A

Main Project Report On

AUTOMATIC PREDICTION OF CORONA VIRUS USING CHEST X-RAY AND CT-SCAN

Submitted to "JNTUK" for fulfillment of requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

D. JHANSI LAKSHMI	**17NU1A0512**
B. LAKSHMI PRASANNA	**17NU1A0506**
B. NARASIMHA RAJU	**17NU1A0507**
V. S. TEJASWI	**17NU1A0552**

UNDER THE GUIDANCE OF

Mr A. SURAJ KUMAR, MTech
Assistant professor

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NADIMPALLI SATYANARAYANA RAJU INSTITUTE OF TECHNOLOGY

(AUTONOMOUS)

(Affiliated to JNTUK Kakinada, Approved by AICTE, New Delhi)

SONTYAM, VISAKHAPATNAM-531173 (2017-2021)

NADIMPALLI SATYANARAYANA RAJU INSTITUTE OF TECHNOLOGY

(AUTONOMOUS)

(Affiliated to JNTUK Kakinada, Approved by AICTE, New Delhi)

SONTYAM, VISAKHAPATNAM-531173

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled " **AUTOMATIC PREDICTION OF CORONA VIRUS USING CHEST X-RAY AND CT-SCAN**" that is being submitted by **D. JHANSI LAKSHMI (17NU1A0512), B. LAKSHMI PRASANNA (17NU1A0506), B. NARASIMHA RAJU (17NU1A0507), V.S TEJASWI (17NU1A0552)** for fulfilment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ENGINEERING** to Jawaharlal Nehru Technological University-Kakinada is a record of bonafide work carried out by them under my guidance and supervision.

Internal Guide	Head of the Department
Mr. A. Suraj Kumar	**Dr. B. Ravi Kiran**
Assistant Professor	Professor
Department of CSE	Department of CSE
Held on:	**Batch no:**

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

I'm wish to express our deep sense of gratitude in its humbleness to our Project guide **Mr**. **A. Suraj Kumar**, Assistant Professor Department of Computer Science & Engineering, for her immune interest and whole hearted involvement in the project. She had given us splendid and gracious guidance, provided an excellent environment and also motivated us throughout our project. I'm truly grateful for his valuable suggestions and advice.

I'm grateful to Head of the Department **Dr. B. Ravi Kiran**, Professor for providing necessary facilities to carry out this project successfully.

I would like to express our profound appreciation to our beloved Principal **Dr. M.A. Khadar Baba** for extending his official support for the progress of the project.

Last but not least I would like to thank my parents and friends who helped me a lot in finalizing this project with in the limited time frame.

D. JHANSI LAKSHMI

17NU1A0512

ACKNOWLEDGEMENT

I'm wish to express our deep sense of gratitude in its humbleness to our Project guide **Mr**. **A. Suraj Kumar**, Assistant Professor Department of Computer Science & Engineering, for her immune interest and whole hearted involvement in the project. She had given us splendid and gracious guidance, provided an excellent environment and also motivated us throughout our project. I'm truly grateful for his valuable suggestions and advice.

I'm grateful to Head of the Department **Dr. B. Ravi Kiran**, Professor for providing necessary facilities to carry out this project successfully.

I would like to express our profound appreciation to our beloved Principal **Dr. M.A. Khadar Baba** for extending his official support for the progress of the project.

Last but not least I would like to thank my parents and friends who helped me a lot in finalizing this project with in the limited time frame.

B. LAKSHMI PRASANNA

17NU1A0506

ACKNOWLEDGEMENT

I'm wish to express our deep sense of gratitude in its humbleness to our Project guide **Mr. A. Suraj Kumar**, Assistant Professor Department of Computer Science & Engineering, for her immune interest and whole hearted involvement in the project. She had given us splendid and gracious guidance, provided an excellent environment and also motivated us throughout our project. I'm truly grateful for his valuable suggestions and advice.

I'm grateful to Head of the Department **Dr. B. Ravi Kiran**, Professor for providing necessary facilities to carry out this project successfully.

I would like to express our profound appreciation to our beloved Principal **Dr. M.A. Khadar Baba** for extending his official support for the progress of the project.

Last but not least I would like to thank my parents and friends who helped me a lot in finalizing this project with in the limited time frame.

B. NARASIMHA RAJU

17NU1A0507

ACKNOWLEDGEMENT

I'm wish to express our deep sense of gratitude in its humbleness to our Project guide **Mr. A. Suraj Kumar**, Assistant Professor Department of Computer Science & Engineering, for her immune interest and whole hearted involvement in the project. She had given us splendid and gracious guidance, provided an excellent environment and also motivated us throughout our project. I'm truly grateful for his valuable suggestions and advice.

I'm grateful to Head of the Department **Mr. B. Ravi Kiran**, Professor for providing necessary facilities to carry out this project successfully.

I would like to express our profound appreciation to our beloved Principal **Dr. M.A. Khadar Baba** for extending his official support for the progress of the project.

Last but not least I would like to thank my parents and friends who helped me a lot in finalizing this project with in the limited time frame.

V. S. TEJASWI

17NU1A0552

DECLARATION

We hereby declare that this report for the project entitled "**AUTOMATIC PREDICTION OF CORONA VIRUS USING CHEST X-RAY AND CT-SCAN**" has been developed by us under the supervision of **Mr. A. Suraj Kumar, Department of Computer Science & Engineering, Nadimpalli Satyanarayana Raju Institute of Technology**, Visakhapatnam in the partial fulfilment of requirement. It is our own and not submitted to any other College/University or published any time before.

TEAM MEMBERS

D. JHANSI LAKSHMI
(17NU1A0512)

B. LAKSHMI PRASANNA
(17NU1A0506)

B. NARASIMHA RAJU
(17NU1A0507)

V. S. TEJASWI
(17NU1A0552)

INDEX

AUTOMATIC PREDICTION OF CORONAVIRUS USING CHEST X-RAY AND CT-SCAN

ABSTRACT

Currently, the detection of Corona virus disease 2019 (COVID-19) is one of the main challenges in the world, given the rapid spread of the disease. Recent statistics indicate that the number of people diagnosed with COVID-19 is increasing exponentially, with more than 1.6 million confirmed cases; the disease is spreading too many countries across the world. We analyze the incidence of COVID-19 distribution across the world. We present an artificial-intelligence technique based on a deep convolutional neural network (CNN) to detect COVID-19

The chest CT-based COVID-19 classification of the suspected patients requires radiologists and considerable amounts of their times as the number of COVID-19 suspects increases at a rapid rate. Moreover, it has been found that the COVID-19-infected patients show some patterns on chest CT images that are not easily detectable by the human eye Therefore, an automatic detection tool is much needed to assist in screening.

To develop an artificial-intelligence tool based on a deep convolutional neural network (CNN) which can examine chest X-ray images to identify such Covid patients which be available quickly and at low cost.

1. INTRODUCTION

Coronavirus Disease 2019 (COVID-19) is a highly contagious disease that spreads from one person to another with the appearance of respiratory distress [1]. It has been spread all over the world since December 2019 and so far, it has infected more than millions of people. The clinical tests like reverse transcription-polymerase chain reaction (RT-PCR) are usually used for classifying the suspected patients, but medical imaging techniques such as computed tomography (CT) has also been used to detect and evaluate COVID-19. The chest CT-based COVID-19 classification of the suspected patients requires radiologists and considerable amounts of their time as the number of COVID-19 suspects increases at a rapid rate. Moreover, it has been found that the COVID-19-infected patients show some patterns on chest CT images that are not easily detectable by the human eye. Therefore, an automatic detection tool is much needed to assist in screening COVID-19 pneumonia using chest CT imaging [3,4]. Like many other methodological innovations, artificial intelligence (AI) has been applied to the timely, rapid, and effective diagnosis of COVID-19 using chest CT images. AI-Driven tools may provide automated and fast approaches for the detection and classification of COVID-19 on chest CT.

In this study, we developed a deep learning-based Convolutional Neural Network (CNN) model to classify COVID-19 cases from healthy cases. The preprocessing steps consisted of transforming image intensities into the Hounsfield unit, extracting the lung part by image processing techniques, equalizing histograms with a transformation created by the intensities of the infection regions, and stretching the contrast of the images. Then, after training and evaluating several deep learning networks, such as ResNet, Inception, VGG16, DenseNet, and Xception by theimages of 80% of the cases in each group, VGG16 was chosen as our base model, since it is less complicated compared to the other models. The two-class output was obtained using pooling,

dropout, and dense layers. Finally, our model was tested by the CT images of 20% of the cases in each group. The results showed 89.78% precision, 91.50% sensitivity, 88.66% specificity, 0.9063 F1-Score, and 90.14% accuracy.

1.2 DEEP LEARNING:

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

- Deep learning is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions.
- Deep learning AI is able to learn without human supervision, drawing from data that is both unstructured and unlabeled.
- Deep learning, a form of machine learning, can be used to help detect fraud or money laundering, among other functions.

How Deep Learning Works

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated

support Deep learning unravels huge amounts of unstructured data that would normally take humans decades to understand and process.

A Deep Learning Example

Using the fraud detection system mentioned above with machine learning, one can create a deep learning example. If the machine learning system created a model with parameters built around the number of dollars a user sends or receives, the deep-learning method can start building on the results offered by machine learning .Each layer of its neural network builds on its previous layer with added data like a retailer, sender, user, social media event, credit score, IP address, and a host of other features that may take years to connect together if processed by a human being. Deep learning algorithms are trained to not just create patterns from all transactions, but also know when a pattern is signaling the need for a fraudulent investigation.

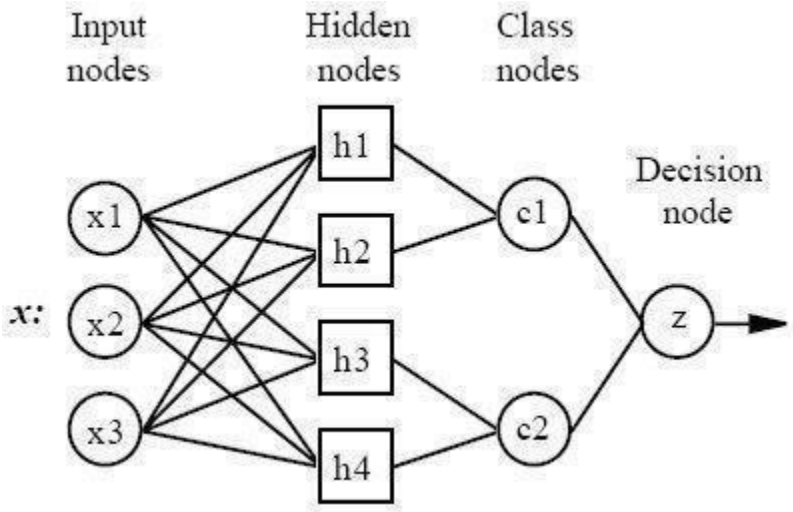
The final layer relays a signal to an analyst who may freeze the user's account until all pending investigations are finalized .Deep learning is used across all industries for a number of different tasks.

1.3 NEURAL NETWORKS:

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity. Tasks in speech recognition or image recognition can take minutes versus hours when compared to the manual identification by human experts. One of the most well-known neural networks is Google's search algorithm.



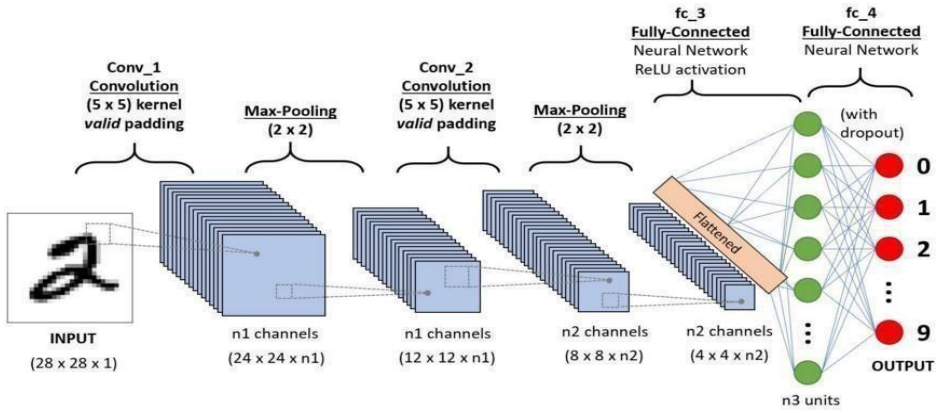
1.3.1 RECURRENT NEURAL NETWORK :

Performance of the RNN classifier was evaluated in terms of training performance and classification accuracies. This network is a kind of radial basis network and It gives fast and accurate classification and is a promising tool for classification of the defects from quality material. Existing weights will never be alternated but only new vectors are inserted into weight matrices when training. So it can be used in real-time. Since the training and running procedure can be implemented by matrix manipulation, the speed of RNN is very fast.

1.3.2 CONVOLUTIONAL NEURAL NETWORK:

Convolutional Neural Networks a special type of neural network that roughly imitates human vision. Over the years CNNs have become a very important part of many Computer Vision applications. So let's take a look at the workings of CNNs.

Background of CNNs:



What exactly is a CNN?

In deep learning, a **convolutional neural network** (**CNN/ConvNet**) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics **convolution** is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other

2. LITERATURE SURVEY

We searched ArXiv, MedRxiv, and Google Scholar for AI for COVID-19 diagnosis with chest CT. At the time of writing (August 24, 2020), there have been nearly 100 studies and only 17 of them were peer-reviewed papers. In total, 30 stud- ies (17 peer-reviewed and 13 non-peer-reviewed papers) were selected for this review. We noticed that very different classification terms are reported by the authors such as "nor- mal", "healthy", "other", "COVID-19", "non-COVID-19", "without COVID-19", "community-acquired pneumonia (CAP)", "other pneumonia", "bacterial pneumonia", "SARS", "lung cancer", "type A influenza (influ-A)", and "severity". Therefore, we categorized the studies into four main tasks as follows: COVID-19/normal, COVID- 19/non-COVID-19, COVID-19/non-COVID-19 pneumonia, and COVID-19 severity classification. COVID- 19 group consists of COVID-19 patients. The normal group includes only healthy subjects. Non-COVID-19 group includes either one of the cases which is not COVID-19 or a combination of all other cases. The non- COVID-19 pneumonia group includes other types of pneumonia, which is not caused by COVID-19, such as viral or bacterial pneumonia, as well as influenza A and SARS. Lastly, COVID-19 severity classification aims at classifying the COVID-19cases as severe or non-severe.

Since the rapid studies on the detection of COVID-19 in CT scans continue, the researchers who take into account the peer-review period in the journals share the results they obtained in their studies with other researchers and scientists as preprints in different publication environments. Machine learning is used to make decisions on tasks that people have difficulty making decisions orproblems that require more stable decisions using both numerical and image-based data. A deep convolutional neural network (CNN) is the most widely used among machine learning methods. Itis one of the first preferred neural networks, especially in image- based problems, since it contains both feature extraction and classification stages and produces very effective results. In image-based COVID-19 researches, the CNN model or different models produced from CNN are widely encountered. In the research, a generally hold-out method and a few k-fold cross-validation

were used during the training phase. In the hold-out method, while training is done by dividing the data into two parts as test and train, in k-fold cross-validation, the data is divided into k-folds, and the folds are trained k-times by shifting the testing fold in each training so that each fold is used in the test phase. It is used as a better method for model evaluation Computational and Mathematical Methods in Medicine

COVID-19/Normal Classification Studies. Alom et al. [9] implemented two deep learning models for COVID-19 detection and segmentation. Inception Recurrent Residual Neural Network (IRRCNN), which is based on transfer learning, was used for the COVID-19 detection task, and theNABLA- N model was for the segmentation task. They considered different datasets to detect COVID-19 on CT images, by using an additional chest X-ray dataset. The pub- licly available dataset was considered for the segmentation procedure of CT images, and the dataset that consists of 425 CT image samples, with 178 pneumonia, and 247 normal images were considered for the COVID-19 detection pur- pose. All images were resized to the dimensions of 192 × 192 pixels, and375 of total images were used for training and validation with a data augmentation procedure. The training was performed using Adam optimizer with a learning rate of $1\times 10{-}3$ and a batch size of

The COVID-19 detection and segmentation accuracy were achieved by 98.78% and 99.56%, respectively.

