation images. Data augmentation has been applied to achieve better results from the dataset. The models have been trained on the training dataset, each with 4 of convolutional layers. Model was trained for 25 epochs, with training and testing batch sizes of 32 and 1, respectively. The following sub-headings further explain the above stages in depth.

3.2.1 Introduction to Convolutional Neural Network:

CNN models are feed-forward networks with convolutional layers, pooling layers, flattening layers and fully connected layers employing suitable activation functions

Convolutional layer:

It is the building block of the CNNs. Convolution operation is done in mathematics to merge two functions. In the CNN models, the input image is first converted into matrix form. Convolution filter is applied to the input matrix which slides over it, performing element-wise multiplication and storing the sum. This creates a feature map. 3 × 3 filter is generally employed to create 2D feature maps when images are black and white. Convolutions are performed in 3D when the input image is represented as a 3D matrix where the RGB color represents the third dimension. Several feature detectors are operated with the input matrix to generate a layer of feature maps which thus forms the convolutional layer.

Activation functions:

The models presented in this project use two different activation functions, namely ReLU activation function and sigmoid activation function. The ReLU activation function stands for rectified linear function . It is a nonlinear function that outputs zero when the input is negative and outputs one when the input is positive. The ReLU function is given by the following formula: This type of activation function is broadly used in CNNs as it deals with the problem of vanishing gradients and is useful for increasing the nonlinearity of layers. ReLU activation function has many variants such as Noisy ReLUs, Leaky ReLUs and Parametric ReLUs. Advantages of ReLU over other activation functions are computational simplicity and representational sparsity. Sigmoid activation function is used in model presented in this project.

This broadly used activation function is employed in the last dense layer of the model. This activation function normalizes inputs into a probability distribution. Binary cross-entropy cost function is mostly used with this type of activation function.

Pooling layer:

Convolutional layers are followed by pooling layers. The type of pooling layer used in all four models is max-pooling layers. The max-pooling layer having a dimension of 2 × 2 selects the maximum pixel intensity values from the window of the image currently covered by the kernel. Max-pooling is used to down sample images, hence reducing the dimensionality and complexity of the image. Two other types of pooling layers can also be used which are general pooling and overlapping pooling. The models presented in this paper use max-pooling technique as it helps recognize salient features in the image.

Flattening layer and fully connected layers:

After the input image passes through the convolutional layer and the pooling layer, it is fed into the flattening layer. This layer flattens out the input image into a column, further reducing its computational complexity. This is then fed into the fully connected layer/dense layer. The fully connected layer has multiple layers, and every node in the first layer is connected to every node in the second layer. Each layer in the fully connected layer extracts features, and on this basis, the network makes a prediction. This process is known as forward propagation. After forward propagation, a cost function is calculated. It is a measure of performance of a neural network model. The cost function used in all four models is categorical cross-entropy. After the cost function is calculated, back propagation takes place. This process is repeated until the network achieves optimum performance. Adam optimization algorithm has been used in all four models.

Reducing Overfitting:

The first model exhibits substantial overfitting; hence, dropout technique was employed in the later models. Dropout technique helps to reduce overfitting and tackles the problem of vanishing gradients. Dropout technique encourages each neuron to form its own individual representation of the input data. This technique on a random basis cuts connections between neurons in successive layers during the training process. Learning rate of models was also modified, to reduce overfitting. Data augmentation technique can also be employed to reduce overfitting.

3.2.2 Algorithm of CNN classifiers:

The algorithms used in the convolutional neural network classifiers have been explained in Fig.4.1 shows the flowchart of the overall schema of project. The number of epochs for all the classifier models presented in this paper was fixed at 25 after training and testing several CNN models over the course of project. Classifier models trained for more number of epochs have showed overfitting. Several optimizer functions were also

