DETECTION AND CLASSIFICATION OF COVID-19 USING DEEP LEARNING

A PROJECT REPORT

Submitted by

AJAY R (212919205003)
ARUNKUMAR A (212919205005)
SHERLIN A G (212919205043)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

ST. JOSEPH COLLEGE OF ENGINEERING, SRIPERUMBUDUR

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"DETECTION AND CLASSIFICATION OF COVID-19 USING DEEP LEARNING"** is the Bonafide work of **"AJAY.R (212919205003) ARUNKUMAR.A (212919205005)** and **SHERLIN.A.G (212919205043)"** who carried out the project work under my supervision.

SIGNATURE

Mr.S.MUTHUKUMRAN,M.E.,(Ph.D.)
HEAD OF THE DEPARTMENT
Assistant Professor
Department of Information Technology
St. Joseph College of Engineering
Sriperumbudur, Chennai-602 117.

SIGNATURE

Mr.S.KARTHI,M.E.,(Ph.D.)
SUPERVISOR
Assistant Professor
Department of Information Technology
St. Joseph College of Engineering
Sriperumbudur, Chennai-602 117.

Submitted for the Bachelor of Technology Degree Viva-Voice held at **St. Joseph college of Engineering** on ……

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are grateful and gifted in taking up this opportunity to thank the **Lord Almighty** for showering his unlimited blessing upon us.

We express our respect and thanks to **Rev.Fr.Dr.J.E.ARUL RAJ,** Founder and Chairman **MMI,DMI** and **Rev.Fr.S.GNANA SELVAM, DMI,** Managing Trustee for facilitating us to do our project successfully.

We thank our Administrator **Rev.Fr.L.SAVARIAPPAN, MMI**, for his kind and encouraging support.

We wish to eloquent our genuine thanks to our Principal **Dr.T.AHILAN,M.E.,Ph.D.,** for their support and guidance.

We express our thanks to our Head of the Department **Mr.S.MUTHUKUMARAN,M.E.,(Ph.D).,** for his scintillating and guidance for the development and completion of this project.

Words fail to express our gratitude to our project guide **Mr.S.KARTHI,M.E.,(Ph.D).,** Assistant Professor, who took special interest on our project and gave his constant support and guidance during all stages of this project.

Our special thanks to all Staff Members of our department who aided us ingeniously to bring our project as effective one.

Our special thanks to Non-Teaching Staffs for extending the Lab facilities. We thanks our family members and friends for their honorable supports.

ABSTRACT

Coronavirus disease 2019 (COVID-19) is an ongoing global pandemic that has spread rapidly since December 2019. Real-time reverse transcription polymerase chain reaction (RT-PCR) and chest computed tomography (CT) imaging both play an important role in COVID-19 diagnosis.

Chest CT imaging offers the benefits of quick reporting, a low cost, and high sensitivity for the detection of pulmonary infection. Originally CT scan image is considered as input and it will be pre-processed using adaptive bilateral filter for eliminating noises exist in input image.

Meanwhile, augmentation of data will be executed based on rotation, shifting, flipping and zooming methods. After that, lung lobe segmentation will be performed using U-Net, which segments the lung lobe region from augmented image. Besides, lung lesion segmentation will be done using kernel-based Bayesian fuzzy clustering model.

Finally, classification of Covid-19 will be executed based on DenseNet model, which affords the outcome as positive and negative. In addition, the DenseNet model will be trained by proposed optimization algorithm, named Residual Neural Network (ResNet).

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The COVID-19 pandemic has had a significant impact on global health and economy, with over 200 million confirmed cases and over 4 million deaths worldwide. To tackle this pandemic, early and accurate detection of COVID-19 cases is crucial. Deep learning has emerged as a promising technique for automated diagnosis of COVID-19, leveraging large amounts of data to train neural networks that can detect patterns and features specific to the disease.

With the availability of large-scale datasets of chest X-rays and CT scans, deep learning models have been trained to detect COVID-19 with high accuracy. The use of deep learning in COVID-19 diagnosis has the potential to reduce the workload of healthcare workers, increase the speed of diagnosis, and improve the accuracy of detection, enabling timely and effective treatment of patients.

