Detection of COVID-19 using patient’s CT scans

Ishan Bhargava

Rutvi Rahul Gadre

Under the guidance of : Prof Donghui Yan

Abstract

The global spread of COVID-19 has made it essential to identify the disease early on for effective management and treatment. CT scans are proving to be a promising method for detecting COVID-19 due to their high sensitivity and specificity. Recent studies investigating the use of CT scans for COVID-19 detection are reviewed in this project, including their advantages and disadvantages, such as precision, speed, and accessibility. The project also discusses the potential of applying artificial intelligence (AI) techniques like deep learning to automatically detect COVID-19 from CT scans. The conclusion emphasizes the potential of CT scans and AI in improving the accuracy and speed of COVID-19 detection and how they can have a positive impact on the global fight against the pandemic.

Introduction

The rapid and accurate diagnosis of COVID-19 is an urgent necessity due to the ongoing pandemic. While the current gold standard for COVID-19 diagnosis is real-time RT-PCR testing, this technique has limitations like false negatives and a lengthy turnaround time for results. CT scans have become a promising alternative for detecting COVID-19 because of their exceptional sensitivity and capacity to detect slight alterations in lung tissue.

The project introduces a technique for identifying COVID-19 in CT scans using the VGG-16 deep learning model and OpenCV. The proposed method involves pre-processing the CT scans with OpenCV, which includes segmenting the lung regions and standardizing pixel intensities. The pre-processed images are then entered into the VGG-16 model, which has been pre-trained on a vast collection of natural images and modified on a COVID-19 CT scan dataset to identify characteristics particular to COVID-19 detection.

The performance of the proposed approach is evaluated on a dataset of COVID-19 CT scans and non-COVID-19 CT scans provided to us by Prof. Donghui Yan. The outcomes show that the suggested method attains outstanding accuracy, sensitivity, and specificity in detecting COVID-19 from CT scans. The method can even pinpoint the location of COVID-19 lesions in the lung tissue, which might be beneficial for predicting the disease's progression and planning treatment.

In general, the proposed approach offers a potential solution for enhancing the precision and speed of COVID-19 detection using CT scans. This has the potential to make a significant contribution to the worldwide effort to manage the pandemic.

Python packages used:

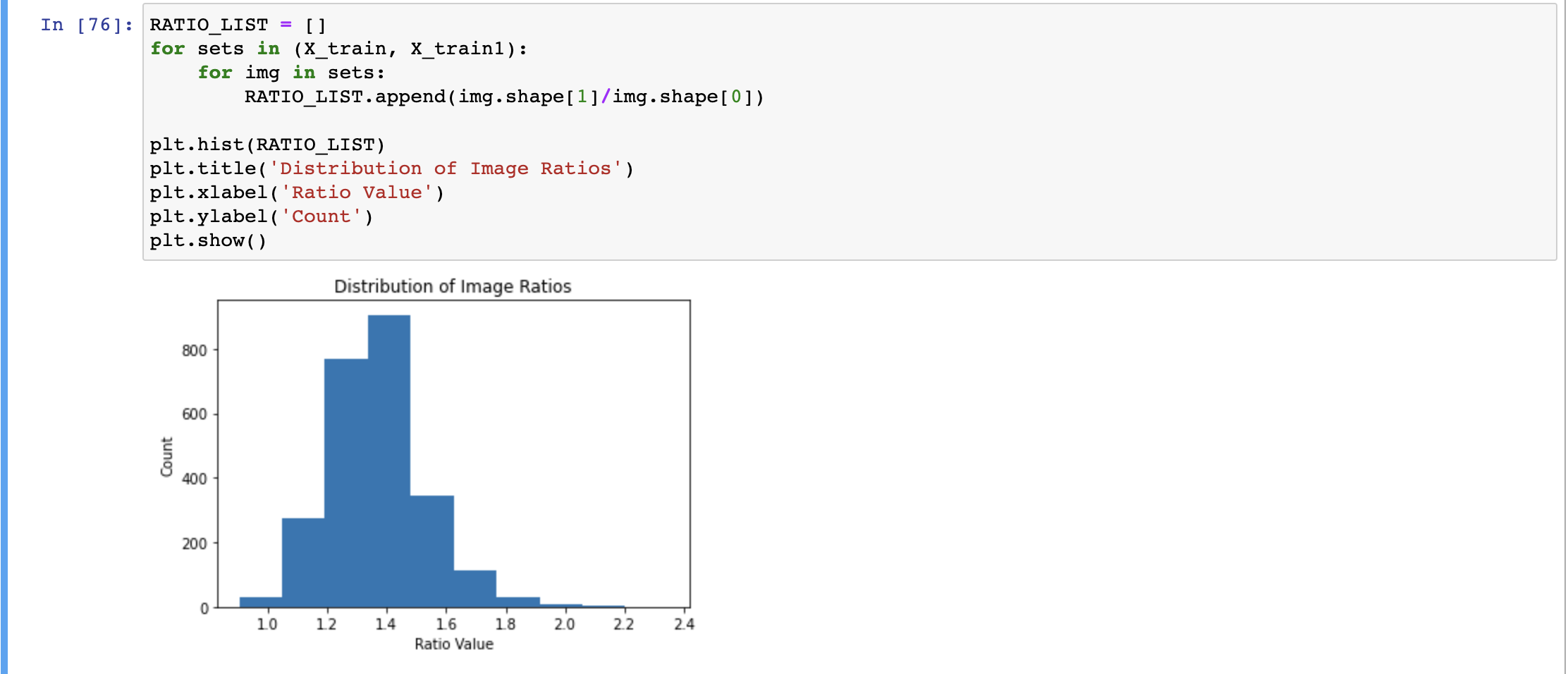
* Pandas
* Numpy
* OpenCV
* VGG-16 pre-trained model
* Math
* Matplotlib
* Seaborn
* Scipy