Hu et al. [10] constructed an AI model on ShuffleNet V2 [11], which provides fast and accurate training in transfer learning applications. The considered CT dataset consists of 521 COVID-19 infected images, 397 healthy images, 76 bacterial pneumonia images, and 48 SARS images. The data augmentation procedure as flip,rotation, translation, brightness adjustment, and flip+brightness adjustment was applied in this study to increase the number of training images. The first experiment was performed on the classification of COVID- 19 images from normal healthy images. The average sensitivity, specificity, and area under the curve (AUC) score were obtained as 90.52%, 91.58%, and 0.9689, respectively.

Gozes et al. [12] proposed a comprehensive system to detect COVID-19 from normal cases. The proposed system included lung segmentation, COVID-19 detection in CT slices, and marking cases as COVID-19 using a pretrained threshold based on the counted COVID-19 positive slices. Several datasets were considered in training and testing phases,

and pretrained network ResNet50 was used for the detection of COVID-19. The sensitivity, specificity, and the AUC score were achieved as 94%, 98%, and 0.9940, respectively.

In another study for differentiation of COVID-19 from normal cases, Kassani et al. [13] used several pretrained net-works such as MobileNet [14], DenseNet [15], Xception [16],InceptionV3 [17], InceptionResNetV2 [18], and ResNet [19] to extract the features of images within the publicly available dataset. Then, extracted features were trained using six machine learning algorithms, namely, decision tree, random forest, XGBoost, AdaBoost, Bagging, and LightGBM. Kassani et al.[13] concluded that the Bagging classifier obtained the optimal results with a maximum of 99:00%± 0:09 accuracy on features extracted by pretrained network DesnseNet121.

Jaiswal et al. [20] implemented a pretrained network DenseNet201-based deep model on classifying 2,492 CT-scans (1,262 positive for COVID-19, and the rest 1,230 are negative) as positive or negative. They compared their results with VGG16, ResNet152V2, and Inception-ResNetV2. They concluded that their model outperformed other considered models and achieved an overall accuracy of 96.25%. Table 1 summarizes the studies on COVID-19 vs. normal cases.COVID-19/Non-COVID-19 Classification Studies. Jin et al. [30] considered 496 COVID-19 positive and 260 nega- tive images collected in Wuhan Union Hospital, Western Campus of WuhanUnion Hospital, and Jianghan Mobile Cabin Hospital

In wuhan. Besides, they used two publicly available international databases, LIDC-IDRI [28] and ILD- HUG [31] (1012 and 113 subjects, respectively) as negative cases to develop the system. A 2D convolutional neural net- work was used for the segmentation of CT slices, and then, a model was trained for positive and negative cases. Jin et al. reported that the proposed system achieved by AUC score of 0.9791, sensitivity of 94.06%, and specificity of 95.47% for the external text cohort.

Singh et al. [32] proposed a multi objective differential evolution-(MODE-) based convolutional neural networks to detect COVID-19 in chest CT images. It was concluded that the proposed method outperformed the CNN, ANFIS, and ANN models in all considered metrics between1.6827% and 2.0928%

Amyar et al. [33] developed another model architec- ture that included image

segmentation, reconstruction, and classification tasks, which was based on the encoder and convolutional layer. The experiments were performed on three datasets that included 1044 CT images, and the obtained results showed that the proposed architecture achieved the highest results in their experiment, with 0.93% of the AUC score. Ahuja et al. [34] used data augmentation and pretrained networks to classify COVID-19 images. Data augmentation was performed using stationary wavelets, and the random rotation, translation, and shear operations were applied to the CT scan images. ResNet18, ResNet50, ResNet101, and SqueezeNet were implemented for the classification task, and Ahuja et al. concluded that ResNet18 outperformed other models by obtaining a 0.9965 AUC score.

Liu et al. [35] proposed another deep neural network model, namely, lesion-attention deep neural networks, where the backbone of the model used the weights of pretrained networks such as VGG16, ResNet18, and ResNet50. The pro- posed model was capable of classifying COVID-19 images, which was the main aim of the study, with 0.94 of the AUC score using VGG16 as the backbone model. Besides this, the model was able to make a multi label prediction on the five lesions.Instead of deep learning approaches,

Barstugan et al. [36] considered machine learning algorithms to classify 150 COVID-19 and non- COVID-19 images. Several feature extraction methods such as grey-level size zone matrix(GLSZM) and discrete wavelet transform (DWT) were con- sidered in the feature extraction process, and the extracted features were classified using a support vector machine. K-fold cross-validations were performed in the experiments with 2, 5, and 10 folds. Barstugan et al. concluded that 99.68% of accuracy was achieved by SVM using the GLSZM feature extraction method.

Wang et al. [37] conducted another study on differentiat- ing COVID-19 from non-COVID-19 CT scans. In their pro- posed network, UNet was first trained for lung region segmentation, and then, they used a pretrained UNet to test CT volumes to obtain all lung masks. They concatenated CT volumes with corresponding lung masks and sent them to the proposed DeCoVNet for the training. Wang et al. con- cluded that the proposed network achieved a 0.959 ROC AUC score.

Chen et al. [38] performed a study on collected 46,096 images from 106 patients (Renmin Hospital of Wuhan University–Wuhan, Hubei province, China). The proposed

system was based on segmenting CT scans using UNet++ and predicting the COVID-19 lesions. The prediction was performed by dividing an image into four segments and counting the consecutive images. If three consecutive images were classified as containing lesions, the case was classified as positive for COVID-19. The proposed system was evaluated using five different metrics, and it achieved92.59% and 98.85% of accuracy in prospective and retro- spective testing, respectively.

Jin et al. [39] considered the segmentation and pretrained models to classify COVID-19, healthy images, and inflammatory and neoplastic pulmonary diseases. Initially, pre-processing was applied to CT scan images to standardize images that were collected from five hospitals in China. Several seg- mentation models such as V-Net and 3D U-Net++ were considered, and segmented images were trained using pretrained network ResNet50 [19], Inception networks [17], DPN-92 [40], and Attention ResNet-50 [41]. Jin et al. concluded that the ResNet50 achieved the highest classification rates by 0.9910 of AUC score, 97.40% of sensitivity, and 92.22% of specificity with the images segmented by 3D U-Net++ segmentation model.

Pathak et al. [42] proposed a system for the detection of COVID-19 in CT scans that considereda pre- proposed trans- fer learning. The system used the ResNet50 to extract the features from CT images, and a 2D convolutional neural network was considered for the classification. The proposed system was tested on 413 COVID-19 and 439 non-COVID-19 images with 10-fold cross-validation, and it achieved 93.01% of accuracy.

Polsinelli et al. [43] proposed a light architecture by modifying the CNN. The proposed model was tested on two different datasets, and several experiments with different combinations were performed. The proposed CNN achieved 83.00% of accuracy and 0.8333 of F1 score.

Han et al. [44] proposed a patient-level attention-based deep 3D multiple instance learning (AD3D-MIL) that learns Bernoulli distributions of the labels obtained by a pooling approach. They used a total of 460 chest CT examples, 230 CT examples from 79 COVID-19 confirmed patients, 100 CT examples from 100 patients with pneumonia, and 130 CT examples from 130 people without pneumonia. Their proposed model achieved an accuracy, AUC, and the Cohen kappa score of 97.9%, 99.0%, and 95.7%, respectively, in the classification of COVID-19 and non-COVID-19.

Harmon et al. [45] considered 2724 CT scans from 2617 patients in their study. Lung regions were segmented by using 3d anisotropic hybrid network architecture (AH-Net), and the classification of segmented 3D lung regions was per- formed by using pretrained model DenseNet121. The proposed algorithm achieved an accuracy, specificity, and AUC score of 0.908, 0.930, and 0.949, respectively.

Wang et al. [53] proposed another deep learning method to distinguish COVID-19 and other pneumonia types. The segmentation, suppression of irrelevant area, and COVID-19 analysis were the processes of the proposed method. DenseNet121-FPN [15] was implemented for lung segmentation, and COVID19Net that had a DenseNet-like structure was proposed for classification purposes. Two validation sets were considered, and the authors reported 0.87 and 0.88 ROC AUC scores for these validation sets.

In addition to classify COVID-19 and normal cases, Hu et al. [10] performed another experiment to differentiate COVID-19 cases from other cases as bacterial pneumonia and SARS. The average sensitivity, specificity, and the AUC score were obtained as 0.8571, 84.88%, and 92.22%, respectively.

Bai et al. [54] implemented the deep learning architecture EfficientNet B4 [55] to classify COVID-19 and pneumonia slices of CT scans. The diagnosis of the six radiologists on the corresponding patients were used to evaluate the efficiency of the results obtained by an AI model. The AI model achieved 96% of accuracy, while the average accuracy of the diagnosis of radiologists was obtained at 85%.

Kang et al. [56] proposed a pipeline and multi view representation learning technique for COVID-19 classification using different types of features extracted from CT images. They used 2522 CT images (1495 are from COVID-19 patients, and 1027 are from community- acquired pneumonia) for the classification purpose. The comparison was performed using the benchmark machine learning models, namely, support vector machine, logistic regression, Gaussian-naive-Bayes classifier, K-nearest- neighbors, and neural networks. The proposed method outperformed the considered ML models with 95.5%, 96.6%, and 93.2% in terms of accuracy, sensitivity, and specificity.

Another study was performed by Shi et al. [57] to classify COVID-19 and pneumonia. They considered 1658 and 1027 confirmed COVID-19 and CAP cases. Shi et al. proposed a model that is based on random forest and automatically extracted a series

of features as volume, infected lesion number, histogram distribution, and surface area from CT images. The proposed method and considered machine learning models (logistic regression, support vector machine, and neural network) were then trained by the selected fea- tures with 5-fold cross-validation. The authors reported that the proposed method outperformed other models and produced the optimal AUC score(0.942).

Ying et al. [58] designed a network named as DRE-Net, which is based on the modifications on pretrained ResNet-50. The CT scans of 88 COVID-19 confirmed patients, 101 patients infected with bacteria pneumonia, and 86 healthy persons. The designed network was compared by the pre- trained models, ResNet, DenseNet, and VGG16. The presented results showed that the designed network outper- formed other models by achieving 0.92 and 0.95 of AUC scores for the image and human levels.

In addition to COVID-19/non-COVID-19 classification, Han et al. [44] performed experiments to classify COVID-19, common pneumonia, and no pneumonia cases as three classes classification. Their proposed AD3D-MIL model achieved an accuracy, AUC, and the Cohen kappa score of 94.3%, 98.8%, and 91.1%, respectively.

Ko et al. [59] proposed a model, a fast-track COVID-19 classification network (FCONet) that used VGG16, ResNet- 50, InceptionV3, and Xception as a backbone to classify images as COVID-19, other pneumonia, or non pneumonia. They considered 1194 COVID-19, 264 low-quality COVID- 19 (only for testing), and 2239 pneumonia, normal, and other disease CT-scans in their study. All images were converted into grayscale image format with dimensions of 256× 256. They used rotation and zoom data augmentation procedures to maximize the number of training samples. It was concluded that FCONet based on ResNet-50 outper- formed other pretrained models and achieved 96.97% of accuracy in the external validation data set of COVID-19 pneumonia images.

Li et al. [8] proposed a COVNet that used ResNet50 as a backbone to differentiate COVID-19, non pneumonia, and community-acquired pneumonia. In their study, 4352 chest CT scans from 3322 patients were considered. A max- pooling operation was applied to the features obtained from COVNet using the slices of the CT series, and the resultant feature map was fed to a fully connected layer. This led to generate a probability score for each considered class. It was concluded that the proposed model achieved a sensitivity, specificity, and ROC AUC scores of 90%, 96%, and 0.96, respectively, for the COVID-

19 class.

Ni et al. [60] considered a total of 19,291 CT scans from 14,435 individuals for their proposed model to detect COVID-19 in CT scans. Their proposed model included the combination of Multi-View Point Regression Networks (MVPNet), 3D UNet, and 3D UNet-based network for lesion detection, lesion segmentation, and lobe segmentation, respectively. Their algorithm analyzed the volume of abnor- malities and the distance between lesion and pleura to diag- nose theCOVID-19, and it was concluded that the proposed algorithm outperformed three radiologists in terms of accuracy and sensitivity by achieving 94% and 100%, respectively. Table 3 summarizes the classification results for COVID-19/non-COVID-19 pneumonia cases.

COVID-19 Severity Classification Studies. Xiao et al. [61] implemented a pretrained network ResNet34 to diagnose COVID-19 severity. The experiments were performed using five-fold cross-validation, and 23,812 CT images of 408 patients were considered. They concluded that the model achieved the ROC AUC score of 0.987, and the prediction quality of detecting severity and non severity of 87.50% and 78.46%.

Zhu et al. [62] proposed a model that was optimized by traditional CNN and VGG16 to stage the COVID-19 severity. A publicly available dataset was considered, and 113 COVID-19 confirmed cases were used to test their hypothe- sis. Obtained scores were compared by scores given by radi- ologists, and it was concluded that the top model achieved a correlation coefficient (R^2) and mean absolute error of 0.90 and 8.5%, respectively. Pu et al. [63] proposed an approach that initially seg- mented lung boundary and major vessels at two t points using UNet and registered these two images using a bidirectional elastic registration algorithm. Then, the average density of the middle of the lungs was used to compute a threshold to detect regions associated with pneumonia. Finally, the radiologist used to rate heat map accuracy in representing progression. In their study, two datasets that consisted of 192 CT scans

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

In Most of the Existing System a support vector machine (SVM) model for the classification of patients with COVID-19 and other types of pneumonia using CT chest images. This was done by extracting textural and histogram features of the infections and obtaining a radionics features vectorfrom each sample.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM:

- CXR-based detection of Covid-19 patients is that trained doctors may not be available all the time
- COVID-19 detection accuracy from X-ray images is very less
- Samples are very less to find the detection of disease

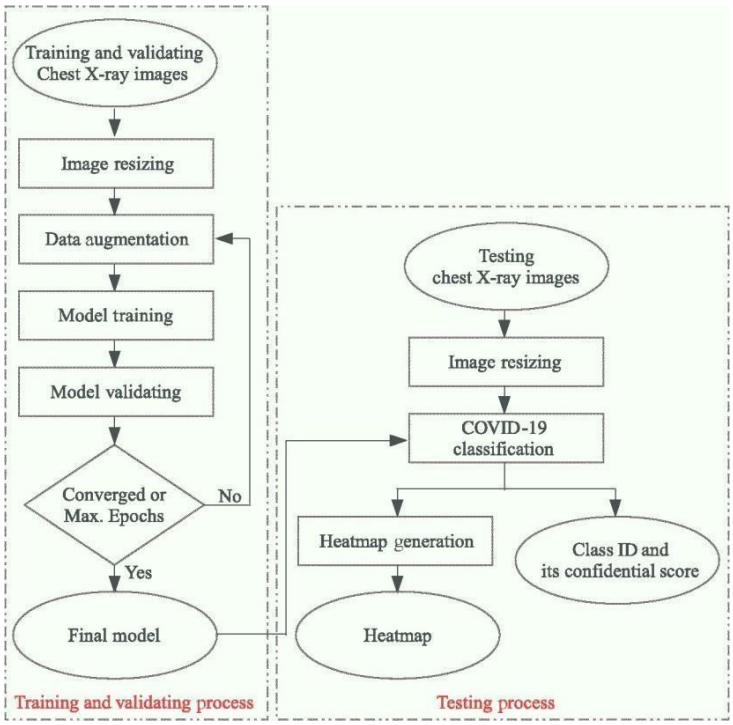
3.2 PROPOSED SYSTEM:

We present an artificial-intelligence technique based on a deep convolution neural network (CNN) to detect COVID19 patients using real-world datasets. Our system examines chest X-ray images to identify such patients. three forecasting methods—the prophet algorithm (PA), autoregressive integrated moving average (ARIMA) model, and long short-term memory neural network (LSTM)—were adopted to predict the numbers of COVID-19 confirmations.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM:

- Multiple architures are applied
- Accuracy is better
- We predict the numbers of COVID-19 confirmations, recoveries, and deaths over the next 7ds

3.2.2 PROPOSED SYSTEM ARCHITECTURE



3.3 ALGORITHM

To build a deep learning network for abnormalities detection using CNN we propose a structure with three convolutional layers as follows:

- step 1: 16 Convolutions layer with filter mask of size 3x3x1 and padding filter [1111].
- Step2: Maxpooling layer with filter mask size 2x2 max pooling with stride [2 2] and padding filter [0000].
- Step 3: Activate layer
- step 4: 32 Convolutions layer with filter mask of size 3x3x16 and padding filter [1111].
- Step5: Maxpooling layer with filter mask size 2x2 max pooling with stride [2 2] and padding filter [0000].
- Step 6: Activate layer

- step 7: 240 Convolutions layer with a filter mask of size 3x3x32 and padding filter[1111].