In this context, the detection of COVID-19 using deep learning has gained significant attention in the research community, with various studies proposing novel models and techniques for automated diagnosis.

In 2019, Chinese health authorities got to know a new unknown origin of pneumonia. The name SARS-CoV-2 (severe acute respiratory syndrome) means the lungs and respiratory tract are highly affected, leading to the cause of novel coronavirus. The disease first emerged in Wuhan, China in December 2019 and has since become a global pandemic, affecting millions of peoples around the world. Severe corona symptoms are difficulty breathing, chest pain, and several other common and mild symptoms like cold, headache, fever. Some people may also experience body aches, sore throat, loss of taste or smell, and diarrhea.

At the beginning of the coronavirus, an RT-PCR test (Reverse Transcription-polymerase chain reaction) was used to predict the virus. This method is inaccurate because it gives false positives, which means people who are not infected by the virus are told that they have been tested positive, and false negatives, which means people who have the virus, are classified as harmful by our algorithm.

This way of wrongly classifying is alarmingly shortcoming as it would allow many infected people to go home and spread the virus. RTPCR results lead to an increase of spreading the virus as testing consumes more time, whereas now there are computer- based models to predict the values and give accurate results. For the RT-PCR test, all the collected samples are delivered to testing in less than two days; else, there is a considerable possibility of false results. The accuracy decreases with an increase in time. Hence this method is not reliable.

As a non-invasive imaging approach, CT-Scan can depict specific characteristics manifestations in the lung associated with COVID-19. Therefore, CT-Scan could serve as the most accurate way for early diagnosis of COVID-19.

The use of deep learning in COVID-19 diagnosis has the potential to reduce the workload of healthcare workers, increase the speed of diagnosis, and improve the accuracy of detection, enabling timely and effective treatment of patients. In this context, the detection of COVID-19 using deep learning has gained significant attention in the research community, with various studies proposing novel models and techniques for automated diagnosis.

We used the VGG-19 with batch normalization as our inspiration model. We shall be constructing our own neural network model and shall be training the model with our training data.

1.1 Artificial Intelligence And Deep Learning

Definition: Artificial Intelligence (AI) is the study and creation of computer systems that can perceive, reason and act.

The primary aim of Al is to produce intelligent machines. The intelligence should be exhibited by thinking, making decisions, solving problems, more importantly by learning. Al is an interdisciplinary field that requires knowledge in computer science, linguistics, psychology, biology, philosophy and so on for serious research.

According to the father of Artificial Intelligence, John McCarthy, it is the science and engineering of making intelligent machines, especially intelligent computer programs. It is a way of Making a Computer, a Computer-Controlled Robot, or a Software Think Intelligently in the similar manner the intelligent humans think.

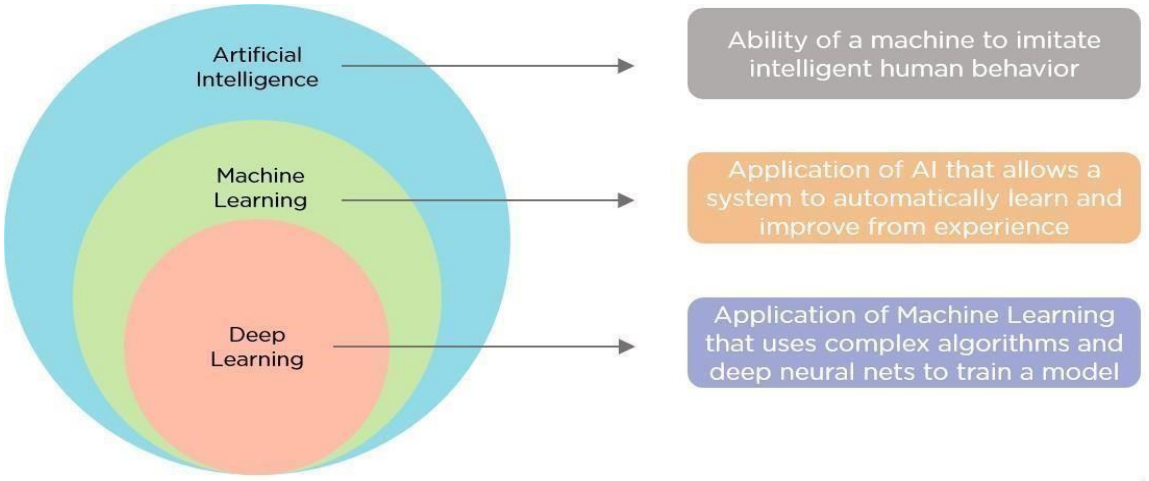


Figure: 1.1 Artificial Intelligence

Artificial intelligence (AI), the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings.