Open CV and VGG-16 Deep Learning model

OpenCV (Open-Source Computer Vision) is a free, open-source computer vision and machine learning library developed by Intel. It provides a wide range of tools and algorithms that enable developers to build computer vision applications, including object detection, image processing, facial recognition, and more. OpenCV supports multiple programming languages, including C++, Python, Java, and MATLAB, and is widely used in various fields such as robotics, healthcare, security, and entertainment.

VGG-16 is a deep convolutional neural network (CNN) model that was developed by the Visual Geometry Group at the University of Oxford. It consists of 16 layers of trainable parameters and has achieved high accuracy on various computer vision tasks such as image classification and object detection. The VGG-16 model is widely used as a baseline model in deep learning research and has been employed in various applications, including medical image analysis and natural language processing.

Data Pre-processing

* The following code snippet we are loading the and then calculating the aspect ratio (width to height ratio) of images in two sets of training data, X\_train and X\_train1.
* It is then creating a list, RATIO\_LIST, containing these aspect ratios for each image in the sets. The aspect ratio is calculated by dividing the width of the image (img.shape[1]) by its height (img.shape[0]).  
  
* We are then plotting a histogram of the distribution of aspect ratios using the Matplotlib library.

x-axis = aspect ratio value

y-axis = count of images with that aspect ratio.

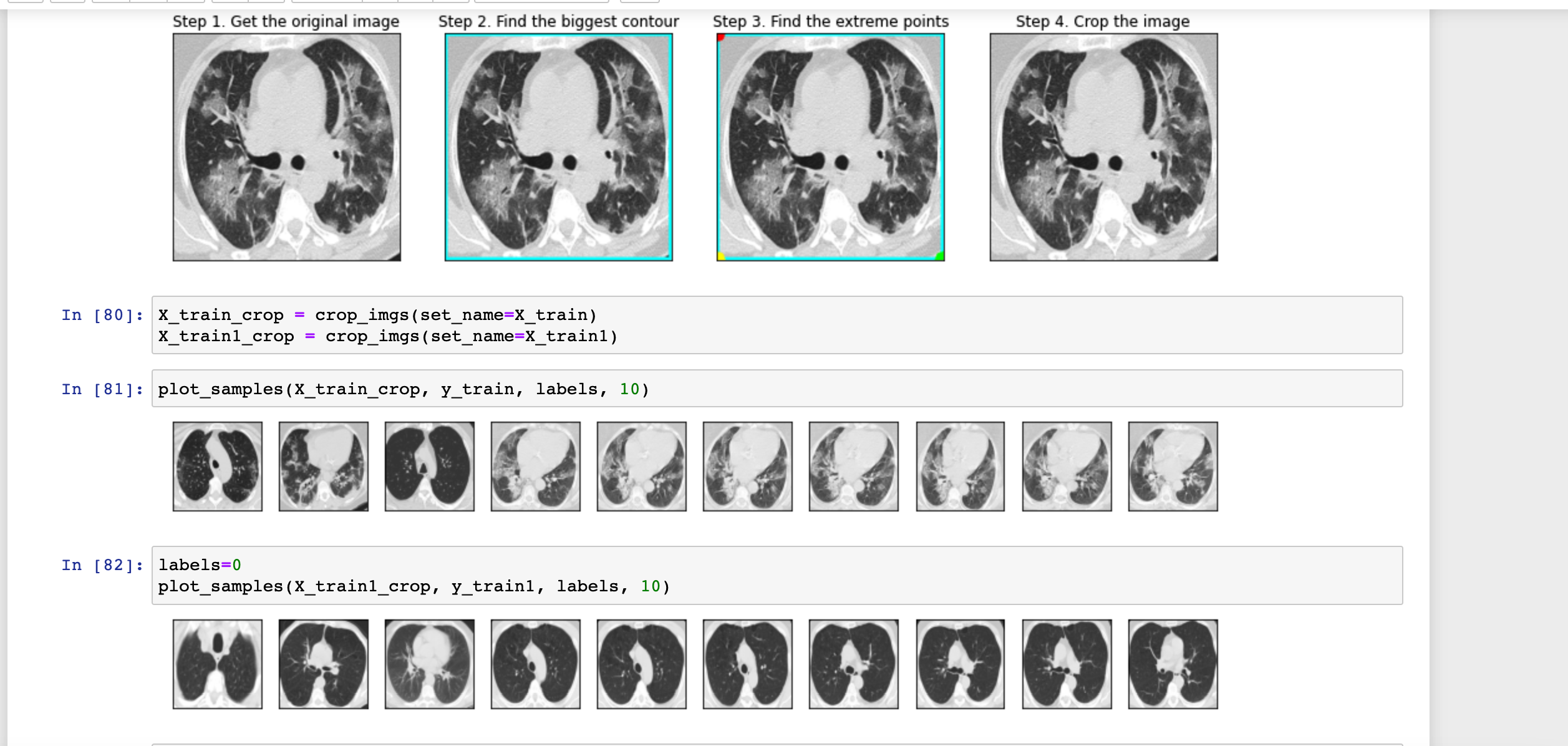
Title = 'Distribution of Image Ratios’