*The K-Means algorithm is summarized by the following steps:

- step 1:Place *Cj* points into the space represented by the objects that are being clustered.These points represent initial group centroids.
- step 2:Assign each object to the group that has the closest centroid.
- step 3:When all objects have been assigned, recalculate the positions of the *Cj* centroids.
- step 4:Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into

4. REQUIREMENTS

4.1 OBJECTIVE:

To develop an artificial-intelligence tool based on a deep convolutional neural network (CNN) which can examine chest X-ray images to identify such Covid patients which be available quickly and at low costs

4.2 FUNCTIONAL REUIREMENTS:

In Software engineering, a functional requirement defines a function of a software system or its components. A function is described as set of inputs, the behavior and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements

4.2.1 FUNCTIONAL REQUIREMENTS FOR THIS PROJECT

The functional requirements describe the interaction between the system an its environment independent of its implementation

MODULE 1: ResNet Architecture

Residual neural network (ResNet) is proposed by He et al. in 2015 [HZR16]. The hypothesis behind ResNet is that deeper networks are harder to optimize, since the deeper model should be able to perform as well as the shallower model by copying the learned parameters from the shallower model and setting additional layers to identity mapping. To help optimize deeper models,residual blocks are designed to fit a residual mapping $F(x)$ instead of the desired underlying mapping $H(x)$, and full ResNet architecture is built by stacking residual blocks. More specifically, every residual block has two 3×3 convolutional layers. Periodically, the number of filters are doubled and spatial down sampling is operated. Figure 3.1 illustrates the structure within the residual block and Figure

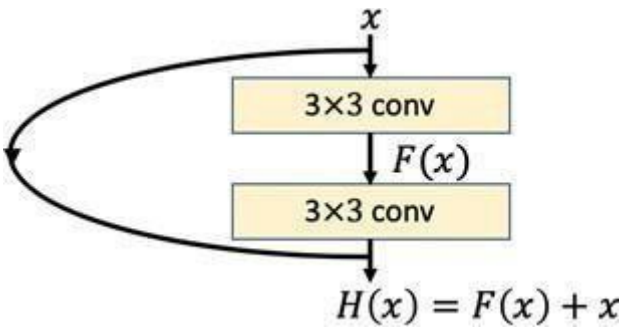


Figure 3.1: Structure of residual block

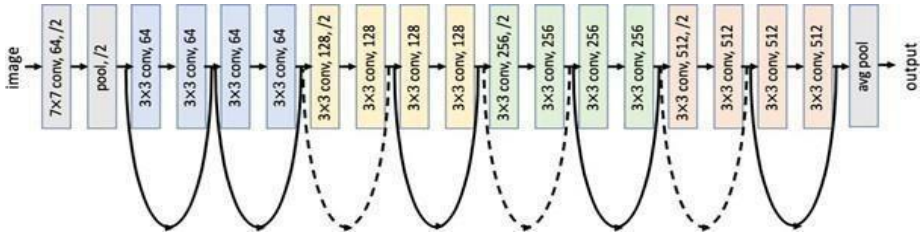


Figure 3.2: Schema of ResNet18 architecture

In this study, we build two deep models based on ResNet18 for the classification under two schemes. One is a binary classification that distinguishes COVID-19 pneumonia and non-COVID-19 cases, while the other is a four-class classification that classifies COVID-19 pneumonia, viral pneumonia, bacterial pneumonia, and normal cases.

MODULE 2: Inception V3

The “Inception” micro-architecture was first introduced by Szegedy et al. in their 2014 paper,

Going Deeper with Convolutions:

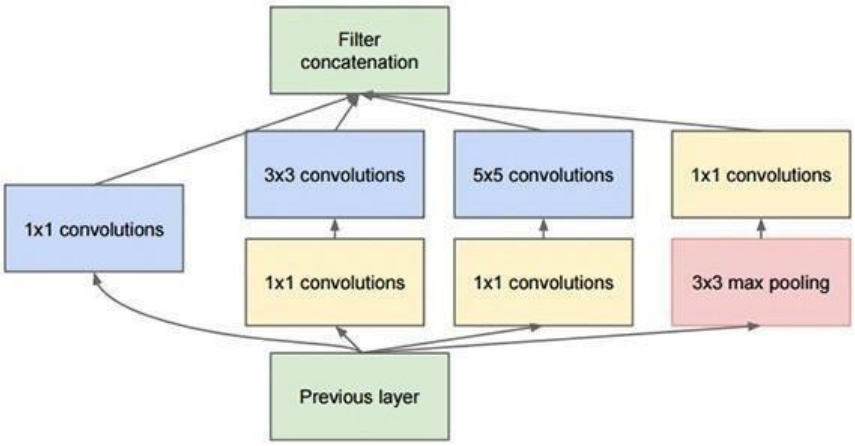


Figure : The original Inception module used in GoogLeNet.

The goal of the inception module is to act as a "multi-level feature extractor" by computing *1×1*, *3×3*, and *5×5* convolutions within the *same* module of the network — the output of these filters are then stacked along the channel dimension and before being fed into the next layer in the network.

The original incarnation of this architecture was called *GoogLeNet*, but subsequent manifestations have simply been called *Inception vN* where *N* refers to the version number put out by Google.

The Inception V3 architecture included in the Keras core comes from the later publication by Szegedy et al., ***Rethinking the Inception Architecture for Computer Vision*** (2015) which proposes updates to the inception module to further boost ImageNet classification accuracy.

The weights for Inception V3 are smaller than both VGG and ResNet, coming in at 96MB.

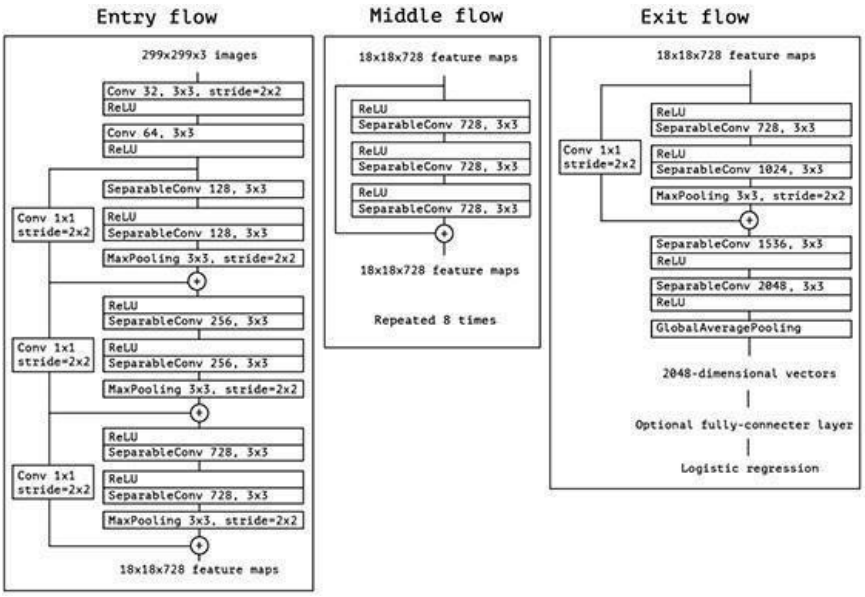
MODULE 3 : Xception

Xception was proposed by none other than **François Chollet** himself, the creator and chief maintainer of the Keras library.

Xception is an extension of the Inception architecture which replaces the standard Inception modules with depthwise separable convolutions.

We present an interpretation of Inception modules in convolutional neural networks as being an intermediate step in-between regular convolution and the depthwise separable convolution

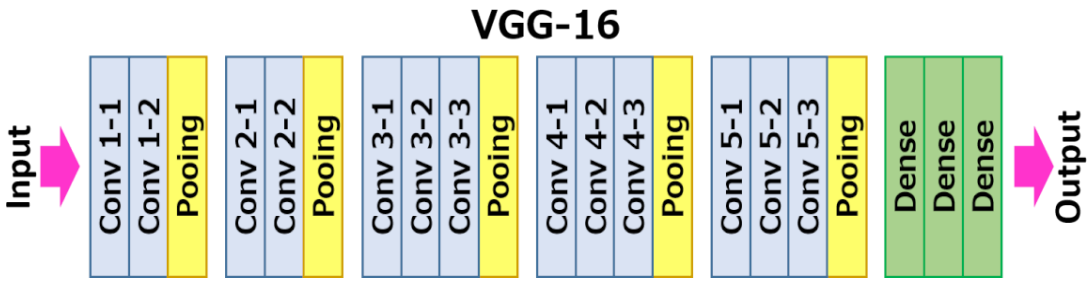
operation (a depthwise convolution followed by a pointwise convolution). In this light, a depthwise separable convolution can be understood as an Inception module with a maximally large number of towers. This observation leads us to propose a novel deep convolutional neural network architecture inspired by Inception, where Inception modules have been replaced with depthwise separable convolutions. We show that this architecture, dubbed Xception, slightly outperforms Inception V3 on the ImageNet dataset (which Inception V3 was designed for), and significantly outperforms Inception V3 on a larger image classification dataset comprising 350 million images and 17,000 classes. Since the Xception architecture has the same number of parameters as Inception V3, the performance gains are not due to increased capacity but rather to a more efficient use of model parameters.



MODULE 4 :VGG16

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters

(11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU's.



4.3 NON-FUNCTIONAL REQUIREMENTS:

Non functional requirements are the constraints that must be adhered to during development. They limit what resources can be used and set bounds on aspect of the software's quality. One of the most important things about non-functional requirements is to make them verifiable. The verification is normally done by measuring various aspects of the system and seeing if the measurements confirm to the requirements.

4.4 SOFTWARE REQUIREMENT SPECIFICATION:

Requirements analysis in system engineering software engineering encompasses those tasks that go into determining the needs or conditions to meet for a new or

altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. Systematic requirements analysis is also known as requirements engineering. It is sometimes referred to loosely by names such as requirements gathering, requirements capture or requirement specification .The term requirements analysis can also be applied especially to the analysis proper(as opposed to elicitation or documentation of the requirements, for instance). To develop the system, The required software and hardware requirements are as listed below.

4.4.1 HARDWARE REQUIREMENTS:

- RAM : 8GB
- System : Intel core
- Hard disk : 40GB
- CPU Frequency : 2GHZ and above

4.4.2 SOFTWARE REQUIREMENTS:

- Programming Language : python
- Operating system :- windows 10

4.4.3 SOFTWARE TOOLS:

- Programming : python 3.6.8
- IDE : Jupiter notebook
- Libraries : Numpy , cv2 , Matplotlib , Keras , tensorflow , os
- Web framework : Flask

4.5 SOFTWARE DEVELOPMENT ENVIRONMENT:

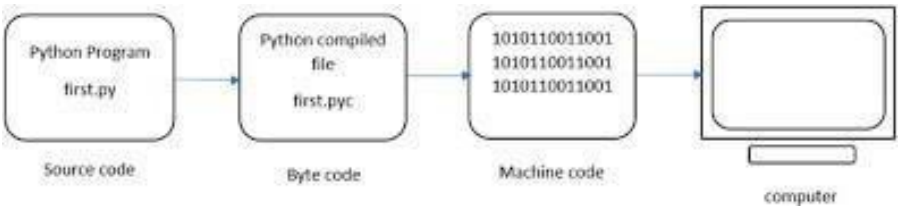
4.5.1 AN INTRODUCTION TO PYTHON:

Python is an object-oriented, high-level programming language with dynamic semantics. Its high-linter pretedevel built in data structures, combined with dynamic typing and dynamic binding make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Some of the benefits of python programming is as shown below

- Presence Of Third Party Modules
- Open source and Community Development
- Extensive Support Libraries
- User-Friendly Data Structures

Python doesn't convert its code into machine code, something that hardware can understand. It actually converts it into something called byte code. So within python, compilation happens, but it's just not into a machine language. It is into byte code and this byte code can't be understood by the CPU. So we actually need an interpreter called the python virtual machine. The python virtual machine executes the byte codes.



4.5.2 CHARACTERISTICS OF PYTHON:

Following are important characteristics of Python Programming

➔ It supports automatic garbage collection.

➔ It supports functional and structured programming methods as well as OOP.

➔ It can be used as a scripting language or can be compiled to byte-code for building large applications.

➔ It provides very high-level dynamic data types and supports dynamic type checking.

➔ It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

4.5.3 APPLICATIONS OF PYTHON:

As mentioned before, Python is one of the most widely used languages over the web. I'm going to list few of them here:

➔ Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax.This allows the student to pick up the language quickly.

➔ Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

➔Extendable − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

➔GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

4.5.4 INSTALLATION OF PYTHON 3.6.8:

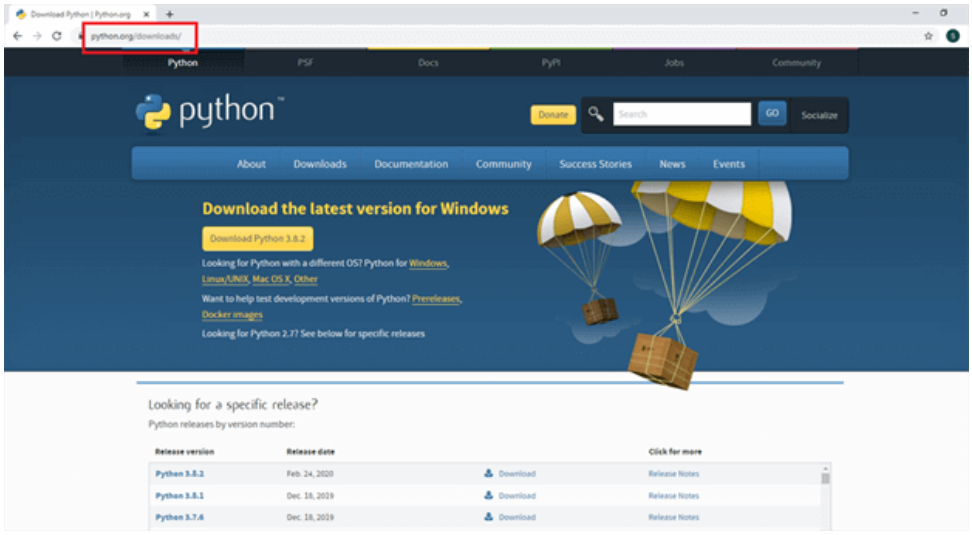
Introduction

I will explain how to install Python on Windows. After installing Python, we will run a simple program in idle Python editor and command prompt. Python is a free open source softwareavailable on different platforms such as Windows, Linux, or macOS. Verify if your computer is 64 or 32 bits. Select either Windows x86-64 executable installer for 64-bit orWindows x86 executable installer for 64-bit for 32-bit.

HOW TO INSTALL PYTHON?