The term is frequently applied to the project of developing systems endowed the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. Since the development of the digital computer in the 1940s, it has been demonstrated that computers can be programmed to carry out very complex tasks—as, for example, discovering proofs for mathematical theorems or playing chess—with great proficiency.

Still, despite continuing advances in computer processing speed and memory capacity, there are as yet no programs that can match human flexibility over wider domains or in tasks requiring much everyday knowledge. On the other hand, some programs have attained the performance levels of human experts and professionals in performing certain specific tasks, so that artificial intelligence in this limited sense.

Definition: 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.

1.1.1 Introduction To 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. It is a form of machine learning, can be used to help detect fraud or money laundering, among other functions.

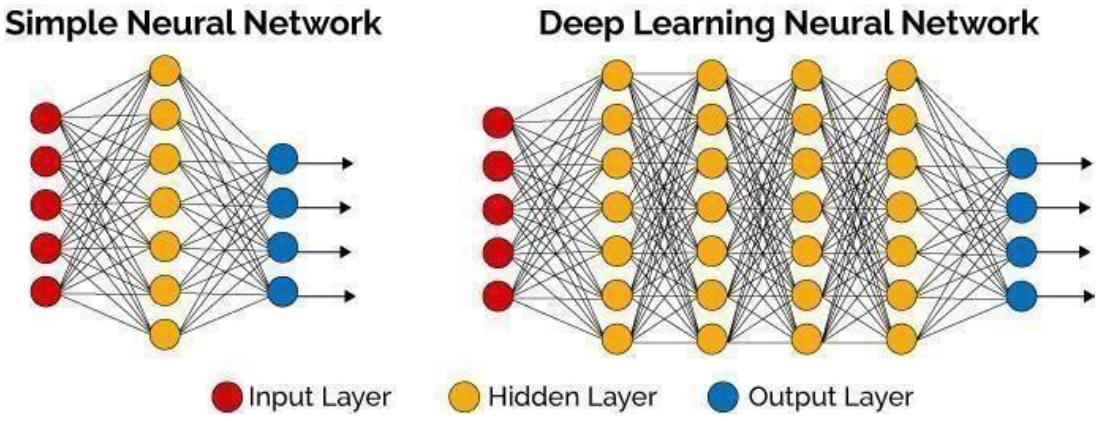


Figure: 1.1.1 Neural Network Test

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 unravelling this wealth of information and are increasingly adapting to AI systems for automated support.

The ongoing pandemic of Coronavirus Disease (COVID19) is causing a serious global crisis at this moment. At present, the tests for detecting the presence of COVID19 are performed based on reverse transcription-polymerase chain reaction (RT-PCR), which usually takes 4–6 hours to generate results and there are no sufficient availability of testing kits which is a major problem .To deal with these issues, radiology imaging based approaches have been proposed application of Artificial Intelligence (AI) and Deep Learning (DL) based approaches for efficient detection of the disease from chest CT images.

Relevancy: Data to be trained should be directly relevant to the problem. It must resemble as real-world data to process. The training data should resemble the real- world data that they will classify in production.

Proper Classification: If we want to build a deep-learning solution that classifies data, then they need to have a labeled dataset. That is, needs to apply labels to the raw data: this a covid images and non- covid images. With time and tuning, this training dataset can teach a neural network to classify new images it has not seen before.

Formatting: All data needs to be vectorized, and the vectors should be the same length when they enter the neural net. To get vectors of the same length, it's helpful to have, say, images of the same size. So sometimes we need to resize the images. This is called data pre-processing.

Accessibility: The data needs to be stored in a place that's easy to work with. A local file system, for example. If the data is stored in many different databases that are unconnected, we have to build data pipelines.

1.1.2 How We Split Data In Deep- Learning

Training Data: The sample of data used to fit the model. The part of data we use to train our model. This is the data which your model actually sees and learn from.

