CSE3042

Machine Intelligence for Medical Image Analysis

Covid19 and Pnuemonia Dection

Faculty: Dr Malathi G

Submitted by

EMMAY KOUSHAL 20BAI1210

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Abstract: Covid19 and pneumonia are two of the most prevalent respiratory diseases that have been causing a significant impact worldwide. The early diagnosis and detection of these diseases are essential for prompt medical treatment and efficient control of the spread of the diseases. One of the commonly used methods for diagnosing these diseases is through chest X-ray images, which provide an insight into the respiratory system of the patient. The process involves capturing the images of the patient's chest through an X-ray machine, which creates a digital image that shows the lungs' internal structure. In recent years, machine learning algorithms have been developed to analyze these chest X-ray images for accurate and efficient detection of Covid19 and pneumonia. These algorithms use deep learning techniques, such as convolutional neural networks (CNNs), to extract features from the X-ray images and then classify the images as either Covid19 positive, pneumonia positive, or negative. The training of these algorithms involves feeding them a large dataset of labeled chest X-ray images to learn the features that differentiate between Covid19 and pneumonia images from normal lung images. The models are then fine-tuned to achieve high accuracy, sensitivity, and specificity in predicting the diseases. The prediction process involves inputting a new chest X-ray image into the trained machine learning model, which then processes the image and predicts the likelihood of the presence of Covid19 or pneumonia. The predictions are made based on the features extracted from the image and the patterns learned from the training dataset.

In conclusion, the use of chest X-ray images and machine learning algorithms provides an efficient and accurate method for the early detection and diagnosis of Covid19 and pneumonia. With further development and improvement of these algorithms, the medical community can utilize them to enhance the speed and accuracy of diagnosing respiratory diseases, leading to better patient outcomes and effective disease control.

Introduction: Pneumonia is a common and potentially life-threatening lung infection that affects millions of people worldwide. Early detection and accurate diagnosis of pneumonia is crucial for effective treatment and management of the disease. One of the most common diagnostic tools for pneumonia is the use of chest X-rays, which can reveal characteristic abnormalities in the lungs. In recent years, there has been a growing interest in using artificial intelligence (AI) to develop computer-based models that can accurately detect pneumonia from chest X-ray images. These models use deep learning algorithms to analyze large datasets of chest X-rays and learn to recognize the patterns and features associated with pneumonia.

The goal of this project is to understand different image preprocessing techniques, make a model which performs better than the existing models, comparing them and explore the different type of approaches other than Deep Learning networks. In this project different models are made. Different types of preprocessing is done on images before the training of models. All the results are recorded and presented in section Results and Discussion.

Literature Survey: Pneumonia is a leading cause of morbidity and mortality worldwide, and early detection and accurate diagnosis are crucial for successful treatment and patient outcomes. Deep learning techniques have been increasingly applied in the field of medical imaging to assist clinicians in detecting and diagnosing pneumonia. This literature survey aims to provide an overview of recent research on pneumonia detection using deep learning techniques.

"A Survey on Deep Learning in Medical Image Analysis" by Litjens et al. (2017) [1]
This comprehensive survey provides an overview of deep learning techniques in medical image analysis, including pneumonia detection. The authors discuss various deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), and their applications in medical imaging.

"Deep learning for automated pneumonia detection in chest X-rays: A survey" by Razzak et al. (2018) [2]
This survey provides a detailed analysis of deep learning techniques for pneumonia detection in chest X-ray images. The authors review various deep learning models, including CNNs, and discuss the advantages and limitations of these models.

"Pneumonia Detection Using Deep Learning from X-Ray Images: A Comprehensive Review" by Kumar et al. (2021) [3]
This comprehensive review provides an overview of deep learning techniques for pneumonia detection from chest X-ray images. The authors discuss various deep learning models, including CNNs and their variations, and provide a detailed analysis of the strengths and limitations of these models.

"Deep Learning for Medical Image Analysis: A Review" by Shen et al. (2017) [4]
This review provides a comprehensive analysis of deep learning techniques in medical image analysis, including pneumonia detection. The authors discuss various deep learning models, such as CNNs and RNNs, and their applications in medical imaging.

"Detection of Pneumonia in Chest X-Ray Images using Deep Learning: A Review" by Thakur and Suman (2021) [5]
This review provides an overview of recent research on pneumonia detection in chest X-ray images using deep learning techniques. The authors discuss various deep learning models, including CNNs and their variations, and provide a detailed analysis of the strengths and limitations of these models.

"A Deep Learning Framework for Pneumonia Detection in Chest X-Ray Images" by Zheng et al. (2020) [6]
This study proposed a deep learning framework based on a combination of CNNs and RNNs for pneumonia detection in chest X-ray images. The framework achieved high accuracy and was able to detect pneumonia at an early stage.

"Pneumonia Detection using Convolutional Neural Networks" by Yoo et al. (2019) [7]
This study proposed a pneumonia detection algorithm based on CNNs that outperformed traditional machine learning algorithms. The algorithm was trained on a large dataset of chest X-ray images and achieved high accuracy in detecting pneumonia.

"A Deep Learning Approach to Pneumonia Detection in Chest X-Ray Images" by Rajpurkar et al. (2018) [8]
In this study, the authors developed a deep learning algorithm to detect pneumonia in chest X-ray images. The algorithm achieved high accuracy and outperformed human radiologists in certain cases.

In conclusion, deep learning techniques have shown promising results in detecting and diagnosing pneumonia from chest X-ray images. Further research is needed to validate these algorithms in real-world clinical settings and to explore their potential in other medical imaging applications.

Methodologies:

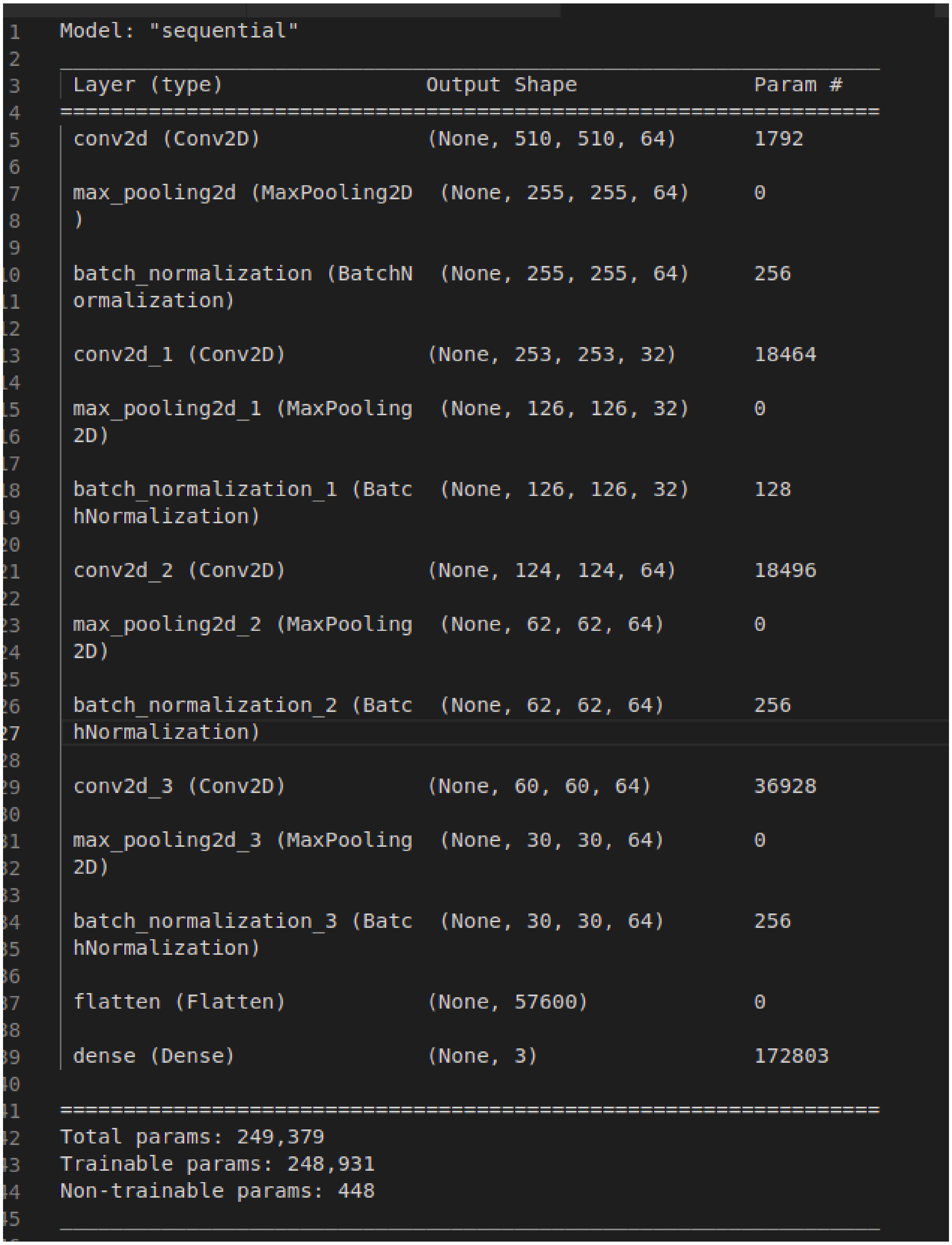
Since the main goal was to explore all preprocessing techniques and implementing different models. There are 6 approaches in this project. Each approach has different methods.

Method1:

Since input is image files which are chest x-ray images. Method1 is made with custom Convolutional Nueral Network. Architecture of the the model is as follows

- Conv2D—it is a convolution layer with 64 filters with dimensions of 3 × 3 and the activation function ReLU, the dimensions of the input data is also introduced
- MaxPooling with a size of 2 × 2
- BatchNormalization
- Conv2D—it is a convolution layer with 32 filters with dimensions of 3 × 3 and the activation
- MaxPooling with a size of 2 × 2
- BatchNormalization
- Conv2D—it is a convolution layer with 64 filters with dimensions of 3 × 3 and the activation function ReLU
- MaxPooling with a size of 2 × 2
- BatchNormalization
- Flatten—it is a data flattening layer, it has no additional parameters
- Dense—with unit parameters equal to 3, with softmax activation function.
- Dense—with unit parameters equal to 3, with softmax activation function.

This model has total of 249,379 parameters in which there are 248,931 trainable parameters and 448 non trainable parameters. This model is compiled with categorical_crossentropy loss, adam optimizer and accuracy metrics. 20% of the data is used as validation data from training data.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 510, 510, 64)      1792

 max_pooling2d (MaxPooling2D  (None, 255, 255, 64)     0
 )

 batch_normalization (BatchN  (None, 255, 255, 64)     256
 ormalization)

 conv2d_1 (Conv2D)           (None, 253, 253, 32)      18464

 max_pooling2d_1 (MaxPooling  (None, 126, 126, 32)     0
 2D)

 batch_normalization_1 (Batc  (None, 126, 126, 32)     128
 hNormalization)

 conv2d_2 (Conv2D)           (None, 124, 124, 64)      18496

 max_pooling2d_2 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 batch_normalization_2 (Batc  (None, 62, 62, 64)       256
 hNormalization)

 conv2d_3 (Conv2D)           (None, 60, 60, 64)        36928

 max_pooling2d_3 (MaxPooling  (None, 30, 30, 64)       0
 2D)

 batch_normalization_3 (Batc  (None, 30, 30, 64)       256
 hNormalization)

 flatten (Flatten)           (None, 57600)             0

 dense (Dense)               (None, 3)                 172803

=================================================================
Total params: 249,379
Trainable params: 248,931
Non-trainable params: 448