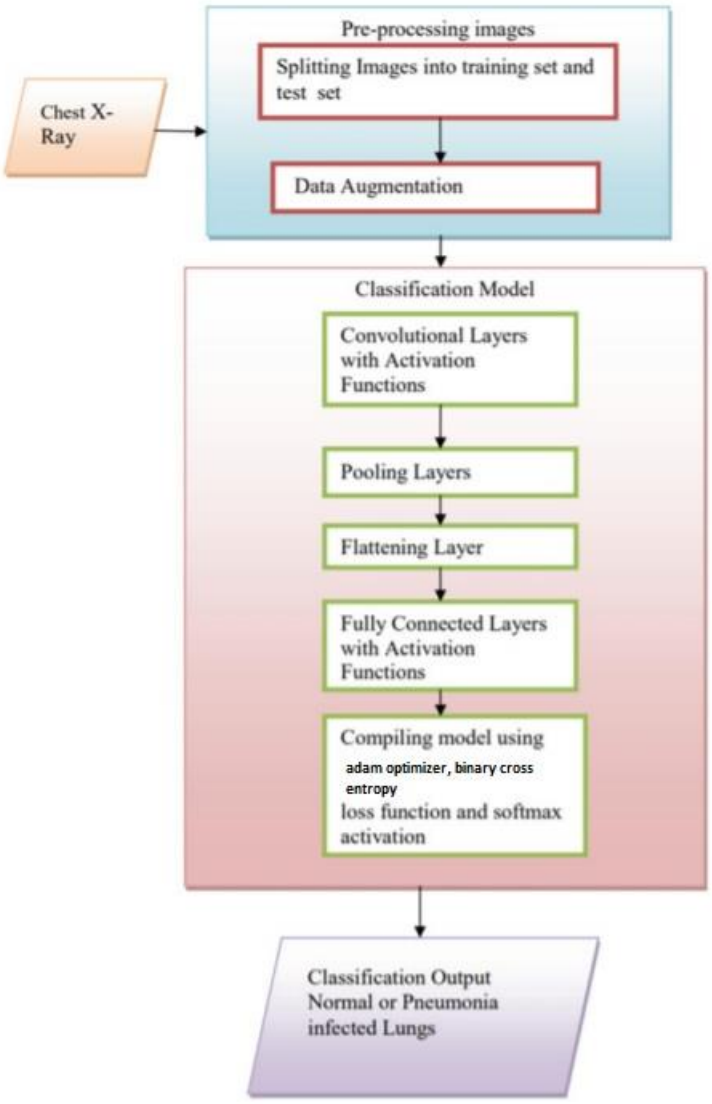

Fig 3.2 Detailed schema of the Convolutional neural network

trained and studied. Adam optimizer function was finalized to be used for all classifiers after it gave the best results. Initially, a simple classifier model with convolutional layer of image size set to 64 * 64, 32 feature maps and employing ReLU activation function was trained. Fully connected dense layer with 32 perceptrons was utilized. To improve the result, This model was trained for three

convolutional layers with 32 feature maps in third convolutional layer for more detailed feature extraction. Dense layer was kept unchanged. Dropout layer was introduced at 0.5, and learning rate of optimizer was lowered to 0.0001 to reduce the overfitting.

3.2.3 Dataset:

Chest X-Ray Images (Pneumonia) dataset of 1.16 GB size has been imported from Kaggle, with total of 2916 jpeg images split into Train, Test and Val folders each divided into category Pneumonia and Normal. Chest X-ray images (front and back) were selected from pediatric patients of one- to five-year olds from Guangzhou Women and Children's Medical Center, Guangzhou. Figure 4.3 provides the sample images from the dataset used during the research.

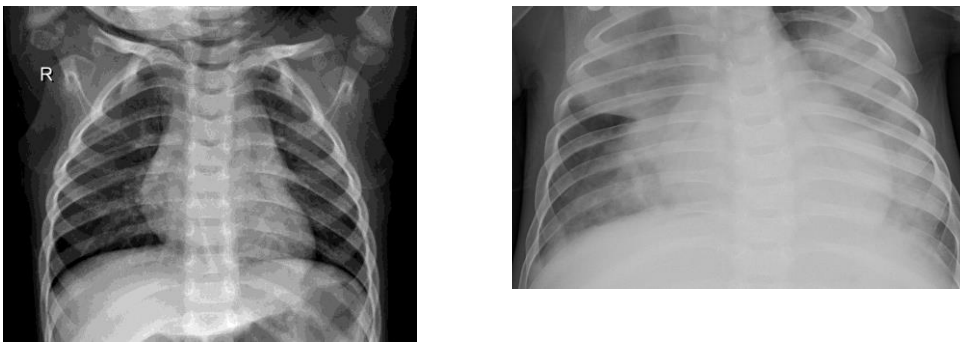

Fig 3.3 left image depicts normal lungs and right image depicts pneumonic lungs

CHAPTER 4

SOFTWARE REQUIREMENT

4.1 INTRODUCTION TO PYTHON:

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object oriented approach aim 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 structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

4.2 FEATURES OF PYTHON:

Python provides many useful features which make it popular and valuable from the other programming languages. It supports object-oriented programming; procedural programming approaches and provides dynamic memory allocation. We have listed below a few essential features.