Validation Data: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The part of data which is used to do a frequent evaluation of model, fit on training dataset along with improving involved hyperparameters (initially set parameters before the model begins learning). This data plays it's part when the model is actually training.

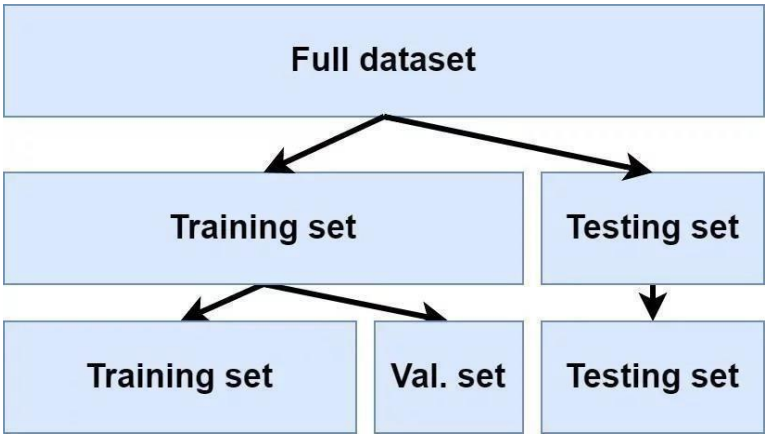
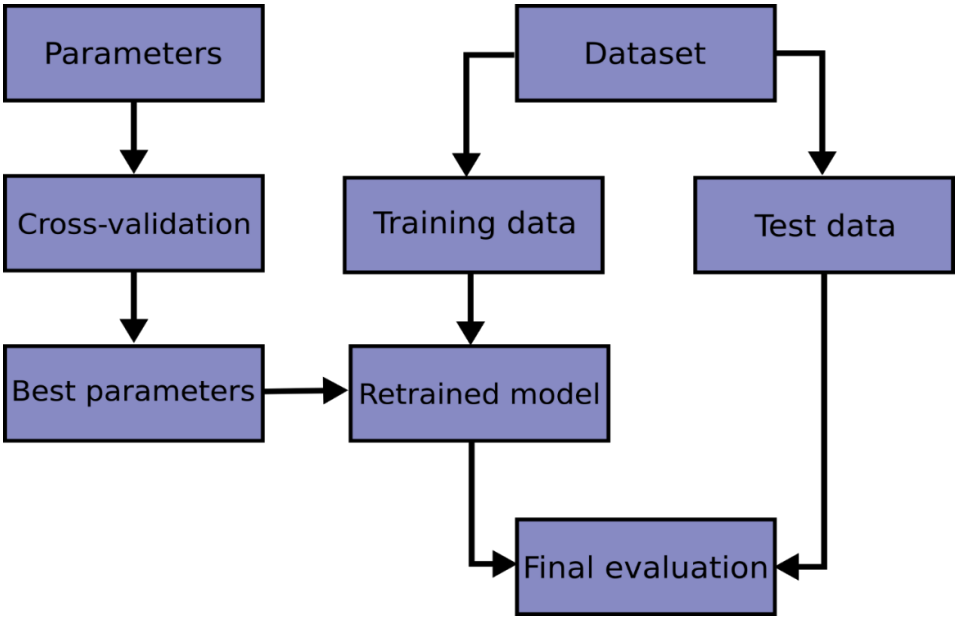


Figure: 1.1.2 Train split

Testing Data: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. Once our model is completely trained, testing data provides the unbiased evaluation. When we feed in the inputs of Testing data, our model will predict some values(without seeing actual output). After prediction, we evaluate our model by comparing it with actual output present in the testing data.

Parameters: A parameter is a function argument that could have one of a range of values. In machine learning, the specific model you are using is the function and requires parameters in order to make a prediction on new data. Whether a model has a fixed or variable number of parameters determines whether it may be referred to as “parametric” or “nonparametric“.



1.1.3 Framework For Approaching The Machine Learning

1. **Problem Statement:** Rephrasing our problem as machine learning problem. The four major types of machine learning are supervised learning, unsupervised learning, transfer learning and reinforcement learning (there’s semi-supervised as well but I’ve left it out for brevity). The three most used in business applications are supervised learning, unsupervised learning, transfer learning.