_________________________________________________________________
```

Fig 1

This data has been trained with 5, 10 and 20 epochs individually. The results are recorded. The accuracy in each case are mentioned in table1.

Epochs	**Accuracy**
5	63.64%
10	39.39%
20	48.%

Table 1

Method2:

Since the custom model was very insatisfacoty incase of accuracy. Pretrained architectures has been chosen. I chose VGG16 model to train the same data. Since VGG16 has deep network this will exctract hidden features as well. The same architecture has been implemented without changing layers, kernels etc. The architecture is shown in the Fig 2.

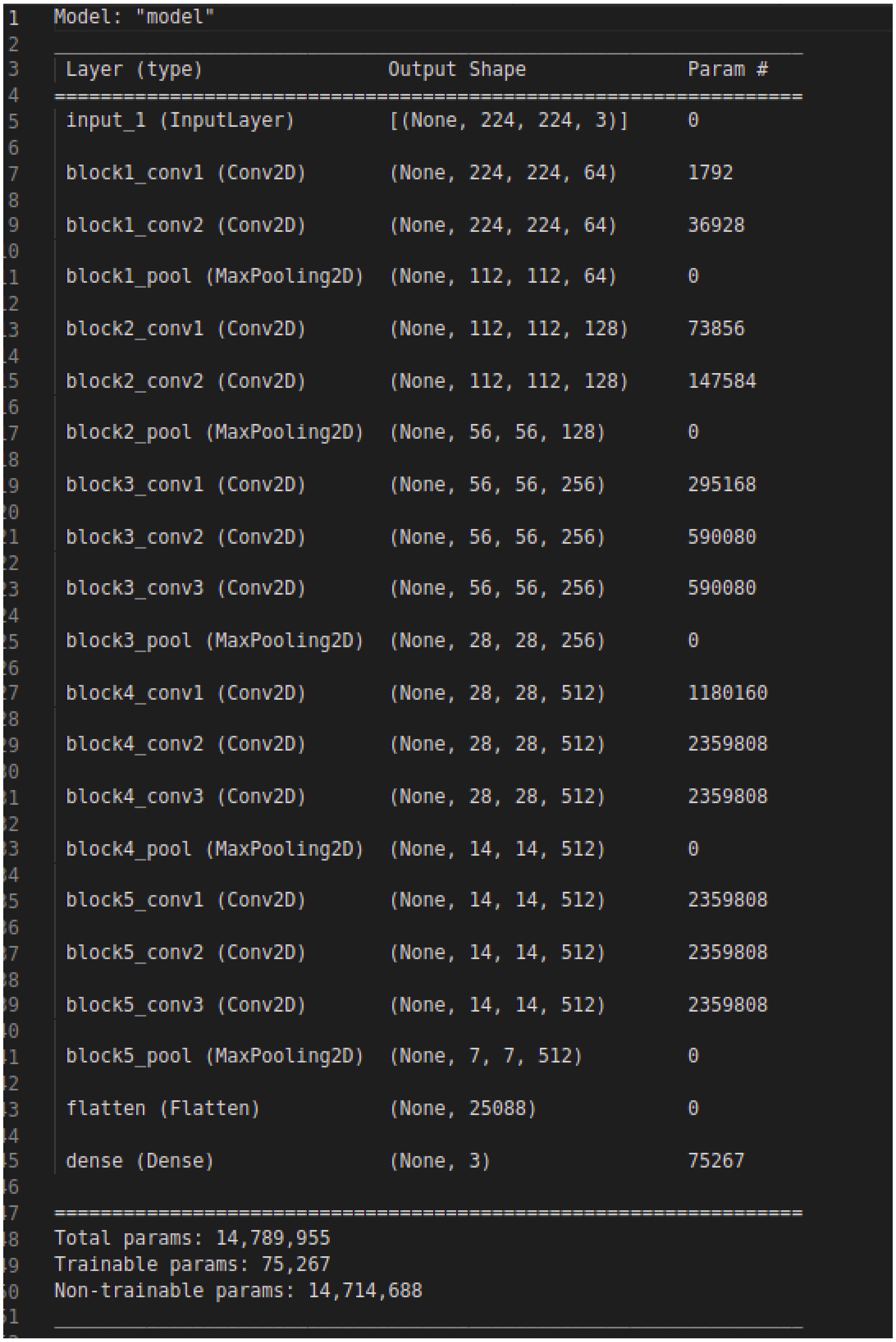
This model has total of `14,789,955` parameters in which there are `75,267` trainable parameters and `14,714,688` non trainable parameters. This model is compiled with categorical_crossentropy loss, adam optimizer and accuracy metrics. 20% of the data is used as validation data from training data.

This data has been trained with 5 and 10epochs individually. The results are recorded. The accuracy in each case are mentioned in table 2.

Epochs	**Accuracy**
5	86.36%
10	100%

Table 2

The accuracy has been increased drastically after shifting from custom CNN to VGG16.



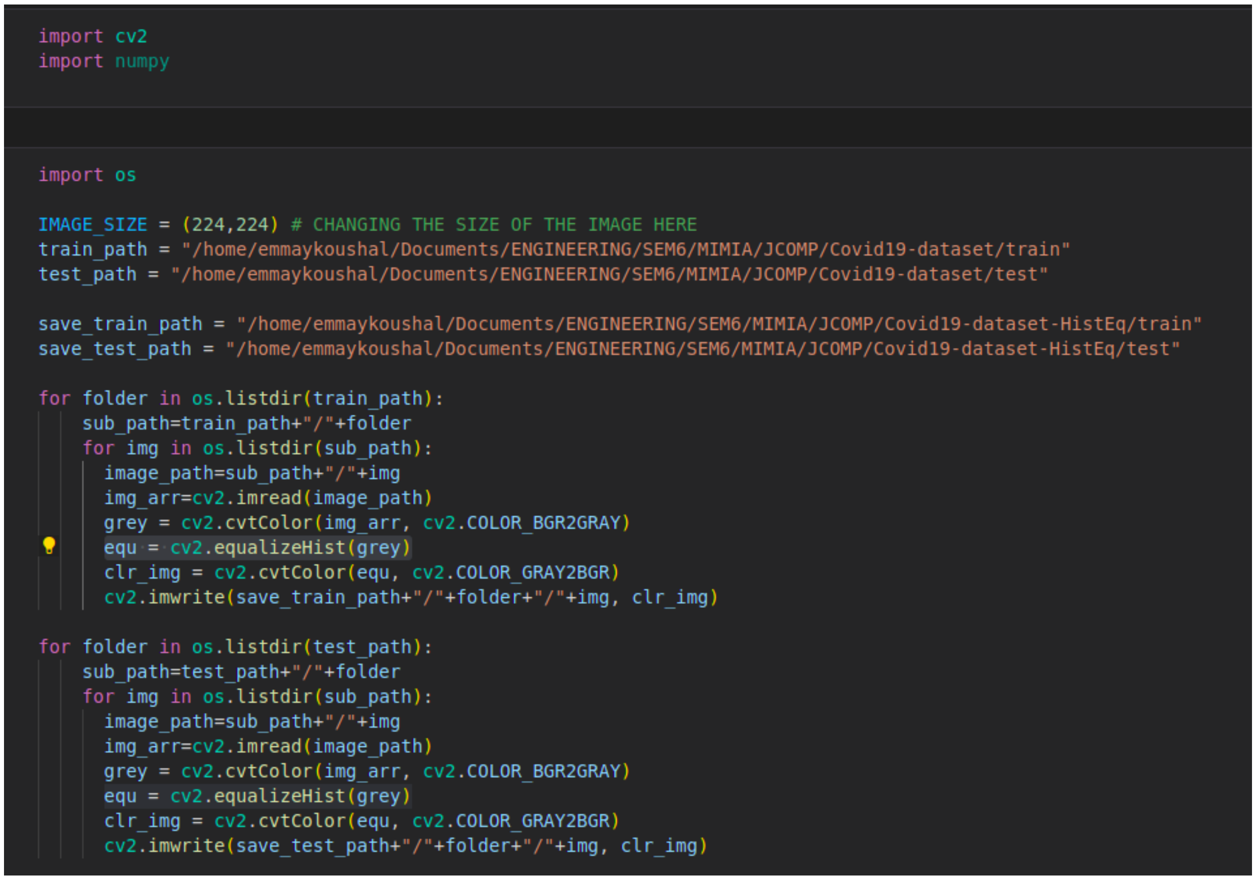
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 3)                 75267

=================================================================
Total params: 14,789,955
Trainable params: 75,267
Non-trainable params: 14,714,688
_________________________________________________________________
```

Fig 2

Method 3:

Even 86% of accuracy is not appreciable incase of medical applications. This model has chance that it can predict in 14% of times. So now rather than changing models and architectures, i chose to preprocess the data so that Nosie can be removed and features will be visible clearly. Contact of the images should be proper in such a way that it can extract features properly. So by understanding the characteristics of pneumonia and finding how does they effect chest X-ray images proper preprocessing techniques have been chosen.

In this method before training the model, images are went through Histogram Equlization then those are given to VGG16 architecture for training. Code for the preprocessing of images with Histogram Equlization is in Fig 3



```
import cv2
import numpy
```

```
import os

IMAGE_SIZE = (224,224) # CHANGING THE SIZE OF THE IMAGE HERE
train_path = "/home/emmaykoushal/Documents/ENGINEERING/SEM6/MIMIA/JCOMP/Covid19-dataset/train"
test_path = "/home/emmaykoushal/Documents/ENGINEERING/SEM6/MIMIA/JCOMP/Covid19-dataset/test"

save_train_path = "/home/emmaykoushal/Documents/ENGINEERING/SEM6/MIMIA/JCOMP/Covid19-dataset-HistEq/train"
save_test_path = "/home/emmaykoushal/Documents/ENGINEERING/SEM6/MIMIA/JCOMP/Covid19-dataset-HistEq/test"

for folder in os.listdir(train_path):
    sub_path=train_path+"/"+folder
    for img in os.listdir(sub_path):
      image_path=sub_path+"/"+img
      img_arr=cv2.imread(image_path)
      grey = cv2.cvtColor(img_arr, cv2.COLOR_BGR2GRAY)
      equ = cv2.equalizeHist(grey)
      clr_img = cv2.cvtColor(equ, cv2.COLOR_GRAY2BGR)
      cv2.imwrite(save_train_path+"/"+folder+"/"+img, clr_img)

for folder in os.listdir(test_path):
    sub_path=test_path+"/"+folder
    for img in os.listdir(sub_path):
      image_path=sub_path+"/"+img
      img_arr=cv2.imread(image_path)
      grey = cv2.cvtColor(img_arr, cv2.COLOR_BGR2GRAY)
      equ = cv2.equalizeHist(grey)
      clr_img = cv2.cvtColor(equ, cv2.COLOR_GRAY2BGR)
      cv2.imwrite(save_test_path+"/"+folder+"/"+img, clr_img)
```

Fig 3

This data has been trained VGG16 with 5 and 10epochs individually. The results are recorded. The accuracy in each case are mentioned in table 3

.