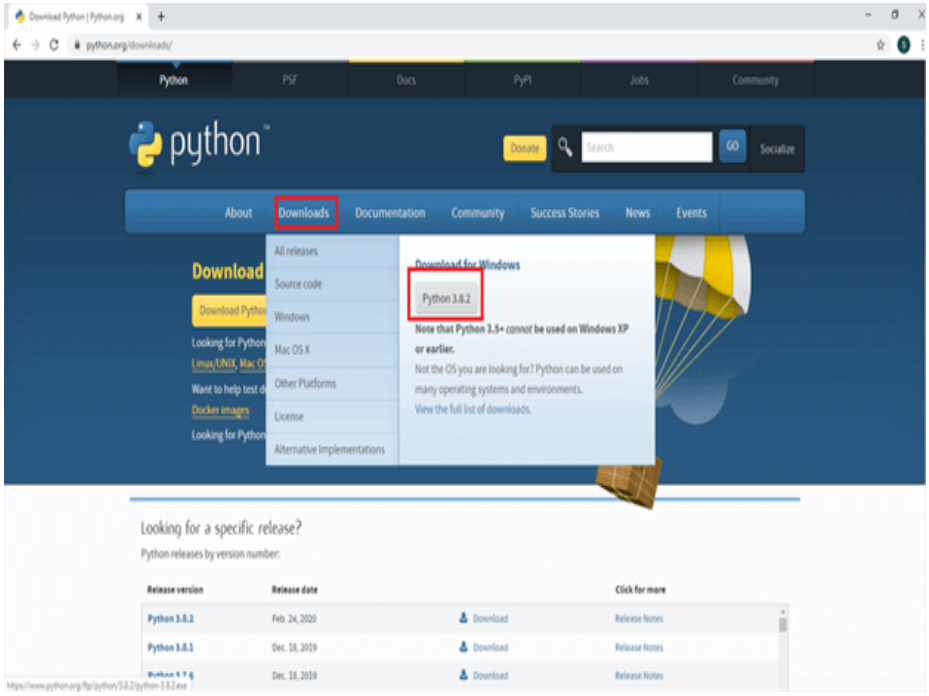
Step 1

Click this link, it will take you to the Python official download website.



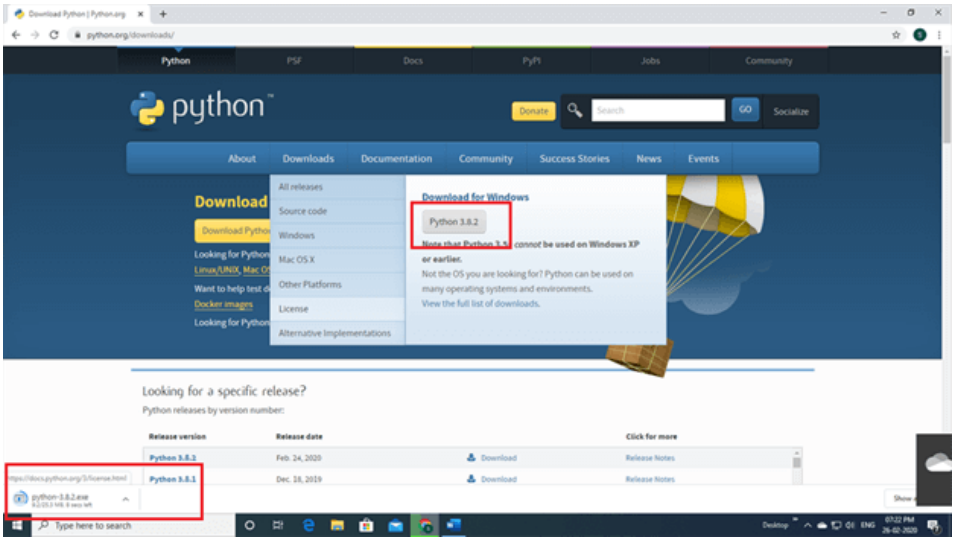
Step 2:

Click the download button and you will see Python 3.8.2.



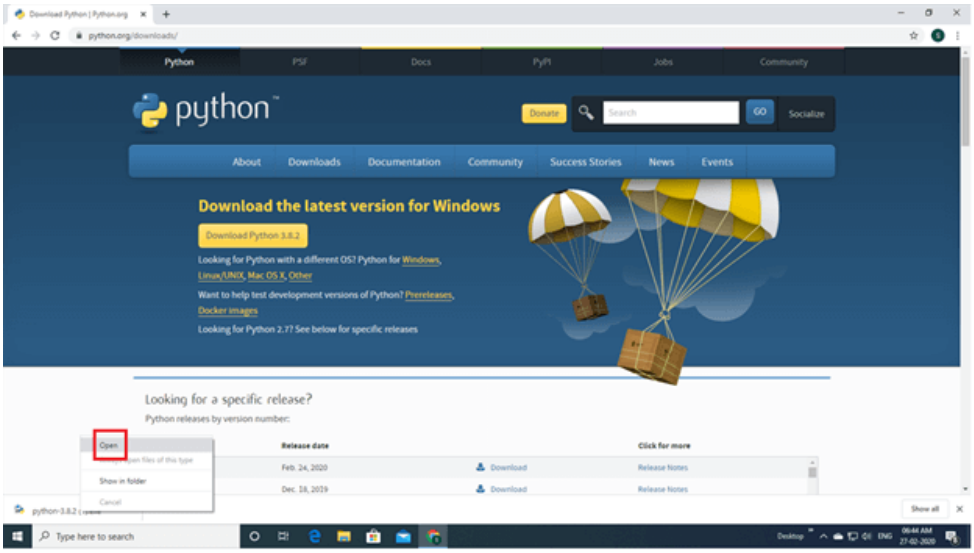
Step 3

Click Python 3.8.2 and Python will start to download.



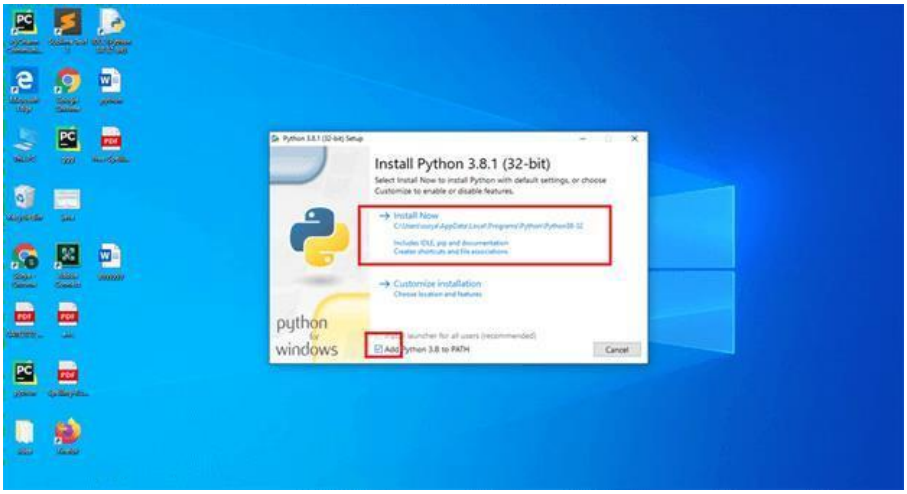
Step 4

Next, right click the mouse button you will see open button click to open.



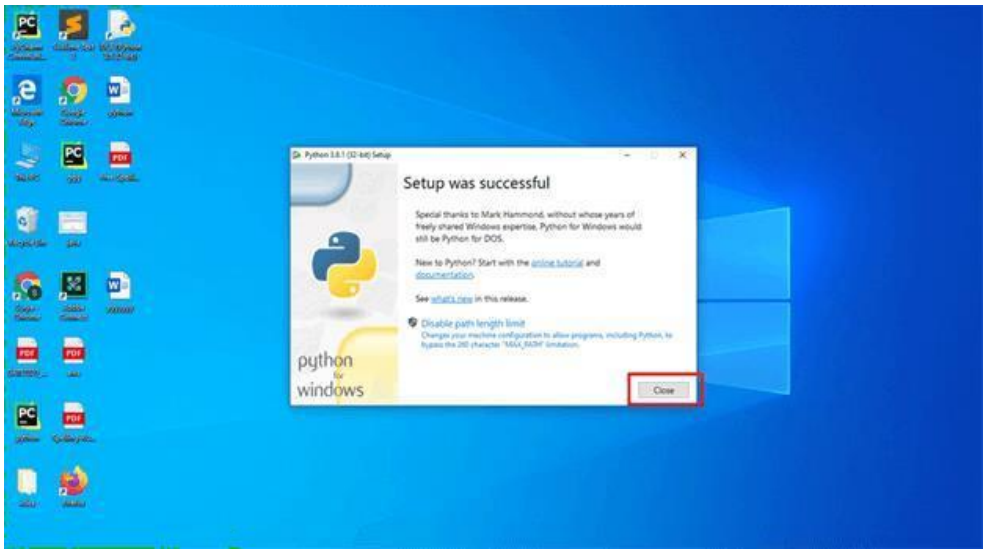
Step 5

Enable to add Python 3.8 to path and click install now.



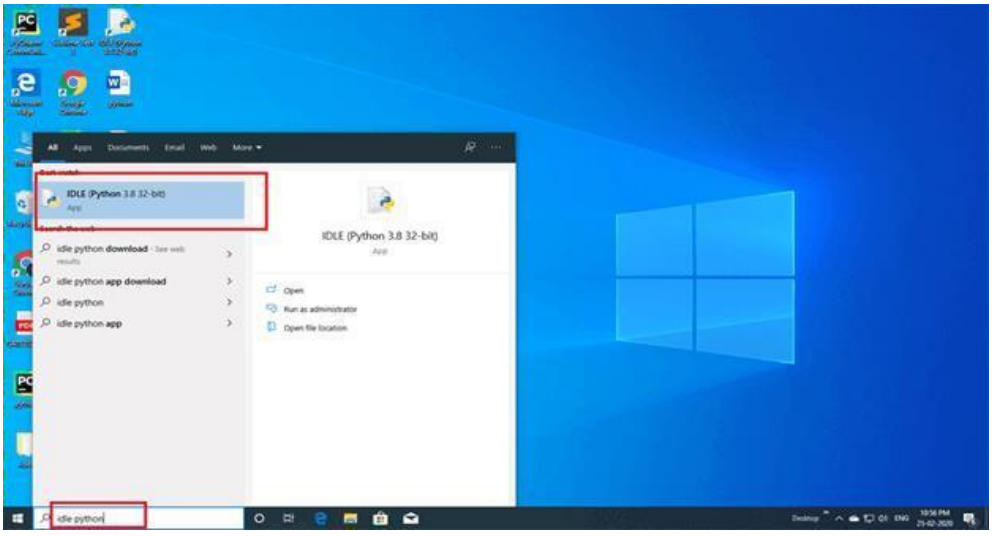
Step 6

Wait a few minutes and display setup was successful. Next you will click the close button.



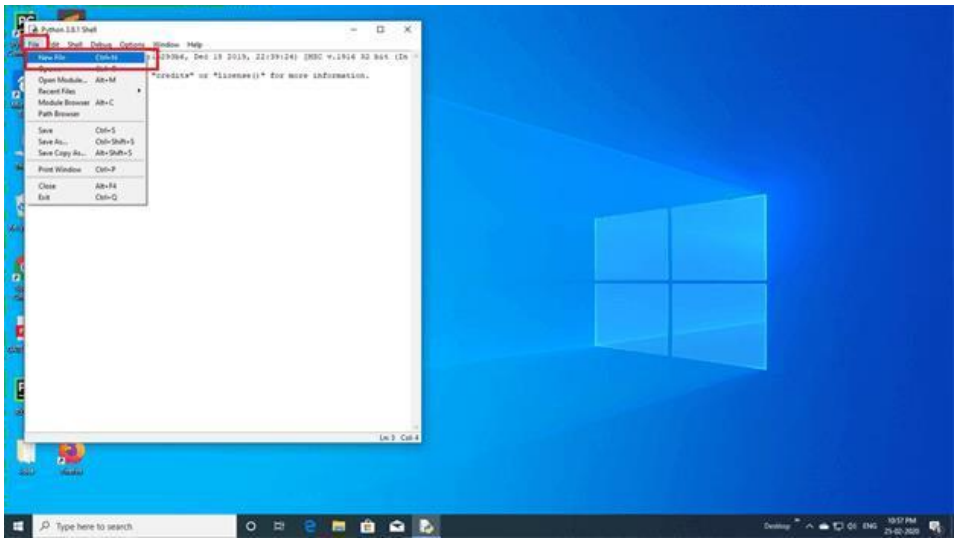
STEP 7

Next you will search for IDLE Python, then click enter.



Step 8

Next, click the file button. You will see a new file again. Click the new file button.



4.6 JUPYTER NOTEBOOK :

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.Getting Up and Running With Jupyter Notebook

The Jupyter Notebook is not included with Python, so if you want to try it out, you will need to install Jupyter.

There are many distributions of the Python language. This article will focus on just two of them for the purposes of installing Jupyter Notebook. The most popular is CPython, which is the reference version of Python that you can get from their website. It is also assumed that you are using Python 3.

Installation

If so, then you can use a handy tool that comes with Python called pip to install Jupyter Notebook like this:

```
Shell
$ pip install jupyter
```

The next most popular distribution of Python is Anaconda. Anaconda has its own installer tool called conda that you could use for installing a third-party package. However, Anaconda comes with many scientific libraries preinstalled, including the Jupyter Notebook, so you don’t actually need to do anything other than install Anaconda itself.

Starting the Jupyter Notebook Server

Now that you have Jupyter installed, let's learn how to use it. To get started, all you need to do is open up your terminal application and go to a folder of your choice. I recommend using something.

like your Documents folder to start out with and create a subfolder there called *Notebooks* or something else that is easy to remember.

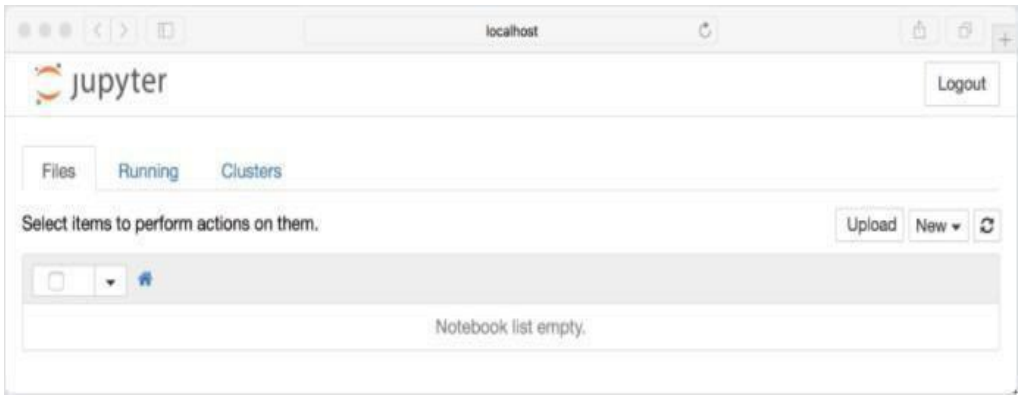
Then just go to that location in your terminal and run the following command:

```
Shell

$ jupyter notebook
```

This will start up Jupyter and your default browser should start (or open a new tab) to the following URL: http://localhost:8888/tree

Your browser should now look something like this:



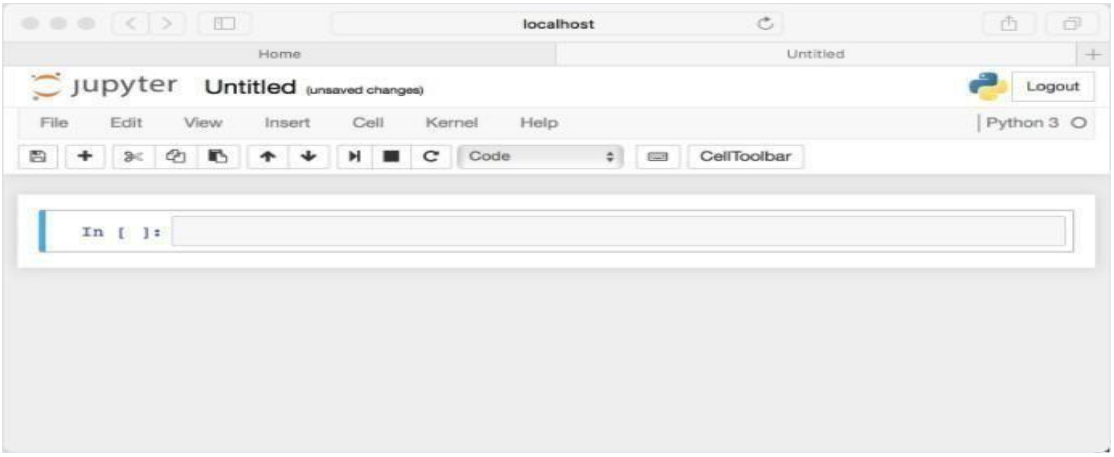
Note that right now you are not actually running a Notebook, but instead you are just running the Notebook server. Let's actually create a Notebook now!