Easy to Learn and Use

Python is easy to learn as compared to other programming languages. Its syntax is straightforward and much the same as the English language. There is no use of the semicolon or curly-bracket, the indentation defines the code block. It is the recommended programming language for beginners.

Expressive Language

Python can perform complex tasks using a few lines of code. A simple example, the hello world program you simply type print("Hello World"). It will take only one line to execute, while Java or C takes multiple lines. The open-source means, "Anyone can download its source code without paying any penny."

Object-Oriented Language

Python supports object-oriented language and concepts of classes and objects come into existence. It supports inheritance, polymorphism, and encapsulation, etc. The

object-oriented procedure helps to programmer to write reusable code and develop applications in less code.

Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our Python code. It converts the program into byte code, and any platform can use that byte code.

Large Standard Library

It provides a vast range of libraries for the various fields such as machine learning, web developer, and also for the scripting. There are various machine learning libraries, such as Tensor flow, Pandas, NumPy, Keras, and Pytorch, etc. Django, flask, pyramids are the popular framework for Python web development.

GUI Programming Support

Graphical User Interface is used for the developing Desktop application. PyQT5, Tkinter, Kivy are the libraries which are used for developing the web application.

Integrated

It can be easily integrated with languages like C, C++, and JAVA, etc. Python runs code line by line like C, C++ Java. It makes easy to debug the code.

Embeddable

The code of the other programming language can use in the Python source code. We can use Python source code in another programming language as well. It can embed other language into our code.

Dynamic Memory Allocation

In Python, we don't need to specify the data-type of the variable. When we assign some value to the variable, it automatically allocates the memory to the variable at runtime. Suppose we are assigned integer value 15 to x, then we don't need to write int x = 15. Just write x = 15.

Interpreted Language

Python is an interpreted language; it means the Python program is executed one line at a time. The advantage of being interpreted language, it makes debugging easy and portable.

Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, UNIX, and Macintosh, etc. So, we can say that Python is a portable language. It enables programmers to develop the software for several competing platforms by writing a program only once.

Free and Open Source

Python is freely available for everyone. It is freely available on its official website www.python.org. It has a large community across the world that is dedicatedly working towards make new python modules and functions. Anyone can contribute to the Python community. The open-source means, "Anyone can download its source code without paying any penny."

4.3 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.
- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

4.3.1 Installation(using python (IDLE)):

- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
- Download python from www.python.org.
- When the download is completed, double click the file and follow the instructions to install it.
- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python
- Go to the file where your project is to be stored and select the path of the folder and type cmd then click enter.

- Command prompt will appears, next type jupyter notebook and click enter
- Jupyter notebook will be opened default browser.

4.4 PYTHON VARIABLE TYPES:

- Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
- Variables are nothing but reserved memory locations to store values.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.
- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.
- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
- Python has five standard data types –
  1. Numbers
  2. Strings
  3. Lists
  4. Tuples
  5. Dictionary

4.4.1 Python Numbers:

- Number data types store numeric values. Number objects are created when you assign a value to them.
- Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

4.4.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

4.4.3 Python Lists:

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets ([]).
- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([] and[:]) with indexes starting at 0 in the beginning of the list and working theirway to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

4.4.4 Python Tuples:

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.
- Tuples can be thought of as read-only lists.
- For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples

CHAPTER 5

RESULTS

- Visualizing the data in the training directory using matplot library in python in Fig 5.1.
- There are 1566 images in Pneumonia class indicated with ‘1’ and 1341 images in normal class indicated with ‘0’.
- The numbers has been shown as bar chat in Fig 5.2.

```
In [57]: cases_count=train_data['label'].value_counts()
         print(cases_count)
         plt.figure(figsize=(10,8))
         sns.barplot(x=cases_count.index,y=cases_count.values)
         plt.title('Number of cases',frontsize=14)
         plt.xlabel('case type',fontsize=12)
         plt.ylabel('count',fontsize=12)
         plt.xticks(range(len(cases_count.index)),['Normal(0)','Pneumonia(1)'])
         plt.show()

1    1566
0    1341
Name: label, dtype: int64
```