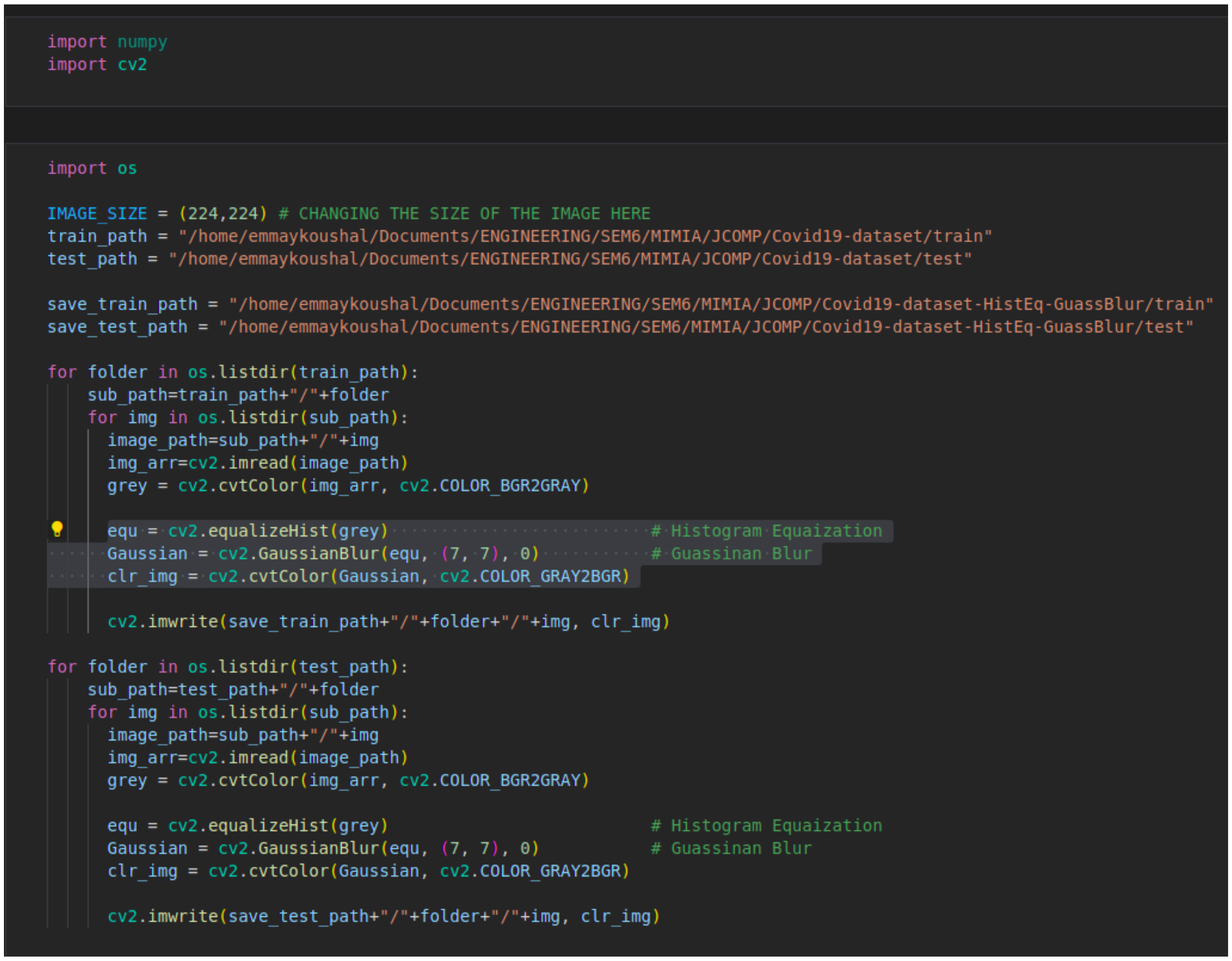
Epochs	Accuracy
5	92.42%
10	95.45%

Table 3

The accuracy has increased to 92.42% which is very good improvement. So this preprocessing has increased accuracy.

Method 4:

In this method similar to previous one before training the model, images are went through Histogram Equlization + Gaussian Blur then those are given to VGG16 architecture for training. Code for the preprocessing of images with Histogram Equlization is in Fig 4



```python
import numpy
import cv2
```

```python
import os

IMAGE_SIZE = (224,224) # CHANGING THE SIZE OF THE IMAGE HERE
train_path = "/home/emmaykoushal/Documents/ENGINEERING/SEM6/MIMIA/JCOMP/Covid19-dataset/train"
test_path = "/home/emmaykoushal/Documents/ENGINEERING/SEM6/MIMIA/JCOMP/Covid19-dataset/test"

save_train_path = "/home/emmaykoushal/Documents/ENGINEERING/SEM6/MIMIA/JCOMP/Covid19-dataset-HistEq-GuassBlur/train"
save_test_path = "/home/emmaykoushal/Documents/ENGINEERING/SEM6/MIMIA/JCOMP/Covid19-dataset-HistEq-GuassBlur/test"

for folder in os.listdir(train_path):
    sub_path=train_path+"/"+folder
    for img in os.listdir(sub_path):
        image_path=sub_path+"/"+img
        img_arr=cv2.imread(image_path)
        grey = cv2.cvtColor(img_arr, cv2.COLOR_BGR2GRAY)

        equ = cv2.equalizeHist(grey)                          # Histogram Equaization
        Gaussian = cv2.GaussianBlur(equ, (7, 7), 0)           # Guassinan Blur
        clr_img = cv2.cvtColor(Gaussian, cv2.COLOR_GRAY2BGR)

        cv2.imwrite(save_train_path+"/"+folder+"/"+img, clr_img)

for folder in os.listdir(test_path):
    sub_path=test_path+"/"+folder
    for img in os.listdir(sub_path):
        image_path=sub_path+"/"+img
        img_arr=cv2.imread(image_path)
        grey = cv2.cvtColor(img_arr, cv2.COLOR_BGR2GRAY)

        equ = cv2.equalizeHist(grey)                          # Histogram Equaization
        Gaussian = cv2.GaussianBlur(equ, (7, 7), 0)           # Guassinan Blur
        clr_img = cv2.cvtColor(Gaussian, cv2.COLOR_GRAY2BGR)

        cv2.imwrite(save_test_path+"/"+folder+"/"+img, clr_img)
```

Fig 4

This data has been trained VGG16 with 5 and 10epochs individually. The results are recorded. The accuracy in each case are mentioned in table 4