* The subsequent code introduces a function named crop\_imgs, which accepts a group of images named set\_name and an optional parameter called add\_pixels\_value, with a default value of 0.

Graphical user interface, text, application, email

Description automatically generated

* The function loops through each image in set\_name, resizes it to a specified IMG\_SIZE using OpenCV's resize function, and converts it to grayscale using cv2.cvtColor(). The grayscale image is then processed using a Gaussian blur to reduce noise.  
    
  Graphical user interface, text, application

  Description automatically generated
* A binary threshold is applied to the blurred image using cv2.threshold(), followed by erosion and dilation operations using cv2.erode() and cv2.dilate() to remove any small holes or gaps in the image.
* Contours are then identified in the processed image using cv2.findContours() and the largest contour is selected using max(). The four extreme points of the contour are then determined using numpy's argmin() and argmax() functions and assigned to the variables extLeft, extRight, extTop, and extBot.
* Lastly, utilizing the coordinates of the extreme points and the extra number of pixels specified by the add\_pixels\_value parameter, a fresh image is cropped from the original image. Subsequently, the cropped image is appended to a new list named set\_new.  
    
  

VGG16 Model and Training

* The following code introduces a Sequential model known as vgg16, created with the Keras library. The model is constructed on top of the pre-existing VGG16 network, which serves as the first layer and is assigned to the vgg variable.  
    
  Graphical user interface, text, application, email

  Description automatically generated
* In the final step, a dense output layer is added to the model with a sigmoid activation function, and the number of neurons is set equal to the number of classes to be predicted. The VGG16 layer is made non-trainable by setting its trainable attribute to False.
* The model is compiled using the binary cross-entropy loss function, RMSprop optimizer with a learning rate of 1e-4, and accuracy metric. However, the model is re-compiled using the Adam optimizer with a learning rate of 0.0003, beta\_1=0.9, beta\_2=0.999, epsilon=None, decay=0.0, and amsgrad=False, along with the accuracy metric.
* Model summary is shown in the below code snippet  
    
  Graphical user interface, text, application

  Description automatically generated
* The model is trained using the fit\_generator function, which uses batches of data generated by train\_generator and validation\_generator created with the ImageDataGenerator function. The steps\_per\_epoch and validation\_steps parameters define how many steps are taken during each epoch and validation epoch. These parameters are calculated by dividing the total number of images in the training or validation dataset by the batch size.  
    
  Text

  Description automatically generated
* The code trains the model for 10 epochs, and the training progress is stored in the vgg16\_history variable. The total time taken for training the model is measured using start and end time variables. After completing the training, the console displays the total training time by subtracting the end time from the start time.

Model Performance and Results

* The top subplot in the following figure displays the "Vgg16 Loss". The plot shows how the model's loss values vary during training and validation epochs. The loss values for the training and validation datasets are stored in the 'loss' and 'val\_loss' keys of the vgg16\_history.history dictionary, respectively. To plot these values, the plot() method of pyplot is employed, and the labels 'train' and 'Validation' are provided using the label parameter. Finally, the legend() method is called to create a legend in the plot.  
    
  Chart, line chart

  Description automatically generated
* The second subplot (bottom) displays the accuracy values of the Vgg16 model during training and validation epochs. Its title is set to "Vgg16 Accuracy". The accuracy values for both the training and validation sets are retrieved from the 'acc' and 'val\_acc' keys of the vgg16\_history.history dictionary. Using the plot() method of pyplot, the values are plotted with 'train' and 'Validation' labels for distinction. Lastly, a legend is displayed in the plot by calling the legend() method.
* The performance metrics are shown in the following screenshot  
    
  Graphical user interface, application

  Description automatically generated

Conclusion

The global spread of COVID-19 has made it essential to identify the disease early on for effective management and treatment. CT scans are proving to be a promising method for detecting COVID-19 due to their high sensitivity and specificity. Recent studies investigating the use of CT scans for COVID-19 detection are reviewed in this article, including their advantages and disadvantages, such as precision, speed, and accessibility. The article also discusses the potential of applying artificial intelligence (AI) techniques like deep learning to automatically detect COVID-19 from CT scans. The conclusion emphasizes the potential of CT scans and AI in improving the accuracy and speed of COVID-19 detection and how they can have a positive impact on the global fight against the pandemic.