Creating a Notebook

Now that you know how to start a Notebook server, you should probably learn how to create an actual Notebook document.

All you need to do is click on the *New* button (upper right), and it will open up a list of choices. On my machine, I happen to have Python 2 and Python 3 installed, so I can create a Notebook that uses either of these. For simplicity's sake, let's choose Python .

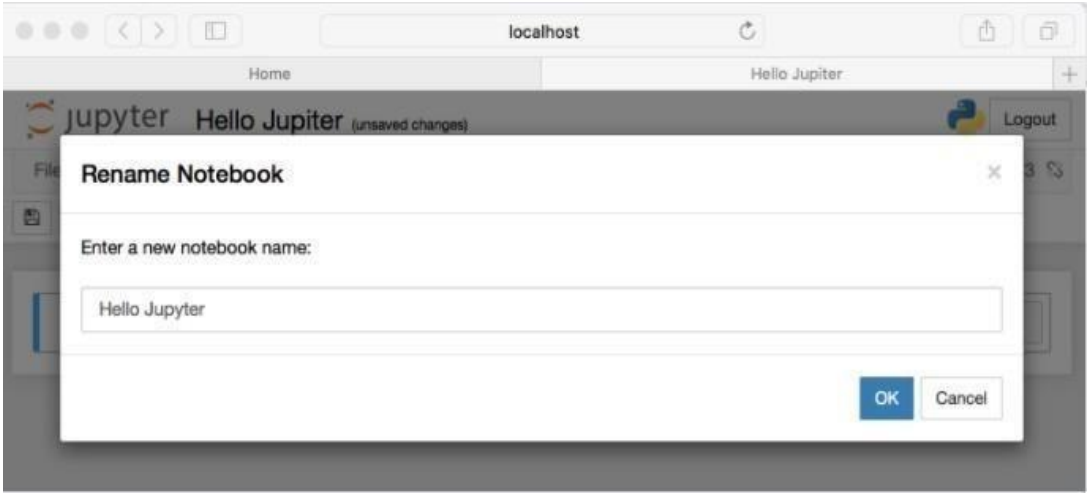
Your web page should now look like this:



Naming

You will notice that at the top of the page is the word *Untitled*. This is the title for the page and the name of your Notebook. Since that isn't a very descriptive name, let's change it!

Just move your mouse over the word *Untitled* and click on the text. You should now see an in- browser dialog titled *Rename Notebook*. Let's rename this one to *Hello Jupyter*:



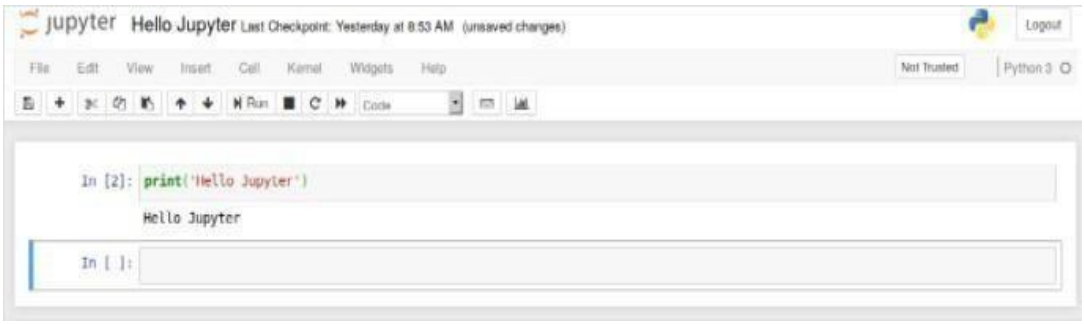
Running Cells

A Notebook's cell defaults to using code whenever you first create one, and that cell uses the kernelthat you chose when you started your Notebook.

In this case, you started yours with Python 3 as your kernel, so that means you can write Python code in your code cells. Since your initial Notebook has only one empty cell in it, the Notebook can't really do anything.

Thus, to verify that everything is working as it should, you can add some Python code to the cell and try running its contents.

Let's try adding the following code to that cell:



If you have multiple cells in your Notebook, and you run the cells in order, you can share your variables and imports across cells. This makes it easy to separate out your

code into logical chunks without needing to re import libraries or recreate variables or functions in every cell.

When you run a cell, you will notice that there are some square braces next to the word *In* to the left of the cell. The square braces will auto fill with a number that indicates the order that you ran the cells. For example, if you open a fresh Notebook and run the first cell at the top of the Notebook,the square braces will fill with the number *1*.

The Menus

The Jupyter Notebook has several menus that you can use to interact with your Notebook. The menu runs along the top of the Notebook just like menus do in other applications. Here is a list of the current menus:

- *File*
- *Edit*
- *View*
- *Insert*
- *Cell*
- *Kernel*
- *Widgets*
- *Help*

Let's go over the menus one by one. This article won't go into detail for every single option in every menu, but it will focus on the items that are unique to the Notebook application.

The first menu is the File menu. In it, you can create a new Notebook or open a preexisting one. This is also where you would go to rename a Notebook. I think the most interesting menu item is the *Save and Checkpoint* option. This allows you to create checkpoints that you can roll back to if you need to.

Next is the *Edit* menu. Here you can cut, copy, and paste cells. This is also where you would go if you wanted to delete, split, or merge a cell. You can reorder cells here too.

Note that some of the items in this menu are greyed out. The reason for this is that they do not apply to the currently selected cell. For example, a code cell cannot have an image inserted into it, but a Markdown cell can. If you see a greyed out menu item, try changing the cell's type and see if the item becomes available to use.

The *View* menu is useful for toggling the visibility of the header and toolbar. You can also toggle *Line Numbers*
within cells on or off. This is also where you would go if you want to mess about with the cell's toolbar.

The *Insert* menu is just for inserting cells above or below the currently selected cell.

The *Cell* menu allows you to run one cell, a group of cells, or all the cells. You can also go here to change a cell's type, although I personally find the toolbar to be more intuitive for that.

The other handy feature in this menu is the ability to clear a cell's output. If you are planning to share your Notebook with others, you will probably want to clear the output first so that the next person can run the cells themselves.

The *Kernel* cell is for working with the kernel that is running in the background. Here you can restart the kernel, reconnect to it, shut it down, or even change which kernel your Notebook is using. You probably won't be working with the Kernel all that often, but there are times when you are debugging a Notebook that you will find you need to restart the Kernel. When that happens, this is where you would go.The *Widgets* menu is for saving and clearing widget state. Widgets arebasically JavaScript widgets that you can add to your cells to make dynamic content using Python (or another Kernel).Finally you have the *Help* menu, which is where you go to learn about the Notebook's keyboard shortcuts, a user interface tour, and lots of reference material.

Notebook Extensions

While Jupyter Notebooks have lots of functionality built in, you can add new functionality through extensions. Jupyter actually supports four types of extensions:

- Kernel
- IPython kernel
- Notebook
- Notebook server

This tutorial will focus on Notebook extensions. What Are Extensions?

A Notebook extension (nb extension) is a JavaScript module that you load in most of the views in the Notebook's front end. If you are handy with JavaScript, you can even write your own extension. An extension can access the page's DOM and the Jupyter JavaScript API.

Where Do I Get Extensions?

You can use Google or search for Jupyter Notebook extensions. There are actually quite a few out there. One of the most popular extension sets is called jupyter_contrib_nb extensions, which you can get from GitHub. This is actually a collection of extensions that is provided by the Jupyter community and installed with pip.

How Do I Install Them?

Most Jupyter Notebook extensions can be installed using Python's pip tool. If you find an extension that can't be installed with pip, then you will likely have to use the following command:

```
Shell
$ jupyter nbextension install EXTENSION_NAME
```

This only installs the extension but does not make it active. You will need to enable an extension after installing it by running the following:

```
Shell

$ jupyter nbextension enable EXTENSION_NAME
```

You may need to restart your Jupyter Notebook kernel to see the extension.

There is a nice meta extension called Jupyter NbExtensions Configurator that is worth getting for managing other extensions. It allows you to enable and disable your extensions from within the Jupyter Notebook's user interface and also shows all the currently installed extensions.

4.7 LIBRARIES:

NumPy :

NumPy is the fundamental package for scientific computing in Python. ..NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences. NumPy is a general- purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python

Cv2:

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. cv2. imread() Smethod loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix. OpenCV is a cross- platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

Matplotlib :

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Keras:

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. *Being able to go from idea toresult as fast as possible is key to doing good research.*

Keras is:

- **Simple** -- but not simplistic. Keras reduces developer *cognitive load* to free you to focus on the parts of the problem that really matter.
- **Flexible** -- Keras adopts the principle of *progressive disclosure of complexity*: simple workflows should be quick and easy, while arbitrarily advanced workflows should be *possible* via a clear path that builds upon what you've already learned.
- **Powerful** -- Keras provides industry-strength performance and scalability:

TensorFlow:

It is an open source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainly used for: Classification, Perception, Understanding, Discovering, Prediction and Creation. TensorFlow is a Python library for fast numerical

computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

OS :

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system

4.8 FEASIBILITY STUDY:

The feasibility of the project is analyzed in this phase and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company.

For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

4.8.1 ECONOMICAL FEASIBILITY:

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.8.2 TECHNICAL FEASIBILITY:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on

the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.8.3 SOCIAL FEASIBILITY:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5. SYSTEM DESIGN

5.1 UML DIAGRAMS:

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with unified modelling language. The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS: The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.

7. Integrate best practices

5.2 BUILDING BLOCKS OF UML:

The vocabulary of the UML encompasses 3 kinds of building blocks

Things:

Things are the data abstractions that are first class citizens in a model. Things are of 4 types Structural Things, Behavioral Things, Grouping Things, and notational Things.

5.3 Relationships:

Relationships tie the things together. Relationships in the UML are Dependency, Association, Generalization, and Specialization

5.4 UML Diagrams:

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships).
There are two types of diagrams, they are:

➔ Structural Diagram

➔ Behavioral Diagrams

Structural Diagrams:

The UML‘s four structural diagrams exist to visualize, specify, construct and document the static aspects of a system. I can View the static parts of a system using one of the following diagrams. Structural diagrams consist of Class Diagram, Object Diagram, Component Diagram, and Deployment Diagram.

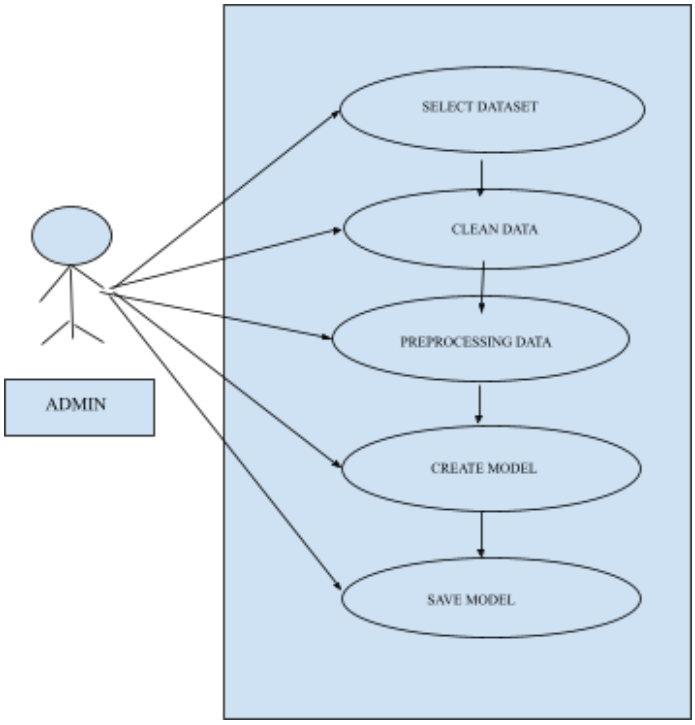
Behavioral Diagrams:

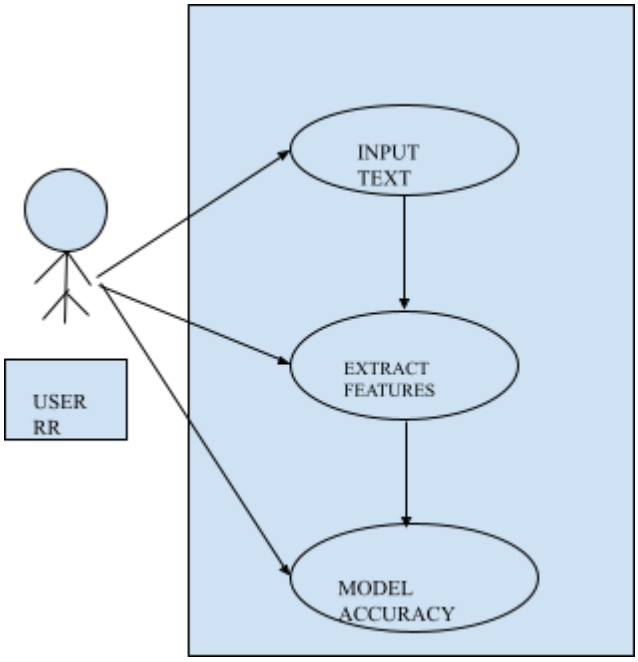
The UML’s five behavioral diagrams are used to visualize, specify, construct, and document the dynamic aspects of a system. The UML’s behavioral diagrams are roughly organized around the major ways which can model the dynamics of a system.

Behavioral diagrams consists of Use case Diagram, Sequence Diagram, Collaboration Diagram, State chart Diagram, Activity Diagram.

5.4.1 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

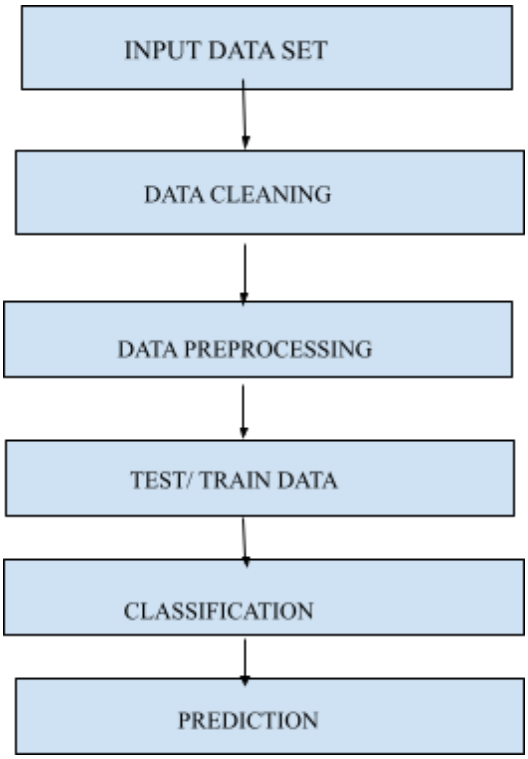




5.4.2 DATA FLOWDIAGRAM:

- The DFD is also called a bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
- The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in thesystem.
- DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
- DFD is also known as a bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent

increasing information flow and functional detail.



5.5 DATASET:

Data Description:

The dataset used to train and evaluate our models consists 6,216 chest X-ray images across6,001 patients. To generate this dataset, we combine and modify two different datasets that are publicly available: COVID Chest X-Ray Dataset [CMD20] and Labeled Optical Coherence Tomography (OCT) and Chest X- Ray Images [KZG18]. The following section gives a description of the two datasets and a summary of how we generate our dataset:

COVID Chest X-Ray Dataset: This dataset comprises 360 chest X-ray images and CT scans across 198 patients which are positive or suspected of COVID-19 pneumonia or other pneumonias. In this study, we only select 232 chest X-ray images with anteroposterior or posteroanterior chest view from 145 patients who are positive

or suspected of COVID-19 pneumonia.