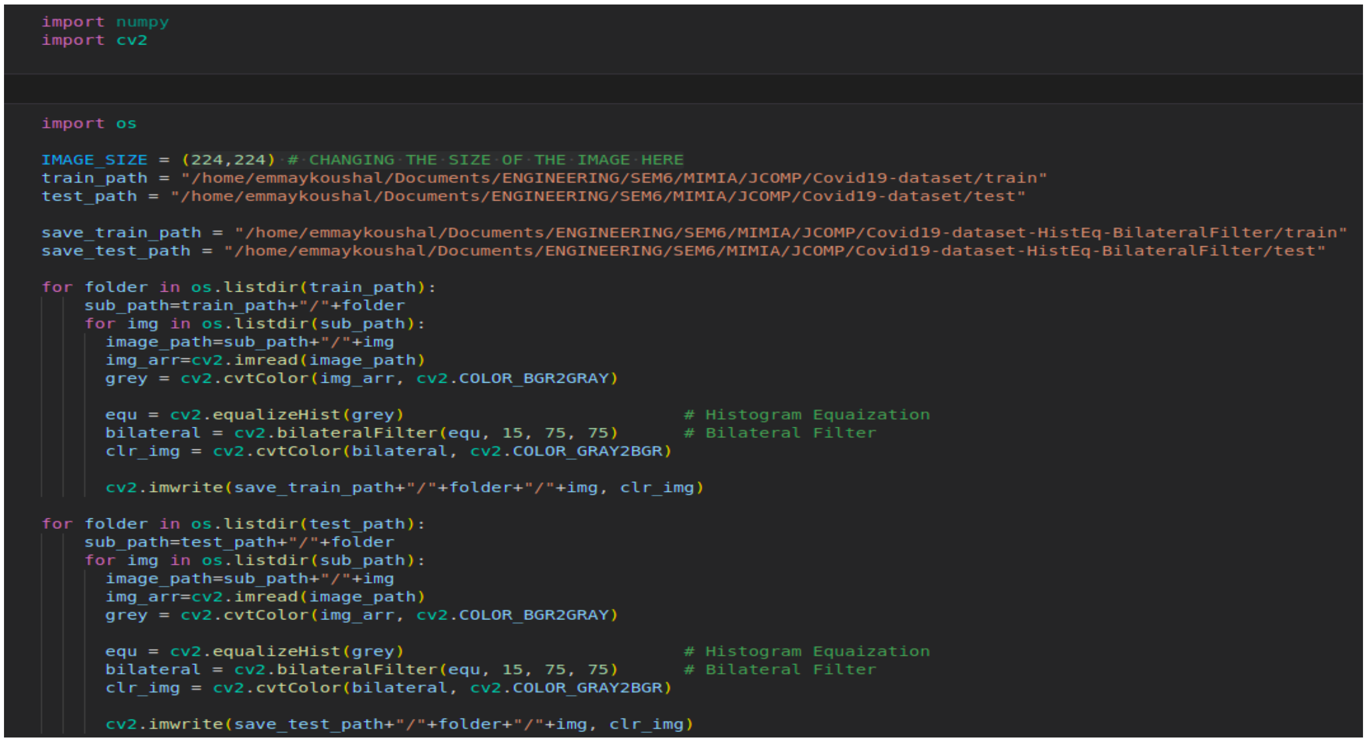
Epochs	Accuracy
5	95.45%
10	98.48%

Table 4

The accuracy has increased to95.45% which is very good improvement. So this preprocessing has increased accuracy. This gaussian blur is used for removing the noise from the image. So since the noise has been removed the accuracy has increased to 95.45%.

Method 5:

In this method similar to previous one before training the model, images are went through Histogram Equlization + Bilateral Fiter then those are given to VGG16 architecture for training. Code for the preprocessing of images with Histogram Equlization is in Fig 5



```
import numpy
import cv2


import os

IMAGE_SIZE = (224,224) # CHANGING THE SIZE OF THE IMAGE HERE
train_path = "/home/emmaykoushal/Documents/ENGINEERING/SEM6/MIMIA/JCOMP/Covid19-dataset/train"
test_path = "/home/emmaykoushal/Documents/ENGINEERING/SEM6/MIMIA/JCOMP/Covid19-dataset/test"

save_train_path = "/home/emmaykoushal/Documents/ENGINEERING/SEM6/MIMIA/JCOMP/Covid19-dataset-HistEq-BilateralFilter/train"
save_test_path = "/home/emmaykoushal/Documents/ENGINEERING/SEM6/MIMIA/JCOMP/Covid19-dataset-HistEq-BilateralFilter/test"

for folder in os.listdir(train_path):
    sub_path=train_path+"/"+folder
    for img in os.listdir(sub_path):
      image_path=sub_path+"/"+img
      img_arr=cv2.imread(image_path)
      grey = cv2.cvtColor(img_arr, cv2.COLOR_BGR2GRAY)

      equ = cv2.equalizeHist(grey)                          # Histogram Equaization
      bilateral = cv2.bilateralFilter(equ, 15, 75, 75)      # Bilateral Filter
      clr_img = cv2.cvtColor(bilateral, cv2.COLOR_GRAY2BGR)

      cv2.imwrite(save_train_path+"/"+folder+"/"+img, clr_img)

for folder in os.listdir(test_path):
    sub_path=test_path+"/"+folder
    for img in os.listdir(sub_path):
      image_path=sub_path+"/"+img
      img_arr=cv2.imread(image_path)
      grey = cv2.cvtColor(img_arr, cv2.COLOR_BGR2GRAY)

      equ = cv2.equalizeHist(grey)                          # Histogram Equaization
      bilateral = cv2.bilateralFilter(equ, 15, 75, 75)      # Bilateral Filter
      clr_img = cv2.cvtColor(bilateral, cv2.COLOR_GRAY2BGR)

      cv2.imwrite(save_test_path+"/"+folder+"/"+img, clr_img)
```

Fig 5

This data has been trained VGG16 with 5 and 10epochs individually. The results are recorded. The accuracy in each case are mentioned in table 5

Epochs	**Accuracy**
5	92.42%
10	98.48%

Table 5

The accuracy has reduced to 92.42% which is unsatiosfactory compared to previous model. Since bilateral filter takes average pixel value of surrounding pixels and changes the pixel value. The pixels will get affected by the sorrounding pixel so herte there is chance that we can loose our features in the images. So the accuracy was dissappointing incase of Bilateral Filter.

Method 6:

Since Pnuemonia is a disease where fluids will form in the lungs. The chest X-rays will have white patches if a person has pneumonia. It just depends on that pixel value. It doesnot depend on pattern in which the patches form or it doesnot depend on surrounding pixel values, so i came across image segmentation algorithms. Since we need to classify each pixel into categories Semantic Segmentation is right for this approach. Particularly Kmeans Semantic Segmentation has been implemented on this data.

The images wil be given into K means model to form clusters. When a new image is given to model it classify each and every pixel into different categories. And if the image has many pixels belong to patch cluster then we can say that the person has pneumonia. The output of the model is shown in Fig 6.