Fig 5.1 count of images in training directory

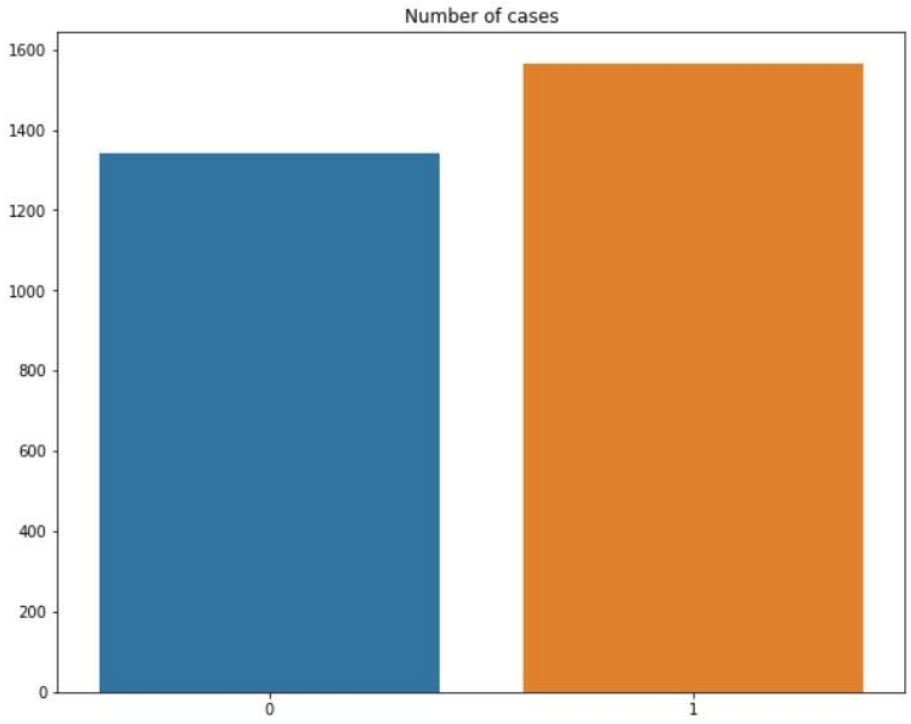

Fig 5.2 Bar graph showing the count of normal(0) and pneumonia(1) cases

- We used sequential model in keras and fitting the model has been completed.
- To fit the model we use fit_generator function with a 25 epochs.
- Input is train_set and validation data is test_set.
- For each epoch it gives 4 output Loss, Accuracy, Val_loss, Val_accuracy.
- On average each epoch took 55 seconds to execute.
- Val_accuracy is **0.86**, which is a good score.

```
Epoch 1/25
91/91 [==============================] - 44s 479ms/step - loss: 0.6027 - accuracy: 0.6784 - val_loss: 22.3141 - val_accuracy:
0.8542
Epoch 2/25
91/91 [==============================] - 43s 472ms/step - loss: 0.3627 - accuracy: 0.8400 - val_loss: 32.3698 - val_accuracy:
0.8462
Epoch 3/25
91/91 [==============================] - 46s 501ms/step - loss: 0.3020 - accuracy: 0.8772 - val_loss: 44.8687 - val_accuracy:
0.8510
Epoch 4/25
91/91 [==============================] - 47s 518ms/step - loss: 0.2828 - accuracy: 0.8879 - val_loss: 36.2445 - val_accuracy:
0.8446
Epoch 5/25
91/91 [==============================] - 43s 472ms/step - loss: 0.2486 - accuracy: 0.9099 - val_loss: 44.5867 - val_accuracy:
0.8606
Epoch 6/25
91/91 [==============================] - 45s 500ms/step - loss: 0.2389 - accuracy: 0.9126 - val_loss: 60.0845 - val_accuracy:
0.8478
Epoch 7/25
91/91 [==============================] - 46s 507ms/step - loss: 0.2396 - accuracy: 0.9102 - val_loss: 59.1197 - val_accuracy:
0.8622
Epoch 8/25
91/91 [==============================] - 46s 508ms/step - loss: 0.2124 - accuracy: 0.9257 - val_loss: 77.4524 - val_accuracy:
0.8558
Epoch 9/25
91/91 [==============================] - 47s 514ms/step - loss: 0.2008 - accuracy: 0.9284 - val_loss: 39.3482 - val_accuracy:
0.8814
Epoch 10/25
91/91 [==============================] - 45s 492ms/step - loss: 0.2144 - accuracy: 0.9188 - val_loss: 66.8569 - val_accuracy:
0.8622
Epoch 11/25
91/91 [==============================] - 45s 499ms/step - loss: 0.1911 - accuracy: 0.9336 - val_loss: 45.0363 - val_accuracy:
0.8686
Epoch 12/25
91/91 [==============================] - 47s 512ms/step - loss: 0.1739 - accuracy: 0.9426 - val_loss: 57.8757 - val_accuracy:
0.8830
Epoch 13/25
91/91 [==============================] - 46s 502ms/step - loss: 0.1838 - accuracy: 0.9329 - val_loss: 113.2128 - val_accuracy:
0.8189
Epoch 14/25
91/91 [==============================] - 46s 508ms/step - loss: 0.1789 - accuracy: 0.9302 - val_loss: 51.3048 - val_accuracy:
0.8814
Epoch 15/25
91/91 [==============================] - 45s 497ms/step - loss: 0.1682 - accuracy: 0.9408 - val_loss: 70.0097 - val_accuracy:
0.8798
Epoch 16/25
91/91 [==============================] - 47s 516ms/step - loss: 0.1743 - accuracy: 0.9415 - val_loss: 107.0485 - val_accuracy:
0.8462
Epoch 17/25
91/91 [==============================] - 45s 495ms/step - loss: 0.1748 - accuracy: 0.9357 - val_loss: 67.4178 - val_accuracy:
0.8862
Epoch 18/25
91/91 [==============================] - 45s 493ms/step - loss: 0.1708 - accuracy: 0.9374 - val_loss: 108.9394 - val_accuracy:
0.8269
Epoch 19/25
91/91 [==============================] - 44s 487ms/step - loss: 0.1679 - accuracy: 0.9401 - val_loss: 56.1159 - val_accuracy:
0.8814
Epoch 20/25
91/91 [==============================] - 46s 502ms/step - loss: 0.1596 - accuracy: 0.9429 - val_loss: 63.2854 - val_accuracy:
0.8910
Epoch 21/25
91/91 [==============================] - 45s 497ms/step - loss: 0.1568 - accuracy: 0.9436 - val_loss: 104.1487 - val_accuracy:
0.8750
Epoch 22/25
91/91 [==============================] - 47s 512ms/step - loss: 0.1638 - accuracy: 0.9388 - val_loss: 114.9646 - val_accuracy:
0.8526
Epoch 23/25
91/91 [==============================] - 46s 506ms/step - loss: 0.1548 - accuracy: 0.9463 - val_loss: 66.0200 - val_accuracy:
0.8846
Epoch 24/25
91/91 [==============================] - 47s 514ms/step - loss: 0.1562 - accuracy: 0.9426 - val_loss: 41.7178 - val_accuracy:
0.8830
Epoch 25/25
91/91 [==============================] - 46s 506ms/step - loss: 0.1586 - accuracy: 0.9529 - val_loss: 112.4254 - val_accuracy:
0.8574
```

Fig 5.3 fitting of the model

Visualization of loss and val_accuracy:

- Matplotlib plots are used to visualize accuracy, losses of train and test sets

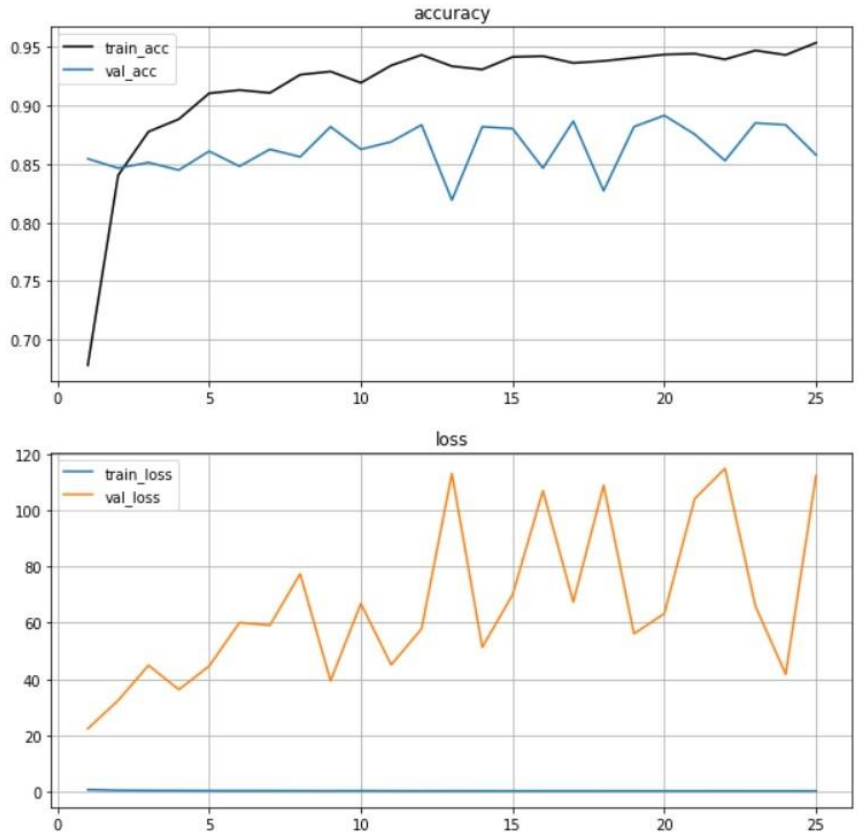

Fig 5.4 visualization of val_loss and val_accuracy

- It is observed that the model obtained good accuracy scores.

Prediction of Pneumonia class:

```
In [69]: print(model.predict_classes(img)[0][0])
         print(f'The Photo is in the category of {mapping[model.predict_classes(img)[0][0]]}')

         WARNING:tensorflow:From <ipython-input-69-c0e9f26a6c5b>:1: Sequential.predict_classes (from tensorflow.python.keras.engine.sequential) is deprecated and will be removed after 2021-01-01.
         Instructions for updating:
         Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` last-layer activation).
         1
         The Photo is in the category of PNEUMONIA
```

```
model.predict(img)

array([[0.94018793]], dtype=float32)
```

Fig 5.5 Program to predict the pneumonia case

- Predict function gives the probability of the image belonging to particular category
- Predict_classes give the name of the class it belongs to
- Pneumonia image is classified correctly

Prediction of Normal class:

```
print(model.predict_classes(img)[0][0])
print(f'The Photo is in the category of {mapping[model.predict_classes(img)[0][0]]}')
```

```
0
The Photo is in the category of NORMAL
```

```
In [71]: model.predict(img)
Out[71]: array([[0.3027675]], dtype=float32)
```

Fig 5.6 Program to predict the normal case

- Predict function gives the probability of the image belonging to particular category
- Predict_classes give the name of the class it belongs to
- We got score as 0.02 which is less than 0.5 so it belongs to 0 category i.e. Normal class.
- Normal image is classified correctly

Analyzes of Matrix:

- Since the data is unbalanced, we cannot take accuracy alone as metric for validating the model.
- So, we need to choose another metric to validate the model.
- Let us plot a confusion matrix to visualize the true positives, true negatives, False positives and False negatives.
- We need to import confusion matrix, classification report and f1_score from sklear. metrics .
- Store all the test data prediction in Y_pred .
- Plotting the confusion matrix with test_set.classes and y_pred as inputs.
- Out of 234 normal class images 150 images are correctly classified, 84 are incorrectly classified.

- Out of 390 Pneumonia images 385 images are correctly classified, 5 are incorrectly classified.
- It is observed that Pneumonia class images are more accurately classified than the images of normal class.

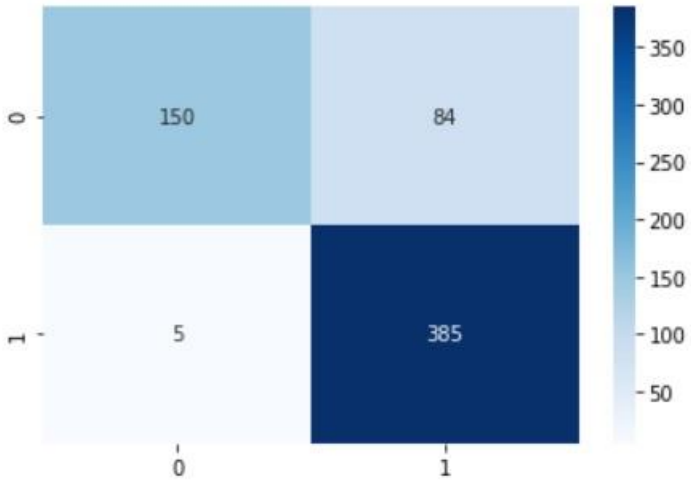

Fig 5.7 Confusion Matrix

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

Throughout the process of developing the CNN model for Pneumonia prediction, I have built a model from scratch which consists of 4 layers and follows with 2 fully connected neural network. Then the trained model is evaluated using separate unseen data to avoid bias prediction. As the result, the accuracy of the test dataset reached 85% which indicates a decent model. This project allows a beginner to obtain an overview of how to build a model to solve a real-world problem. It is no doubt that the predictive model can be improved even better by performing data augmentation or implementing a transfer learning concept which facilitates the model a room for improvement. Therefore, this will be added as further enhancement in the upcoming stories.