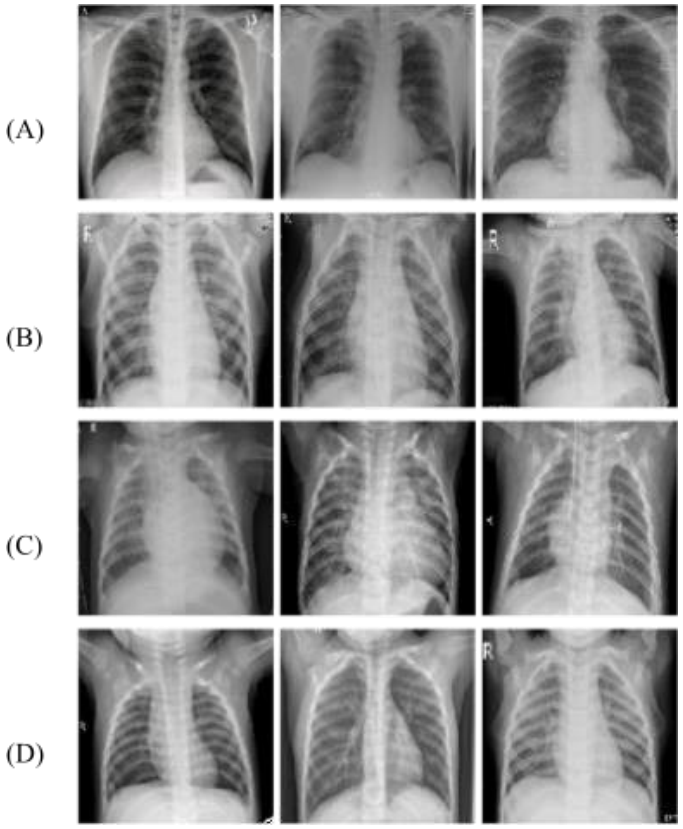
Labeled OCT and Chest X-Ray Images: This dataset has a total of 109,309 OCT images and 5,856 chest X-ray images that have been collected and labeled by the same lab. The chest X-ray images include 4,273 pneumonia images (1,493 viral and 2,780 bacterial) and 1,583normal images from total of 5,856 patients. X-ray images are already split into training set and testing set. In this study, we choose all chest X-ray images and use the predefined training set and testing set.

The summary of our dataset is shown in Table 2.1. Sample chest X-ray images for each pneumonia type are shown in Figure.

Table 2.1: Details of our dataset

Original Dataset	Type	Total Patients	Total Images
COVID Chest X-Ray Dataset	COVID-19	145	232
Labeled OCT and Chest X-Ray Images	Viral	1493	1493
	Bacterial	2780	2780
	Normal	1583	1583

Figure 2.1: Sample chest X-ray images from our dataset: (A) COVID-19 pneumonia, (B) viral pneumonia, (C) bacterial pneumonia, and (D) normal case.



Dataset Preprocessing:

Our dataset consists of 232 COVID-19 pneumonia images, 1,493 viral pneumonia images, 2,780 bacterial pneumonia images and 1,583 normal case images, shown in Table 2.1. For COVID-19 pneumonia images, there are multiple images from some of the patients. In order to ensure that images of each patient appear only in training set or testing set, we separate the data by patient before splitting into training and testing set. Among 232 COVID-19 pneumonia images, 172 images fall into the training set and the rest 60 images fall into the testing set. For chest X-ray images of the other pneumonia types, we use the predefined training set and testing set in the "Labeled OCT and Chest X-Ray Images" dataset in our four-class classification model, and label all of them as non-COVID-19 cases in our binary classification model. The data distributions of the binary classification problem and four-class classification problem are shown in Figure 2.2.

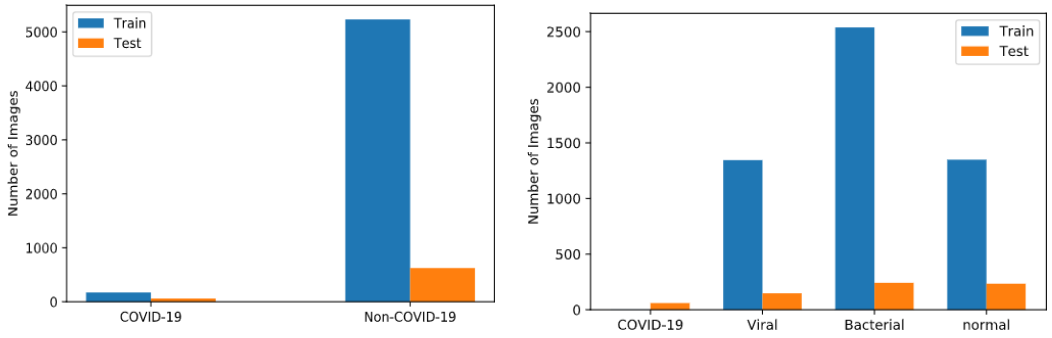


Figure 2.2: Data distribution for: (A) binary classification, and (B) four-class classification.

A noticeable problem is the limited amount of COVID-19 pneumonia images, which causes the data to be imbalanced and adds challenge to the performance of our classification models. To alleviate this problem, we use over-sampling method to even-up the data between classes. More specifically, we make COVID-19 pneumonia images 8-fold in the four-class classification problem and 24-fold in the binary classification problem. The details of our final training sets and testing sets are shown in Table 2.2 and Table 2.3. The final data distribution of two problems are shown in Figure 2.3.

Table 2.2: Details of training and testing set for the binary classification model

Label	Pneumonia Type	Training Set	Testing Set	Total
Non-COVID-19	Viral, Bacterial, and Normal (no infection)	5232	624	5856
COVID-19	COVID-19	4128	60	4188

Label	Pneumonia Type	Training Set	Testing Set	Total
COVID-19	COVID-19	1376	60	1436
Viral	Viral	1345	148	1493
Bacterial	Bacterial	2538	242	2780
Normal	Normal (no infection)	1349	234	1583

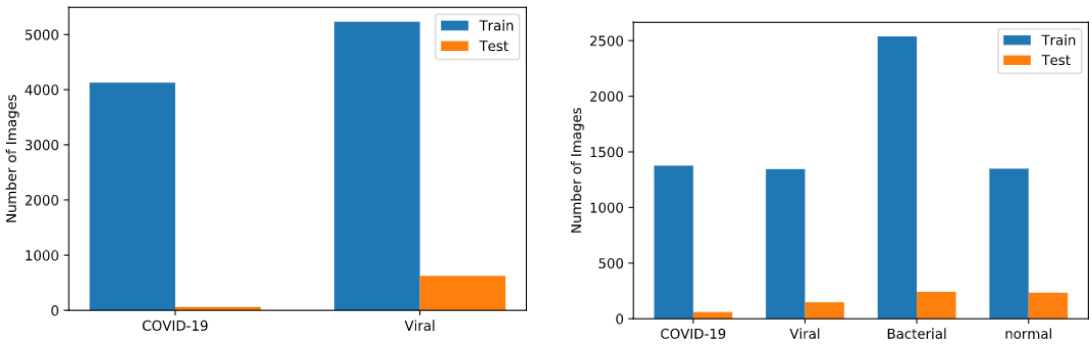


Figure 2.3: Final data distribution for: (A) binary classification, and (B) four-class classification.

6. IMPLEMENTATION

6.1 MODULE IMPLEMENTATION:

6.1.1 PREPROCESSING:

A set of noise-removal functions accompanied with morphological operations that result in clear image of chest CT scan image after passing through high pass filter is the basic idea behind the proposed algorithm. The set of morphological operations used will decide the clarity and quality of the chest ct scan image image After the original image undergoes pre-processing transformations These basic pre-processing transformations include:

- Changing the image to greyscale, as we need to find contour of the final image which works on greyscale images.
- Applying low pass filter, to remove any noise, if present, in the image.
- Applying high pass filter, to obtain sharpened image with clear-defined boundaries.

6.1.2 SEGMENTATION:

The K-means clustering algorithm. The main idea of this approach is to assign each point or data toa cluster whose center (centroid) is the closest. The center of each cluster is the average of all the points in this cluster classification we define a class for the black background of the image with cerebrospinal fluid, a second class for white matter and a third class for gray matter.

6.1.3 ResNet ARCHITECTURE:

Residual neural network (ResNet) is proposed by He et al. in 2015 [HZR16]. The hypothesis behind ResNet is that deeper networks are harder to optimize, since the deeper model should be able to perform as well as the shallower model by copying the learned parameters from the shallower model and setting additional layers to identity mapping. To help optimize deeper models,residual blocks are designed to fit a residual mapping $F(x)$ instead of the desired underlying mapping $H(x)$, and full ResNet

architecture is built by stacking residual blocks. More specifically, every residual block has two 3 × 3 convolutional layers. Periodically, the number of filters are doubled and spatial downsampling is operated. Figure 3.1 illustrates the structure within the residual block and Figure

3.2 shows an example of ResNet architecture - ResNet with 18 layers (ResNet18).

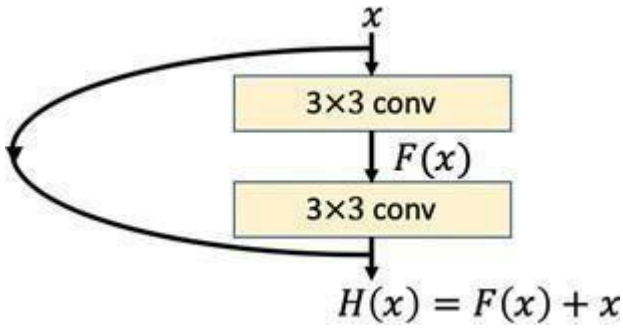


Figure 3.1: Structure of residual block

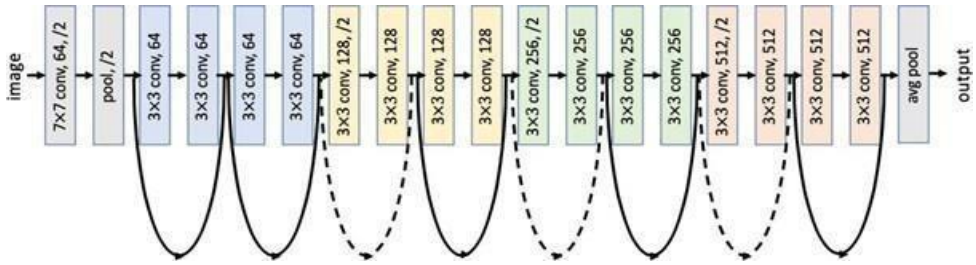


Figure 3.2: Schema of ResNet18 architecture

In this study, we build two deep models based on ResNet18 for the classification under two schemes. One is a binary classification that distinguishes COVID-19 pneumonia and non-COVID-19 cases, while the other is a four-class classification that classifies COVID-19 pneumonia, viral pneumonia, bacterial pneumonia, and normal cases.

6.1.4 TRANSFERRING LEARNING

A successful training of deep neural networks often requires a large-scale dataset and a long training period. Moreover, a basic assumption for many deep learning models is that the training and testing data should be drawn from the same distribution. In many real-word applications, the availability of data is limited due to the high cost of collecting and labelling data. In such cases, retaining and reusing previously learned knowledge for a different data distribution, task or domain is critical. Transfer learning is a machine learning method where a previously trained model is reused as a starting point for the new model and task. It can not only alleviate the need and effort to collect training data, but also accelerate the training process.

There are two commonly used strategies to exploit transfer learning on deep convolutional neural network. The first strategy is called feature extraction where the pretrained model, retaining both its initial architecture and the learned parameters, is only used to extract image features for the input of the new classification model. The second strategy makes some modifications to the pretrained model, such as architecture adjustments and parameter tuning, to improve extracted image features on the new dataset and achieve optimal results.

In this study, the second strategy is used. More specifically, we use ImageNet[RDS15], a large-scale dataset consisting 1.2 million high-resolution images in 1,000 different classes (i.e., fish, bird, tree, flowers, sport, room, etc.), to pretrain ResNet18. Since images in ImageNet are different from our dataset, we cannot directly use the pretrained model to extract image features for classification. Instead, we need to fine-tune parameters of the pretrained model on our dataset to generate better image features. Besides, the architecture of ResNet18 has been changed -the number of outputs in the final fully-connected layer is changed from 1,000 to the number of classes in each classification problem. Thus, the fine-tuned model outputs scores for each class and can classify input image to the class with maximum score. The schematic representation of the training process is depicted in Figure 3.3.

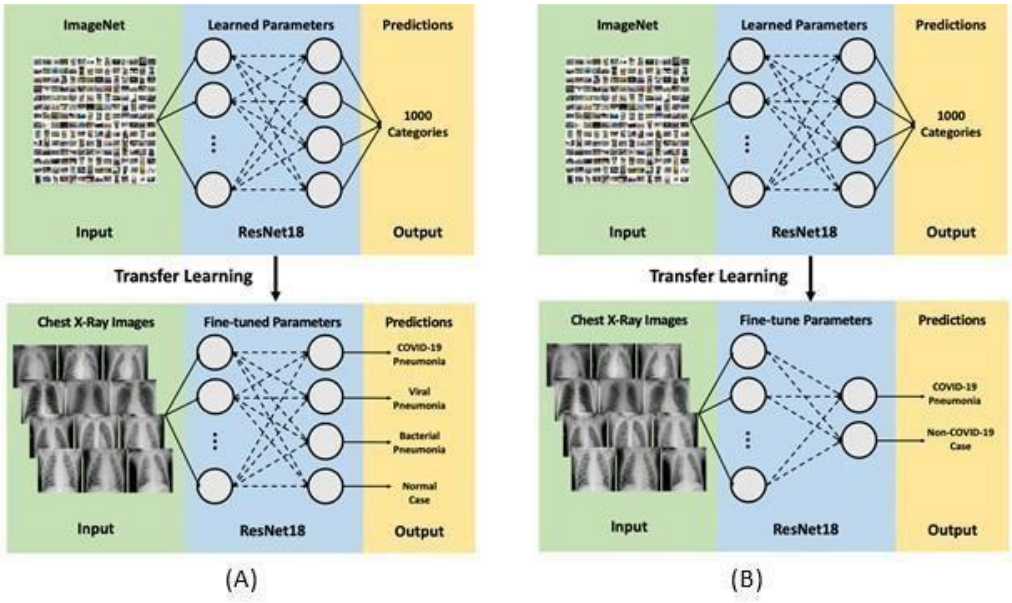


Figure 3.3: Schematic of transfer learning with ResNet18 for: (A) four-class classification problem, and (B) binary classification problem.

6.1.5 IMPLEMENTATION:

Data Augmentation:

We apply data augmentation methods that include rotation, scaling, translation, and horizontal-flip on the training set to address the data deficiency problem. That means, each image in the training set is randomly operated by the following methods:

- **Rotation:** Rotating the image with an angle between 0 and 15 in the clockwise or counter clockwise direction.
- **Scaling:** Sampling the scale of frame size of the image randomly between 90% and 110%.
- **Translation:** Translating image horizontally and vertically between -10% and 10%
- **Horizontal Flip:** Horizontally flipping the image with a probability of 0.5.

Hyper parameters:

In this study, all images are normalized and resized to 224 × 224 pixels. We use five-fold cross validation to evaluate the model and tune the hyper parameters, and obtain the generalized results in the testing stage. The following hyper parameters are used for

training

Binary classification model:

Batch size = 256, cross entropy loss, stochastic gradient descent (SGD) optimizer with momentum

= 0.9 and weight decay (L2 penalty) = 0.01, learning rate = 0.003 (will decay by 0.1 at 10 and 20 epoch), 50 epochs in total, and early stopping with patience = 10.

Four-class classification model: Batch size = 256, cross entropy loss, SGD optimizer with momentum

= 0.9 and weight decay (L2 penalty) = 0.1, cross entropy loss, learning rate = 0.003 (will decay by 0.1 at 20 and 40 epoch), 50 epochs in total, and early stopping with patience = 15.