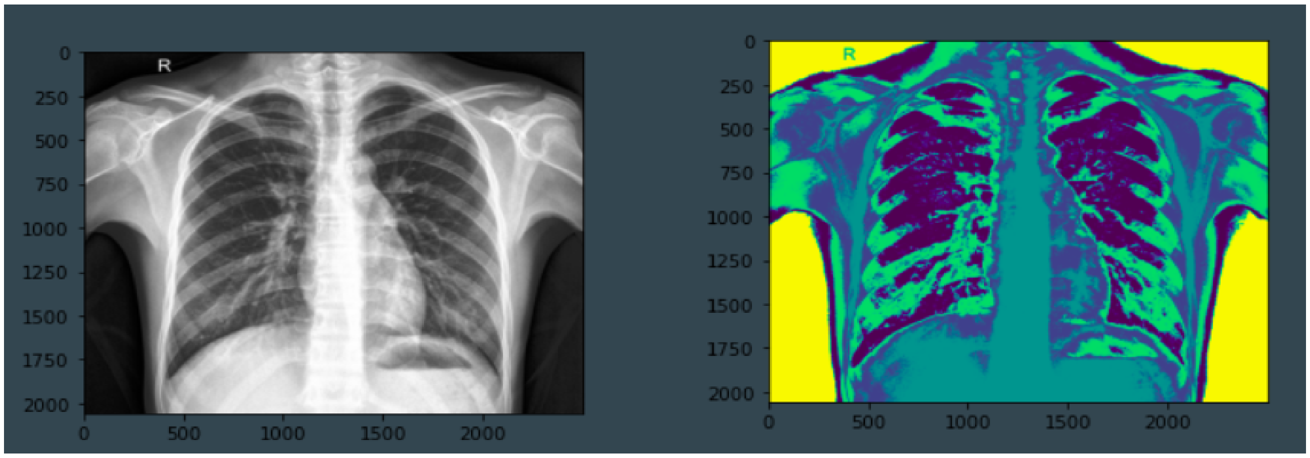


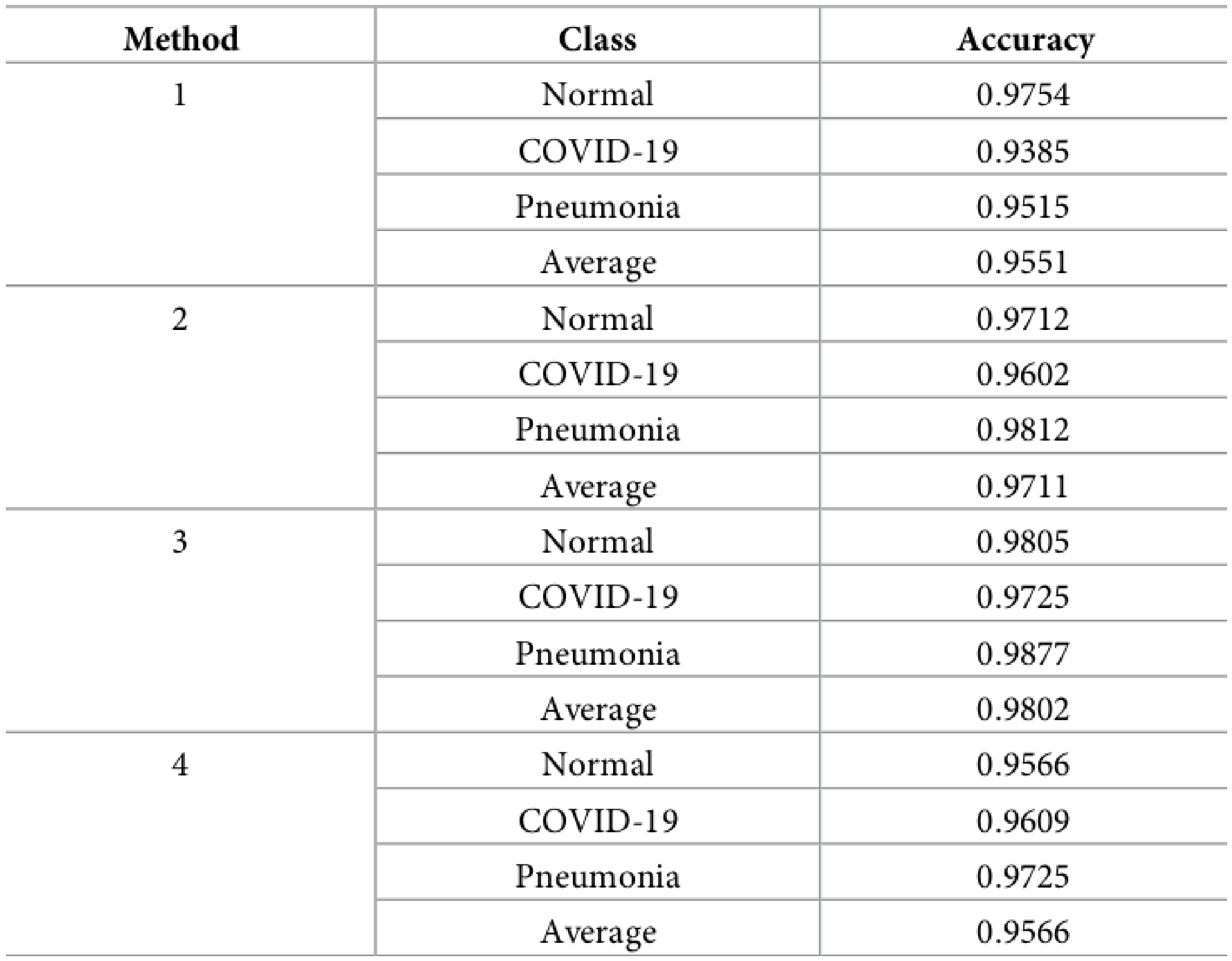
Fig 6

Comparison with other methods from literature

In a paper by A, Marciniak A, Tarczewska M, Lutowski Z (2022) [9] have implemented normal CNN in which they got accuracy of 95.51%. The architecture of the CNN that i implemented and they implemented is different.

Later they have implemented 3 preprocessing methods which are mentioned above methods. The accuracies of their models are given in Fig 7.

Methods 1: without preprocessing, 2: Hist Eq 3. Hist Eq + Gaussian Blur 4. Hist Eq + Bilateral filter



Method	Class	Accuracy
1	Normal	0.9754
	COVID-19	0.9385
	Pneumonia	0.9515
	Average	0.9551
2	Normal	0.9712
	COVID-19	0.9602
	Pneumonia	0.9812
	Average	0.9711
3	Normal	0.9805
	COVID-19	0.9725
	Pneumonia	0.9877
	Average	0.9802
4	Normal	0.9566
	COVID-19	0.9609
	Pneumonia	0.9725
	Average	0.9566

Fig 7

These results are compared with my results. The main difference is that they used their own CNN model but i have implemented VGG16 architecture. And both results are compared in table 6.

Preprocessing used	**My Model's Acccuracy**	**Model's accuracy mentioned in paper**
Without preprocessing	86.36	95.51
Histogram Equilization	92.42	97.11
HIst Eq + Gaussian Blur	98.48	98.02
HIst Eq + Bilateral Filter	92.42	95.66