2. **Evaluation:** A 95% accurate model may sound pretty good for predicting who’s at fault in an insurance claim. But for predicting heart disease, you’ll likely want better results. Other things you should take into

consideration for classification problems. False negatives, false positives, True negatives, True positives, Precision, Recall, ROC.

3. **Features:** What features does our data have and which model can be used to build our model. The three main types of features are categorical, continuous (or numerical) and derived.

4. **Modelling:** Choosing our model and how to improve it and comparing our model with other models. Modelling breaks into three parts, choosing a model, improving a model, comparing it with others. When choosing a model, you'll want to take into consideration, interpretability and ease to debug, amount of data, training and prediction limitations.

5. **Experimentation:** Experimentation should be conducted on different portions of data i.e., testing dataset, training dataset, validation dataset.

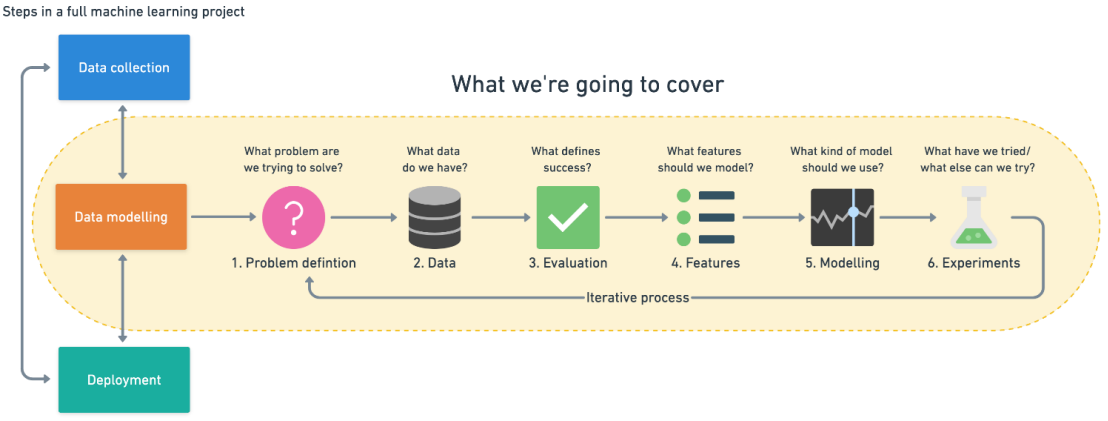


Fig 1.1.3 Machine Learning process

1.2 PYTHON

Python is a popular programming language. Python is a robust programming language and provides an easy usage of the code lines, maintenance can be handled in a great way, and debugging can be done easily too. It was created by Guido van Rossum, and released in 1991. It is used for:

- Web development (server-side),
- Software development,
- Mathematics,
- System scripting.

Python does the things as follow:

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Python benefits:

- Back- end and front- end development
- Cross platform language
- Open source
- Strong community base
- Fewer and simple lines of code

Advantages of python:

The Python language has diversified application in the software development companies such as in gaming, web frameworks and applications, language development, prototyping, graphic design applications, etc. This provides the language a higher plethora over other programming languages used in the industry. Some of its advantages are-

- Extensive support libraries: It provides large standard libraries that include the areas like string operations, Internet, web service tools operating system.
- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

1.2.1 Machine Learning in Python

Machine learning is learning based on experience. As an example, it is like a person who learns to play chess through observation as others play. In this way, computers can be programmed through the provision of information in which they are trained, acquiring the ability to identify elements or their characteristics with high probability. There are various stages of machine learning:

- Collect and prepare data
- Make sense of data
- Use data to answer questions

- Algorithm development
- Checking algorithm generated
- Create predictive applications
- Keep building

Machine learning algorithms divided into two groups:

● Supervised learning:

Supervised learning, as the name indicates, has the presence of a supervisor as a teacher. Basically, supervised learning is when we teach or train the machine using data that is well labeled. Which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples(data) so that the supervised learning algorithm analyses the training data (set of training examples) and produces a correct outcome from labeled data.