Performance Evaluation Metrics

Due to the data imbalance problem, the performance of the model is evaluated using eight evaluation metrics accuracy, confusion matrix, precision, recall (or sensitivity), specificity (or selectivity), F1-score, receiver operating characteristic (ROC) curve and area under the curve(AUC).

Confusion Matrix

A confusion matrix is a summary of predicted labels and true labels on a classification problem. It gives us insights not only into the errors but also into the types of errors that are being made by the classifier. The schema of confusion matrix in a binary classification problem is shown in Figure

		Predicted Label	
		Positive	Negative
True Label	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Figure 3.4: Schema of confusion matrix

The accuracy, precision, recall (or sensitivity), specificity (or selectivity) and F1-score can be computed as follows based on the confusion matrix.

$$\text{accuracy} = \frac{TP + FN}{TP + FP + TN + FN}$$

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall (or sensitivity)} = \frac{TP}{TP + FN}$$

$$\text{Specificity (or selectivity)} = \frac{TN}{TN + FP}$$

$$\text{F1-score} = \frac{2 \times \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

For multi-class classification problems, we can not only use one-vs-all methodology to compute the above metrics but also use macro average and weighted average of metrics to evaluate the overall model performance. The macro average score is average of the target metric (e.g., accuracy) over all classes while the weighted average is average of the target metric over classes weighted by sample size. If each class has the same sample size, the macro average and the weighted average are always the same.

6.1.6 ROC CURVE and AUC:

ROC curve is a graphical plot that shows the performance of a binary classification model. It is created by plotting true positive rate (TPR) against false positive rate (FPR) at various discrimination thresholds, where TPR is also known as sensitivity or recall, and FPR can be calculated as (1 − specificity). By changing the model's discrimination threshold, confusion matrix may be different and a point in the ROC curve can be plotted.

The ROC curve for a random binary classification model would be a diagonal line

from (0,0) to(1,1). Any curve above the diagonal line represents good classification model which is better than random, while curve below the diagonal line means the model is worse than random. A perfect binary classification model would yield a straight line from (0,1) to (1,1), meaning 100% sensitivity and 100% specificity.

AUC refers to the area under the ROC curve, which is always between 0 and 1. Based on the concept of ROC curve, it can be concluded that higher the AUC, better the binary classification model. A random classifier has AUC of 0.5, and a great model has AUC close to 1. With AUC less than 0.5, the model tends to reciprocate the classes. A schema of ROC curve and AUC is shown in Figure 3.5.

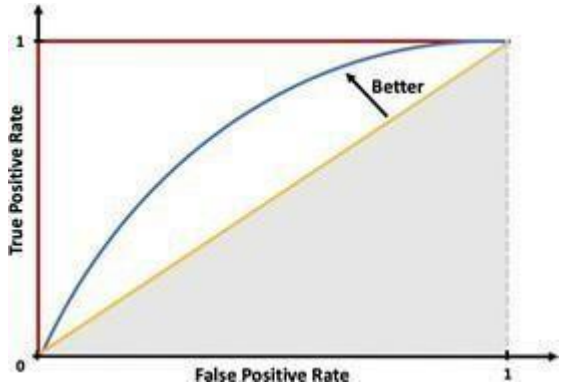


Figure 3.5: Schema of ROC curve and AUC : red line: a perfect classifier, blue curve: a great classifier, yellow line: a random classifier, and shaded area: AUC for the random classifier.

In a multi-class classification model, we can plot the ROC curve for each class using one-vs-all methodology.

Model Interpretation

Gradient-weighted class activation mapping (Grad-CAM) [SCD17] is a deep learning technique that can generate visual explanations of CNN-based models. Thus, it can make models more transparent and easier to interpret. Using the gradients of a target concept (i.e., the COVID-19 class in our four-class classification) flowing into the final

convolutional layer, it can produce a localization map showing important regions for the model to make predictions. Grad-CAM not only provides a way to evaluate models, but also helps humans to study and understand more about deeplearning models.

In our study, we apply Grad-CAM method on the four-class classification model and choose the COVID- 19 pneumonia as the target concept to understand how the four-class classification model makes predictions of COVID-19 cases.

6.2 SAMPLE CODE

```
from flask import Flask, render_template, request, session, redirect, url_for, flash
import os
from werkzeug.utils import secure_filename
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt
import cv2
import numpy as np

UPLOAD_FOLDER = './flask app/assets/images'
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg', 'gif'])
# Create Database if it doesnt exist

app = Flask(_name_,static_url_path='/assets',
            static_folder='./flask app/assets',
            template_folder='./flask app')
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
@app.route('/')
def root():
  return render_template('index.html')
@app.route('/index.html')
def index():
  return render_template('index.html')
@app.route('/contact.html')
def contact():
  return render_template('contact.html')
@app.route('/news.html')
def news():
  return render_template('news.html')
```

```
@app.route('/about.html')
def about():
  return render_template('about.html')
@app.route('/faqs.html')
def faqs():
  return render_template('faqs.html')
@app.route('/prevention.html')
def prevention():
  return render_template('prevention.html')
@app.route('/upload.html')
def upload():
  return render_template('upload.html')
@app.route('/upload_chest.html')
def upload_chest():
  return render_template('upload_chest.html')
@app.route('/upload_ct.html')
def upload_ct():
  return render_template('upload_ct.html')
@app.route('/uploaded_chest', methods = ['POST', 'GET'])
def uploaded_chest():
  if request.method == 'POST':
      # check if the post request has the file part
      if 'file' not in request.files:
          flash('No file part')
          return redirect(request.url)
      file = request.files['file']
      # if user does not select file, browser also
      # submit a empty part without filename
      if file.filename == '':
          flash('No selected file')
          return redirect(request.url)
```

```
            if file:
                # filename = secure_filename(file.filename)
                file.save(os.path.join(app.config['UPLOAD_FOLDER'],
'upload_chest.jpg'))

        resnet_chest = load_model('models/resnet_chest.h5')
        vgg_chest = load_model('models/vgg_chest.h5')
        inception_chest = load_model('models/inceptionv3_chest.h5')
        xception_chest = load_model('models/xception_chest.h5')

      image = cv2.imread('./flask app/assets/images/upload_chest.jpg') # read file
      image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
      # arrange format as per keras
        image = cv2.resize(image,(224,224))
        image = np.array(image) / 255
        image = np.expand_dims(image, axis=0)

        resnet_pred = resnet_chest.predict(image)
        probability =  resnet_pred[0]
        print("Resnet Predictions:")
        if probability[0] > 0.5:
          resnet_chest_pred = str('%.2f' % (probability[0]*100) + '% COVID')
        else:
          resnet_chest_pred = str('%.2f' % ((1-probability[0])*100) + '% NonCOVID')
        print(resnet_chest_pred)

        vgg_pred = vgg_chest.predict(image)
        probability = vgg_pred[0]
        print("VGG Predictions:")
        if probability[0] > 0.5:
          vgg_chest_pred = str('%.2f' % (probability[0]*100) + '% COVID')
```

```
            else:
              vgg_chest_pred = str('%.2f' % ((1-probability[0])*100) + '% NonCOVID')
            print(vgg_chest_pred)
            inception_pred = inception_chest.predict(image)
            probability = inception_pred[0]
            print("Inception Predictions:")
            if probability[0] > 0.5:
              inception_chest_pred = str('%.2f' % (probability[0]*100) + '% COVID')
            else:
              inception_chest_pred = str('%.2f' % ((1-probability[0])*100) + '%
NonCOVID')
            print(inception_chest_pred)
            xception_pred = xception_chest.predict(image)
            probability = xception_pred[0]
            print("Xception Predictions:")
            if probability[0] > 0.5:
              xception_chest_pred = str('%.2f' % (probability[0]*100) + '% COVID')
            else:
              xception_chest_pred = str('%.2f' % ((1-probability[0])*100) + '% NonCOVID')
            print(xception_chest_pred)

            return
render_template('results_chest.html',resnet_chest_pred=resnet_chest_pred,vgg_chest_pre
d=vgg_chest_pred,inception_chest_pred=inception_chest_pred,xception_chest_pred=xce
ption_chest_pred)

        @app.route('/uploaded_ct', methods = ['POST', 'GET'])
        def uploaded_ct():
          if request.method == 'POST':
              # check if the post request has the file part
              if 'file' not in request.files:
```

```
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        # if user does not select file, browser also
        # submit a empty part without filename
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file:
            # filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], 'upload_ct.jpg'))

    resnet_ct = load_model('models/resnet_ct.h5')
    vgg_ct = load_model('models/vgg_ct.h5')
    inception_ct = load_model('models/inception_ct.h5')
    xception_ct = load_model('models/xception_ct.h5')

    image = cv2.imread('./flask app/assets/images/upload_ct.jpg') # read file
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # arrange format as per
keras
    image = cv2.resize(image,(224,224))
    image = np.array(image) / 255
    image = np.expand_dims(image, axis=0)

    resnet_pred = resnet_ct.predict(image)
    probability = resnet_pred[0]
    print("Resnet Predictions:")
    if probability[0] > 0.5:
      resnet_ct_pred = str('%.2f' % (probability[0]*100) + '% COVID')
    else:
      resnet_ct_pred = str('%.2f' % ((1-probability[0])*100) + '% NonCOVID')
```

```
        print(resnet_ct_pred)

        vgg_pred = vgg_ct.predict(image)
        probability = vgg_pred[0]
        print("VGG Predictions:")
        if probability[0] > 0.5:
          vgg_ct_pred = str('%.2f' % (probability[0]*100) + '% COVID')
        else:
          vgg_ct_pred = str('%.2f' % ((1-probability[0])*100) + '% NonCOVID')
        print(vgg_ct_pred)

        inception_pred = inception_ct.predict(image)
        probability = inception_pred[0]
        print("Inception Predictions:")
        if probability[0] > 0.5:
          inception_ct_pred = str('%.2f' % (probability[0]*100) + '% COVID')
        else:
          inception_ct_pred = str('%.2f' % ((1-probability[0])*100) + '% NonCOVID')
        print(inception_ct_pred)

        xception_pred = xception_ct.predict(image)
        probability = xception_pred[0]
        print("Xception Predictions:")
        if probability[0] > 0.5:
          xception_ct_pred = str('%.2f' % (probability[0]*100) + '% COVID')
        else:
          xception_ct_pred = str('%.2f' % ((1-probability[0])*100) + '% NonCOVID')
        print(xception_ct_pred)

        return
render_template('results_ct.html',resnet_ct_pred=resnet_ct_pred,vgg_ct_pred=vgg_ct_pre
```

```
d,inception_ct_pred=inception_ct_pred,xception_ct_pred=xception_ct_pred)

        if__name__== '_main_':
           app.secret_key = ".."
           app.run()
```

7. SYSTEM TESTING

7.1 TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

7.2 TYPES OF TESTS

7.2.1 UNIT TESTING:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.2.2 INTEGRATIONS TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.2.3 FUNCTIONAL TESTING:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

-**valid Input** : identified classes of valid input must accepted.

-**Invalid Input** : identified classes of invalid input must be rejected.

-**Functions** : identified functions must be exercised.

-**Output** : identified classes of application outputs must be exercised.

-**Systems/Procedures**: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined

7.2.4 SYSTEM TESTING:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre- driven process links and integration points.

7.2.5 WHITE BOX TESTING:

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

7.2.6 BLACK BOX TESTING:

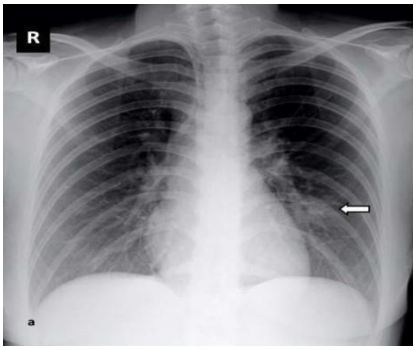
Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated as a black box you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7.2.7 ACCEPTANCE TESTING:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