Table 6

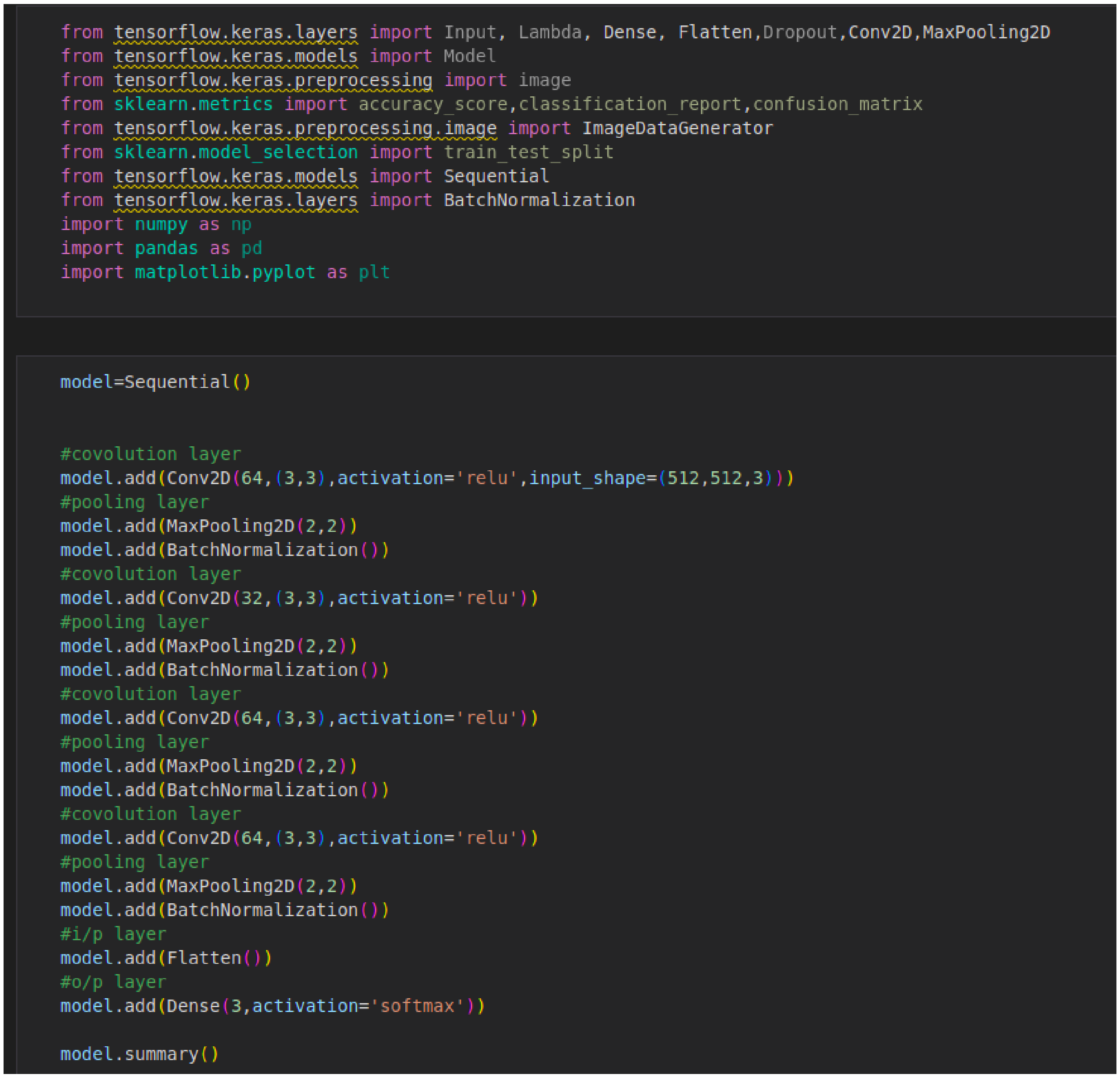
With Histogram Equization and Gaussian Blur my model has performed well compared to existing model. Though there was small difference in the accuracy but it is more than the proposed model. So this is the output and novelty of my work.

In a paper by Yong Yang, Shuying Huang they have used FuzzyC means clustering method for image segmentation. But in semantic segmentation i used Kmeans Custering semantic segmentation for clustering the images.

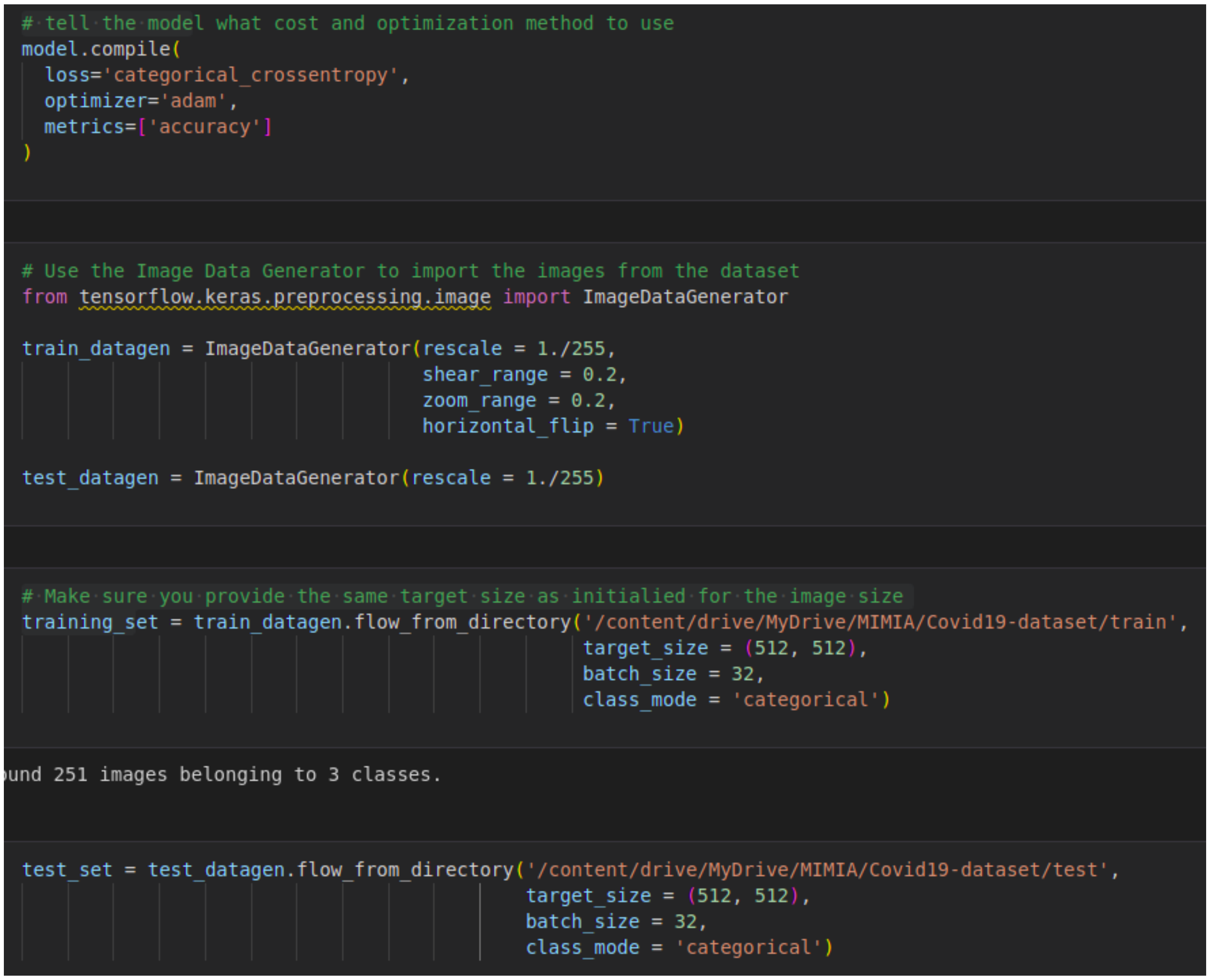
Implementation

For implementing all these models and networks i have used python language. Tensorflow and keras are the python modules. These are used for making Deep learning models and laoding and preprocessing data.