Supervised learning classified into two categories of algorithms:

- **Classification**: A classification problem is when the output variable is a category, such as “Red” or “blue” or “disease” and “no disease”.
- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

● Unsupervised learning:

Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data.

Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore, the machine is restricted to find the hidden structure in unlabeled data by itself.

Unsupervised learning is classified into two categories of algorithms:

- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

1.2.1.1 OS Module

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.
- To change the current working directory(CWD) os.chdir() method is used. This method changes the CWD to a specified path. It only takes a single argument as a new directory path.
- os.mkdir() method in Python is used to create a directory named path with the specified numeric mode. This method raise FileExistsError if the directory to be created already exists.
- os.makedirs() method in Python is used to create a directory recursively. That means while making leaf directory if any intermediate-level directory is missing, os.makedirs() method will create them all.

1.2.1.2 Open CV

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. cv2.imread() method 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.

Python is a general purpose programming language started by Guido van Rossum that became very popular very quickly, mainly because of its simplicity and code readability.

Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to9 code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

Syntax: cv2.imread(path, flag) path: A string
representing the path of the image to be read.

flag: It specifies the way in which image should be read.
It's default value is cv2.IMREAD_COLOR

Return Value: This method returns an image that is loaded from the specified file.

1.2.1.3 GC

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

This module provides an interface to the optional garbage collector. It provides the ability to disable the collector, tune the collection frequency, and set debugging options. It also provides access to unreachable objects that the collector found but cannot free. Since the collector supplements the reference counting already used in Python, you can disable the collector if you are sure your program does not create reference cycles.

The gc module provides the following functions:

- gc.**enable**() - Enable automatic garbage collection.
- gc.**disable**() - Disable automatic garbage collection.
- gc.**isenabled**() - Return True if automatic collection is enabled.
- gc.**get_debug**() - Return the debugging flags currently set.

1.2.1.4 Numpy

NumPy is the fundamental package needed for scientific computing with Python.

This package contains:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Basic linear algebra functions
- Basic Fourier transforms
- Sophisticated random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi- dimensional container of generic data. Arbitrary data types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. NumPy is a successor for two earlier scientific Python libraries: Numeric and Numarray.

1.2.1.5 Random

This module implements pseudo-random number generators for various distributions. For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, a function to generate a random permutation of a list in- place, and a function for random sampling without replacement. The functions supplied by this module are actually bound methods of a hidden instance of the “random.Random “class. You can instantiate your own instances of Random to get generators that don’t share state.

Class Random can also be subclassed if you want to use a different basic generator of your own devising: in that case, override the random(), seed(), getstate(), and setstate() methods. Optionally, a new generator can supply a getrandbits() method — this allows randrange() to produce selections over an arbitrarily large range.

The random module also provides the SystemRandom class which uses the system function os.urandom() to generate random numbers from sources provided by the operating system.

1.2.1.6 Matplot Lib

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.

1.2.1.7 Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types.

In general, you could say that the Pandas Data Frame consists of three main components: the data, the index, and the columns.

1.3 GOOGLE COLAB

Jupyter Notebook Environment: Google Colab provides a Jupyter Notebook interface where you can write and execute Python code in cells. It supports features like syntax highlighting, code completion, and markdown support for documenting your code.

Cloud-based Execution: The code runs on Google's servers, leveraging their computational resources. This allows you to execute code and handle large datasets without worrying about hardware limitations on your local machine.

Free GPU and TPU: Google Colab offers free access to GPU (Graphical Processing Unit) and TPU (Tensor Processing Unit) resources, which are useful for accelerating computations in machine learning and deep learning tasks.

.

Import API and get credentials: This section demonstrates how to import the Earth Engine Python

API and authenticate access. The Earth Engine API is included by default in Google Colaboratory so requires only importing and authenticating. These

steps must be completed for each new Colab session or if you restart your Colab kernel or if your Colab virtual machine is recycled due to inactivity.

Import the API: Run the cell to import the API into your session.

Authenticate and initialize: Run the ee Authenticate function to authenticate your access to Earth
Engine servers and ee.Initialize to initialize it. Add a code cell, enter the following lines, and run the cell.

Test the API: Test the API by printing the elevation of Mount Everest.
Note that before using the API you must initialize it. Run the following Python script in a new cell.

1.4 KERAS

Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow.