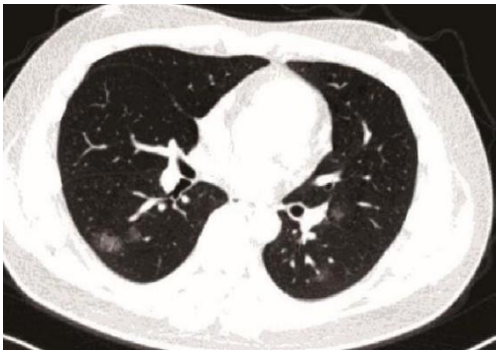
7.3 TEST CASES

1. Chest x-ray of covid +ve person person



2. Chest x-ray of covid –ve person

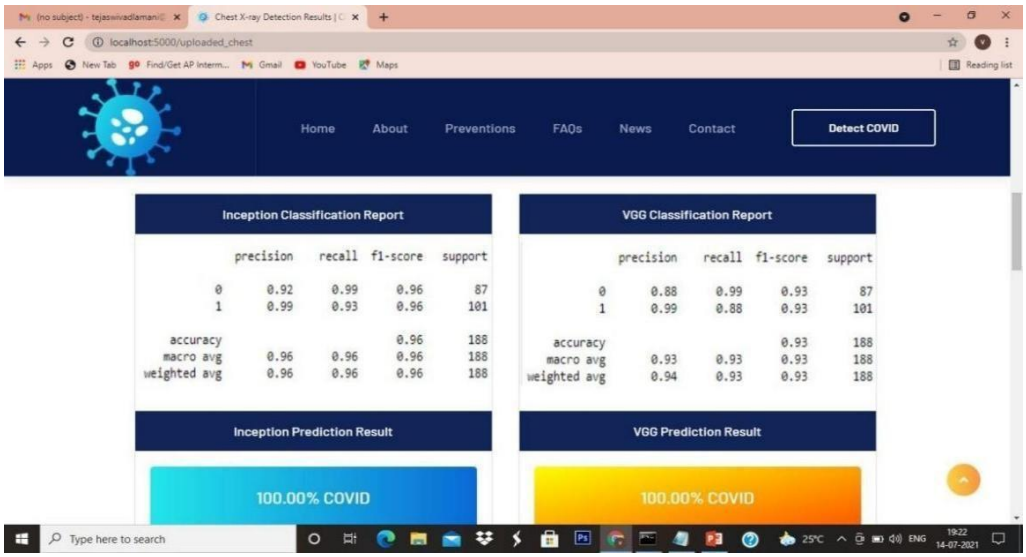
3. CT scan of covid +ve person person

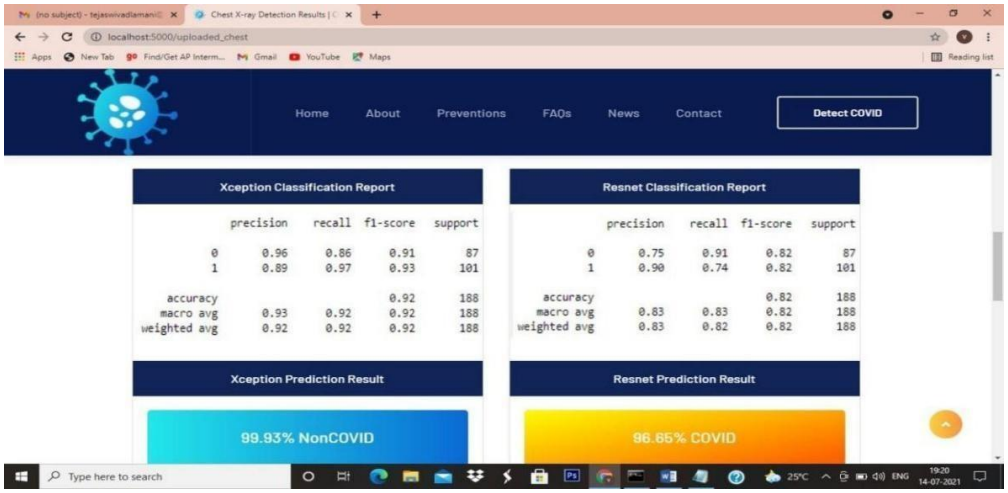


4. **CT scan of covid –ve person**

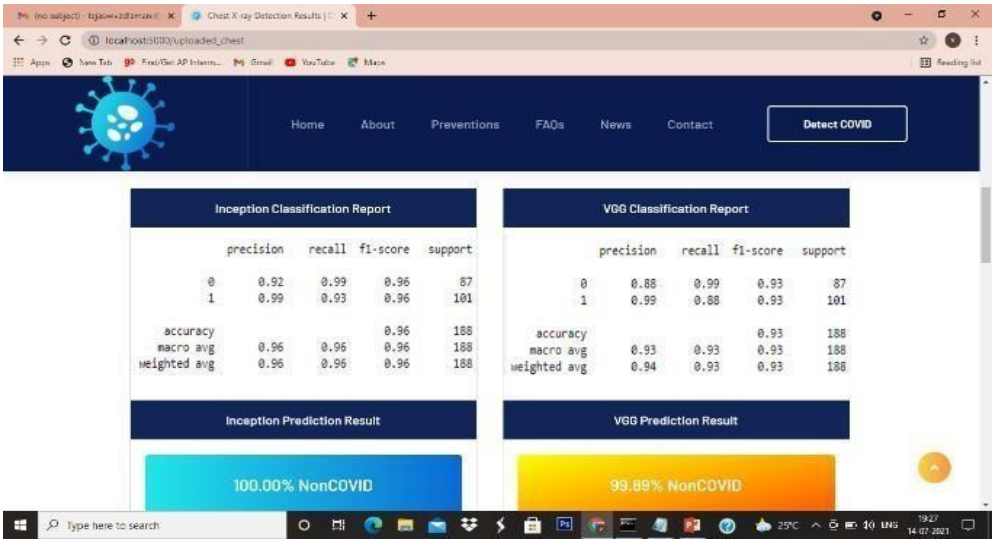
8. OUTPUT SCREENSHOTS

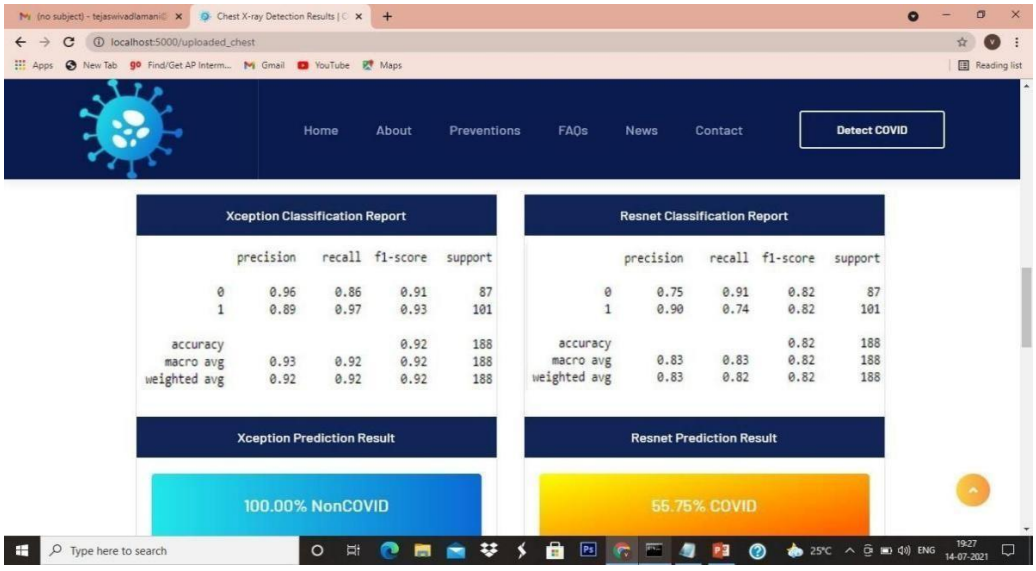
Output for chest x-ray which is covid positive



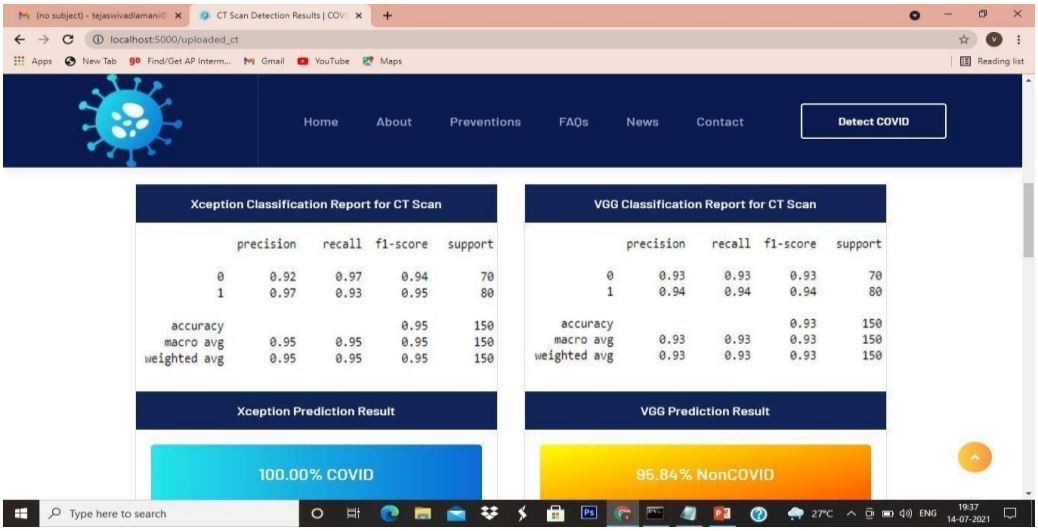


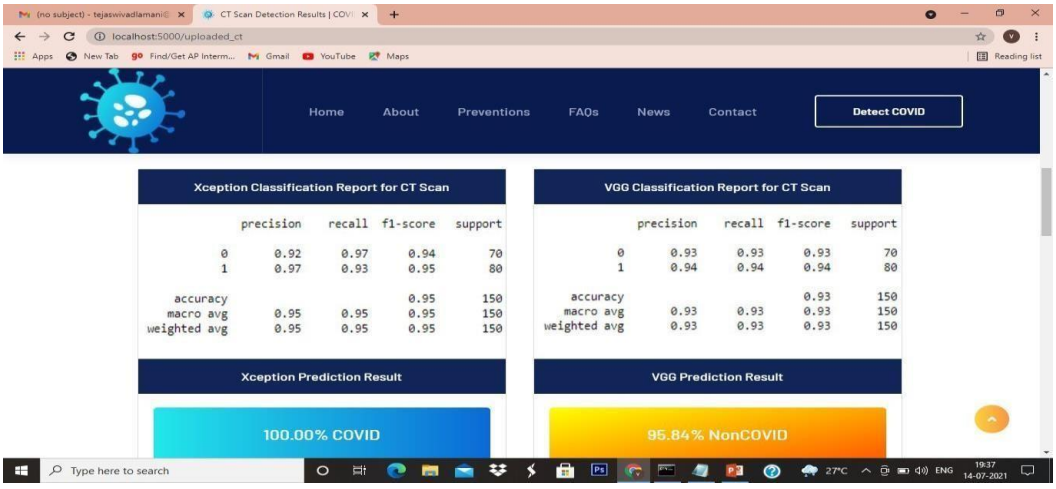
Output for chest x-ray which is covid negative



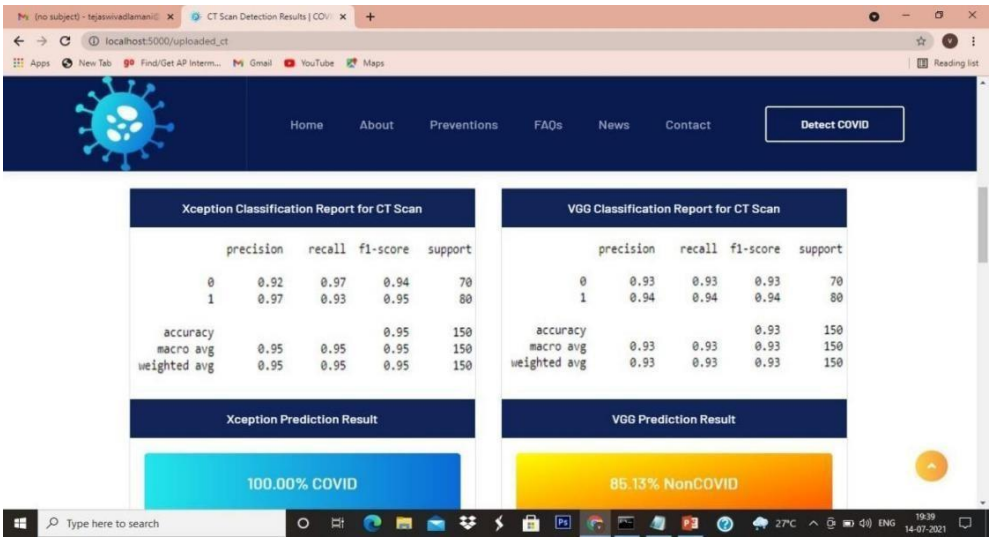


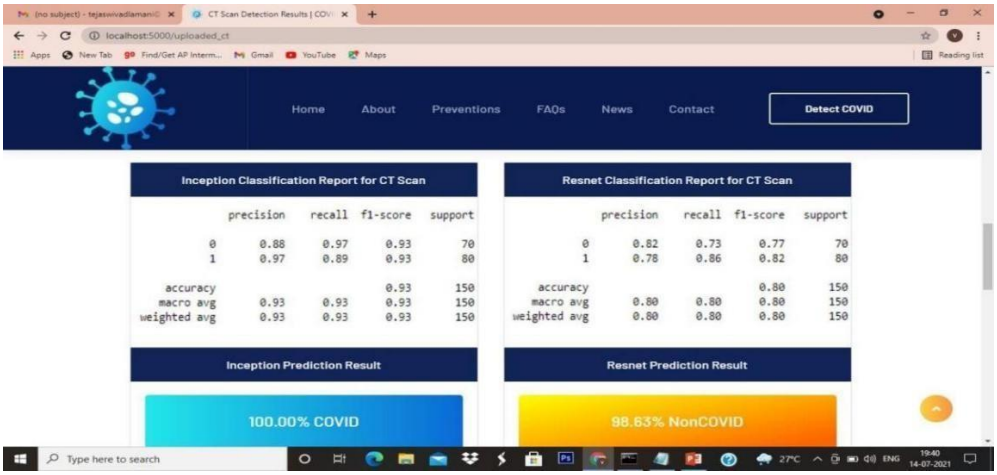
Output for CT-scan which is covid positive





Output for CT-scan which is covid negative





9. CONCLUSION

CONCLUSION:

In this study, we train two deep learning models based on transfer learning concept and ResNet18 architecture for COVID-19 pneumonia detection in chest X-ray images. The first oneis a binary classification model that aims to separate COVID-19 pneumonia and non- COVID-19cases. It is able to classify all test cases correctly. The second one is a four-class classification model that aims to distinguish COVID-19 pneumonia, viral pneumonia, bacterial pneumonia andnormal cases. It reaches an average accuracy, precision, sensitivity, specificity, F1-score, and AUC of 93%, 93%, 93%, 97%, 93%, and 99%, respectively. This model is able to detect all COVID-19 test cases, but it tends to classify normal cases into other classes and tends to classify viral pneumonia images into bacterial pneumonia. To shed a light on how the four-class classification model make predictions of COVID-19 pneumonia cases, we apply Grad-CAM method to generate localized map that highlights important regions the model thinks to make predictions. By comparing the highlighted regions and radiologists' descriptions of chest X-ray images, we find out that our model is more powerful when patchy areas are the main indicatorsof COVID-19 pneumonia. Otherwise, though our model can make accurate predictions, it tends to be unfocused and fails to highlight specific areas that indicating COVID-19 pneumonia.

COVID-19 has already become a global threat and has taken away hundreds of thousands of people's lives. This study shows the feasibility of building a computer-aided diagnostic system that can help clinicians detect COVID-19 pneumonia from radiology images accurately and quickly. Moreover, model interpretation techniques allow us to further evaluate and understand models. However, the limited data adds challenge to the performance of model

By collecting more chest X-ray images of COVID-19 pneumonia, other pneumonias and normal cases, the model will be more robust and powerful.

10. REFERENCE

[1] WorldometerAugust 2020, https://www.worldometers.info/ coronavirus/.

[2] F. Shi, J. Wang, J. Shi et al., "Review of artificial intelligence techniques in imaging data acquisition, segmentation anddiagnosis for COVID-19," *IEEE Reviews in Biomedical Engi-neering*, 2020.

[3] H. Bai, B. Hsieh, Z. Xiong et al., "Performance of radiologists in differentiating COVID-19 from viral pneumonia on chest CT," *Radiology*, vol. 296, no. 2, pp. E46–E54, 2020.

[4] Y. Fang, H. Zhang, J. Xie et al., "Sensitivity of chest CT for COVID-19: comparison to RT-PCR," *Radiology*, vol. 296, no. 2, pp. E115–E117, 2020.

[5] S. S. Hare, A. N. Tavare, and V. Dattani, "Validation of the British Society of Thoracic Imaging guidelines for COVID-19chest radiograph reporting," *Clinical Radiology*, vol. 75, no. 9,2020

[6] F. Pesapane, M. Codari, and F. Sardanelli, "Artificial intelli- gence in medical imaging: threat or opportunity? Radiologistsagain at the forefront of innovation in medicine," *European Radiology Experimental*, vol. 2, no. 1, 2018.

[7] L. Li, L. Qin, Z. Xu et al., "Using Artificial intelligence to DetectCOVID-19 and Community-acquired pneumonia Based on Pulmonary CT: Evaluation of the Diagnostic Accuracy," *Radi- ology*, vol. 296, no. 2, pp. E65–E71, 2020.

[8] M. Z. Alom, M. M. S. Rahman, M. S. Nasrin, T. M. Taha, and V. K. Asari, "COVID MTNet: COVID-19 detection with multi-task deep learning approaches," *arXiv preprint arXiv*, 2004, https://arxiv.org/abs/2004.03747.

[9] R. Hu, G. Ruan, S. Xiang, M. Huang, Q. Liang, and J. Li, "Auto-mated Diagnosis of COVID-19 Using Deep Learning and Data Augmentation on Chest CT," *medRxiv*, 2020, https://medRxiv .org/abs/2020.04.24.20078998.

[10] N. Ma, X. Zhang, H. Zheng, and J. Sun, "ShuffleNet V2: prac-tical guidelines for efficient CNN architecture design," *Com- puter Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings,*

[11] *Part XIV, SpringerInternational Publishing*, vol. 11218, pp. 122–138, 2018.

[12] O. Gozes, M. Frid-Adar, H. Greenspan et al., "Rapid AI devel-opment cycle for the coronavirus (COVID-19) pandemic: ini-tial results for automated detection & patient monitoring usingdeep lear ning CT imag e a na lysis, " *arXiv* , 2020, https://arXiv:2003.05037.

[13] H. Kassani, P. H. K. Sara, M. J. Wesolowski, K. A. Schneider, and R. Deters, "Automatic detection of coronavirus disease (COVID-19) in X-ray and CT images: a machine learning based approach," *arXiv*, vol. 10641, 2004.

[14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: inverted residuals and linear bottle- necks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, Salt Lake City, UT,USA, 2018.

[15] G. Huang, Z. Liu, L. Van Der Maaten, and K. Weinberger, "Densely connected convolutional networks," in *Proceedingsof The Ieee Conference On Computer Vision And Pattern Rec-ognition*, pp. 4700–4708, Honolulu, HI, USA, 2017.

[16] F. Chollet, "Xception: deep learning with depthwise separableconvolutions," in *2017 IEEE Conference on Computer Visionand Pattern Recognition (CVPR)*, pp. 1800–1807, Honolulu, HI, USA, 2017.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Rec- ognition (CVPR)*, pp. 2818– 2826, Las Vegas, NV, USA, 2016.

[18] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception- v4, inception-ResNet and the impact of residual connections on learning, AAAI'17," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 4278–4284, San Francisco, CA, USA, 2017.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning forimage recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, 2016.