REFERENCES

[1] About Convolutional neural network
https://en.wikipedia.org/wiki/Convolutional_neural_network

[2] Data sets from kaggle
https://www.kaggle.com

[3] Anand Nayyar, Gaurav Kataria, Rachna Jain, V. Sirish Kaushik :: Pneumonia detection using CNN
https://www.researchgate.net/publication/340961287_Pneumonia_Detection_Using_Convolutional_Neural_Networks_CNNs

[4] https://data.unicef.org/topic/child-health/pneumonia/ . for data related to Pneumonia.

[5] Ankush Mittal; Dimpy Varshni; Kartik Thakral; Lucky Agarwal; Rahul Nijhawan; :: Pneumonia Detection Using CNN based feature extraction. 10.1109/ ICECCT.2019. 8869364

[6] Abiyev R. H., Ma'aitah M. K. S., Deep Convolutional Neural Networks for Chest Diseases Detection, Journal of Healthcare Engineering, 2018.

[7] A. Shrestha and A. Mahmood, Review of Deep Learning Algorithms and Architectures, in IEEE Access, vol. 7, pp. 53040-53065, 2019, DOI: 10.1109/ACCESS.2019.2912200.

[8] Guan, Q., Huang, Y., Zhong, Z., Zheng, Z., Zheng, L., Yang, Y.: Diagnose Like a Radiologist: Attention Guided Convolutional Neural Network for Thorax Disease Classification (2018). arXiv preprint arXiv:1801.09927

[9] Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., Lungren, M.P.: Chexnet: Radiologist-Level Pneumonia Detection on Chest X-rays with Deep Learning (2017). arXiv preprint arXiv:1711.05225

[10] Krizhevsky, A., Sutskever, I., Hinton, G.E:. Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)

[11] Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition (2014). arXiv preprint arXiv:1409.1556

[12] Xu, Y., Jia, Z., Ai, Y., Zhang, F., Lai, M., Eric, I., Chang, C.: Deep convolutional activation features for large scale brain tumor histopathology image classification and segmentation. In: 2015 international conference on acoustics, speech and signal processing (ICASSP), pp. 947– 951 (2015)
[13] Anthimopoulos, M., Christodoulidis, S., Ebner, L., Christe, A., Mougiakakou, S.: Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. IEEE Trans. Med. Imaging 35(5), 1207–1216 (2016)
[14] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