Opencv, numpy are the python modules which are used for the preprocessing of the images in the dataset. Matplotlib is another python module which is used for plotting graphs and vizualizing data. The nessesary screenshots are given in below images.


```python
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten,Dropout,Conv2D,MaxPooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```python
model=Sequential()


#covolution layer
model.add(Conv2D(64,(3,3),activation='relu',input_shape=(512,512,3)))
#pooling layer
model.add(MaxPooling2D(2,2))
model.add(BatchNormalization())
#covolution layer
model.add(Conv2D(32,(3,3),activation='relu'))
#pooling layer
model.add(MaxPooling2D(2,2))
model.add(BatchNormalization())
#covolution layer
model.add(Conv2D(64,(3,3),activation='relu'))
#pooling layer
model.add(MaxPooling2D(2,2))
model.add(BatchNormalization())
#covolution layer
model.add(Conv2D(64,(3,3),activation='relu'))
#pooling layer
model.add(MaxPooling2D(2,2))
model.add(BatchNormalization())
#i/p layer
model.add(Flatten())
#o/p layer
model.add(Dense(3,activation='softmax'))

model.summary()
```

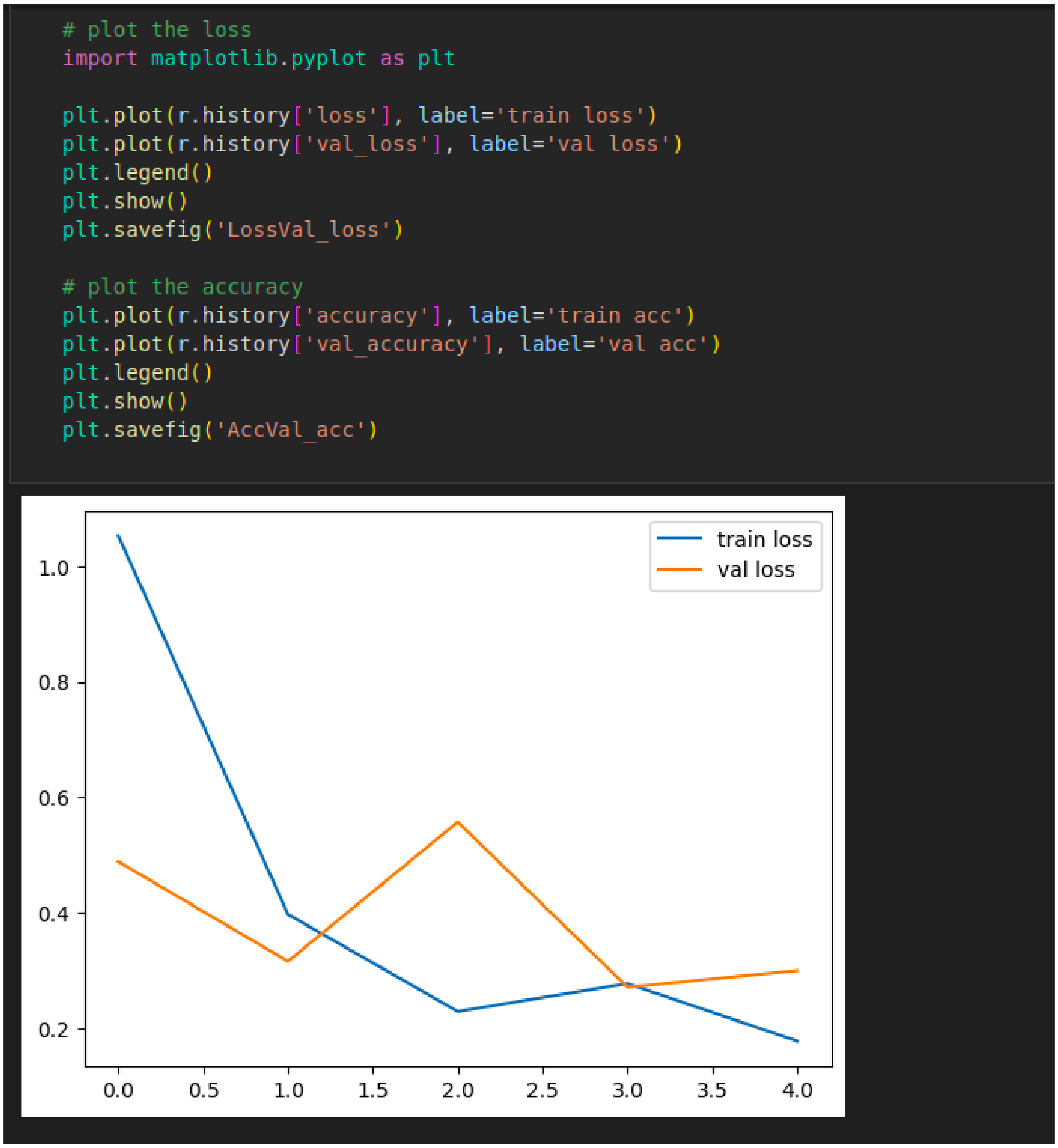

```
# tell the model what cost and optimization method to use
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

```
# Use the Image Data Generator to import the images from the dataset
from tensorflow.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)
```

```
# Make sure you provide the same target size as initialied for the image size
training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/MIMIA/Covid19-dataset/train',
                                                 target_size = (512, 512),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

```
und 251 images belonging to 3 classes.
```

```
test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/MIMIA/Covid19-dataset/test',
                                            target_size = (512, 512),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```



```
# plot the loss
import matplotlib.pyplot as plt

plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```

Results and Discussion

The results from the outputs of my models and their comparisons are given in table 8

MODEL	ACCURACY
Custom CNN Model	48.48%
VGG16	86.36%
VGG16 + Histogram Eq	95.45%
VGG16 + Histogram Eq + GaussianBlur	98.48%
VGG16 + Histogram Eq + BilateralFilter	92.42%

Table 8

If we see all the results the most accuracy has been achieved with VGG16 aloing with Hist Eq + Gaussian Blur which is 98.48%.

Since Histogram Eq increase the range of pixel values the pixels which are mixed and cannot differentiated with bones can be differentiated. And along side Gaussian Blur removes noise from the images which increases ti find proper features of the image. So these reasons helped the model to get more accuracy.

Conclusion:

In conclusion, the use of deep learning techniques for pneumonia detection has shown promising results. The ability of these models to analyze large amounts of medical data and accurately classify pneumonia has the potential to significantly improve patient outcomes and reduce healthcare costs. However, there are still limitations and challenges that need to be addressed, such as the need for large amounts of data for training and the potential for bias in the data.

Moreover, the development of more robust and accurate deep learning models for pneumonia detection is a constantly evolving field of research. With the continued advancement of technology and the growing availability of medical data, there is great potential for deep learning to continue making a significant impact in the field of healthcare.

Overall, the use of deep learning for pneumonia detection is a promising development in medical research, and continued efforts in this field have the potential to greatly benefit patients and healthcare systems around the world.

References

[1] "A Survey on Deep Learning in Medical Image Analysis" by Litjens et al. (2017)

[2] "Deep learning for automated pneumonia detection in chest X-rays: A survey" by Razzak et al. (2018)

[3] "Pneumonia Detection Using Deep Learning from X-Ray Images: A Comprehensive Review" by Kumar et al. (2021)

[4] "Deep Learning for Medical Image Analysis: A Review" by Shen et al. (2017)

[5] "Detection of Pneumonia in Chest X-Ray Images using Deep Learning: A Review" by Thakur and Suman (2021)

[6] "A Deep Learning Framework for Pneumonia Detection in Chest X-Ray Images" by Zheng et al. (2020)

[7] "Pneumonia Detection using Convolutional Neural Networks" by Yoo et al. (2019)

[8] "A Deep Learning Approach to Pneumonia Detection in Chest X-Ray Images" by Rajpurkar et al. (2018)

[9] Pre-processing methods in chest X-ray image classification by A, Marciniak A, Tarczewska M, Lutowski Z (2022)

[10] Image segmentation by fuzzy c-means clustering algorithm with a novel penalty term by Yong Yang, Shuying Huang