APPENDIX

Code for the model

```
import numpy as np
import pandas as pd
import tensorflow as tf
import os
from keras_preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib,keras
from pathlib import Path
import PIL
from PIL import Image
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,Dense,Dropout,Flatten,MaxPooling2D
#define path to the data directory
data_dir=Path('G:\project\chest_xray\chest_xray')
#path to train directory
train_dir=data_dir/'train'
#path to the validation directory
val_dir=data_dir/'val'
#path to the test directory
test_dir=data_dir/'test'
#categories in the data set
categories=['NORMAL','PNEUMONIA']
mapping={}
count=0
for i in categories:
    mapping[count]=i
    count+=1
#Checking the no. of images in each subfolder of [‘train’,’val’,’test’]
print(len(os.listdir(train_dir/'PNEUMONIA')))
print(len(os.listdir(train_dir/'NORMAL')))
```

```
print(len(os.listdir(test_dir/'PNEUMONIA')))
print(len(os.listdir(test_dir/'NORMAL')))
print(len(os.listdir(val_dir/'PNEUMONIA')))
print(len(os.listdir(val_dir/'NORMAL')))
#Applying data augumentation using ImageDataGenerator of keras for training
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,
horizontal_flip=True
training_set=train_datagen.flow_from_directory(train_dir,target_size=(64,64),
batch_size=32,class_mode='binary')
test_datagen=ImageDataGenerator()
test_set=test_datagen.flow_from_directory(test_dir,shuffle=False,target_size=(64,64),
batch_size=32,class_mode='binary')
val_datagen=ImageDataGenerator()
val_set=val_datagen.flow_from_directory(val_dir,shuffle=False,target_size=(64,64),
batch_size=32,class_mode='binary')
#Get the path to the normal and pneumonia sub-directories
train_normal_dir=train_dir/'NORMAL'
train_pneumonia_dir=train_dir/'PNEUMONIA'
#Get the list of all the images
#glob function get the pathnames matching a specified pattern(. Jpeg, etc..)
normal_cases=train_normal_dir.glob('*.jpeg')
pneumonia_cases=train_pneumonia_dir.glob('*.jpeg')
#An empty list. We will insert the data into this list in (img_path, label) format
train_data=[]
#Go through all the normal cases. The label for these cases will be 0
for img in normal_cases:
    train_data.append((img,0))
#Go through all the pneumonia cases. The label for these cases will be 1
for img in pneumonia_cases:
    train_data.append((img,1))
#Get a pandas data frame form the data we have in our list
train_data=pd.DataFrame(train_data,columns=['image','label'],index=None)
train_data.shape()
#Creating an instance of sequential model(model)
```

```
model=Sequential()
#Adding a first convolutional layer
#First convolution extracts 32 filters that are of size(3,3)
model.add(Conv2D(32,(3,3),input_shape=(64,64,3),activation='relu'))
#Convolution is followed by max_pooling layer with a 2x2 window
model.add(MaxPooling2D(pool_size=(2,2)))
#Adding a second convolutional layer
#First convolution extracts 32 filters that are of size(3,3)
model.add(Conv2D(32,(3,3),activation='relu'))
#Convolution is followed by max_pooling layer with a 2x2 window
model.add(MaxPooling2D(pool_size=(2,2)))
#Adding a second convolutional layer
#First convolution extracts 64 filters that are of size(3,3)
model.add(Conv2D(64,(3,3),activation='relu'))
#Convolution is followed by max_pooling layer with a 2x2 window
model.add(MaxPooling2D(pool_size=(2,2)))
#Flatten feature map to a 1-D tensor so we can add fully connected layers
model.add(Flatten())
#Create a fully connected layer with ReLU activation and 64 hidden units
model.add(Dense(units=64,activation='relu'))
model.add(Dropout(0.5))
#Create output layer with a single neuron and sigmoid activation
model.add(Dense(units=1,activation='sigmoid'))
#Compiling the CNN
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
#Fitting the generated model on training data and validation data
history=model.fit_generator(training_set,epochs=25,validation_data=test_set)
train_acc=history.history['accuracy']
val_acc=history.history['val_accuracy']
train_loss=history.history['loss']
val_loss=history.history['val_loss']
epochs=list(range(1,26))
#Plot 1 for visualizing train_acc, val_acc
plt.figure(figsize=(10,10))
```

```
plt.subplot(2,1,1)
plt.plot(epochs,train_acc,label='train_acc',color='black')
plt.plot(epochs,val_acc,label='val_acc')
plt.title('accuracy')
plt.grid()
plt.legend()
#Plot 2 for visualizing train_loss,val_loss
plt.subplot(2,1,2)
plt.plot(epochs,train_loss,label='train_loss')
plt.plot(epochs,val_loss,label='val_loss')
plt.title('loss')
plt.grid()
plt.legend()
from tensorflow.keras.preprocessing import image
import numpy as np
img=image.load_img(val_dir/'PNEUMONIA'/'person1946_bacteria_4874.jpeg')
print(type(img))
plt.imshow(img)
img=tf.keras.preprocessing.image.img_to_array(img)
print(img.shape)
print(type(img))
img=tf.image.resize(img,(64,64))
#Scaling
img=img/255
print(img.shape)
img=np.expand_dims(img,axis=0)
print(img.shape)
#Predicting the image using predict function
model.predict(img)
#Predicting the class using predict_class function
print(model.predict_classes(img)[0][0])
print(f'The Photo is in the category of {mapping[model.predict_classes(img)[0][0]]}')
img=image.load_img(val_dir/'NORMAL'/'NORMAL2-IM-1440-0001.jpeg')
print(type(img))
```

```
plt.imshow(img)
img=tf.keras.preprocessing.image.img_to_array(img)
print(img.shape)
print(type(img))
img=tf.image.resize(img,(64,64))
#Scaling
img=img/255
print(img.shape)
img=np.expand_dims(img,axis=0)
print(img.shape)
#Predicting the image using predict function
model.predict(img)
#Predicting the class using predict_class function
print(model.predict_classes(img)[0][0])
print(f'The Photo is in the category of {mapping[model.predict_classes(img)[0][0]]}')
from sklearn.metrics import classification_report,confusion_matrix,f1_score
Y_pred=model.predict(test_set)
y_pred=np.where(Y_pred>0.50,1,0)
print(confusion_matrix(test_set.classes,y_pred))
sns.heatmap(confusion_matrix(test_set.classes,y_pred),annot=True,fmt='d',cmap='Blu
es')
```