It was developed to make implementing deep learning models as fast and easy as possible for research and development.

Keras was developed and maintained by François Chollet, a Google engineer using four guiding principles:

- **Modularity**: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.
- **Minimalism**: The library provides just enough to achieve an outcome, no frills and maximizing readability.

- **Extensibility**: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.
- **Python**: No separate model files with custom file formats. Everything is native Python.

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.

Benefits Of Keras:

Keras is highly powerful and dynamic framework and comes up with the following advantages −

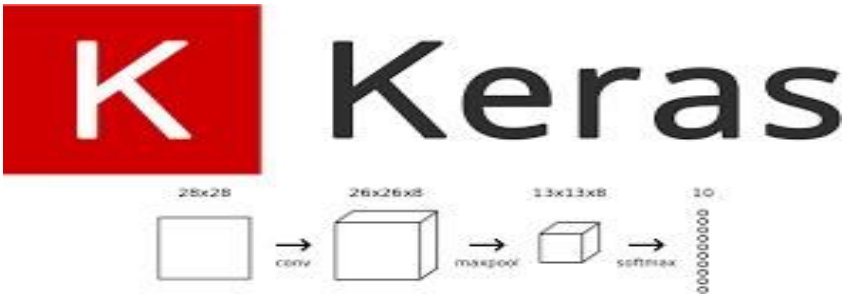
- Larger community support.
- Easy to test.
- Keras neural networks are written in Python which makes things simpler.
- Keras supports both convolution and recurrent networks.
- Deep learning models are discrete components, so that, you can combine into many ways.

1.4.1 Keras layers

Convolution Layer: This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot

product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.



Pooling Layer:

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations.

Fully Connected layer:

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

Dropout:

When all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model

works so well on the training data causing a negative impact in the model's performance when used on a new data.

To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

Activation Functions:

One of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network. It adds non-linearity to the network.

There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions has a specific usage. For a binary classification CNN model, sigmoid and soft max functions are preferred for a multi-class classification, generally soft max us used.

1.5 SkILEARN METRICS

The **skIlearn.metrics** module implements functions assessing prediction error for specific purposes. These metrics are detailed in sections on Classification metrics, Multilabel ranking metrics, Regression metrics and Clustering metrics.

- **Multilabel metrics:** In multilabel learning, each sample can have any number of ground truth labels associated with it. The goal is to give high scores and better rank to the ground truth labels. The **coverage_error** function computes the average number of labels that have to be included in the final prediction such that all true labels are predicted. This is useful if you want to know how many top-scored-labels you have to predict in average without missing any true one. The best value of this metrics is thus the average number of true labels. The **label_ranking_average_precision_score** function implements label ranking average precision (LRAP). This metric is linked to the **average_precision_score** function, but is based on the notion of label ranking instead of precision and recall. Label ranking average precision (LRAP) averages over the samples the answer to the following question: for each ground truth label, what fraction of higher-ranked labels were true labels? This performance measure will be higher if you are able to give better rank to the labels associated with each sample. The obtained score is always strictly greater than 0, and the best value is 1. If there is exactly one relevant label per sample, label ranking average precision is equivalent to the mean reciprocal rank.

Regression metrics: The **sklearn.metrics** module implements several loss, score, and utility functions to measure regression performance. Some of those have been enhanced to handle the multioutput case:

mean_squared_error,**mean_absolute_error**, **explained_variance_score** and **r2_score**.These functions have an multioutput keyword argument which specifies the way the scores or losses for each individual target should be averaged. The default is 'uniform_average', which specifies a uniformly weighted mean over outputs. If an array of shape (outputs,) is passed, then its entries are interpreted as weights and an according weighted average is returned. If multioutput is 'raw_values' is specified, then all unaltered individual scores or losses will be returned in an array of shape (n_outputs,). The r2_score and explained_variance_score accept an additional value 'variance_weighted' for the multioutput parameter. This option leads to a weighting of each individual score by the variance of the corresponding target variable. This setting quantifies the globally captured unscaled variance. If the target variables are of different scale, then this score puts more importance on wel explaining the higher variance variables. multioutput='variance_weighted' is the default value for r2_score for backward compatibility. This will be changed to uniform_average in the future.

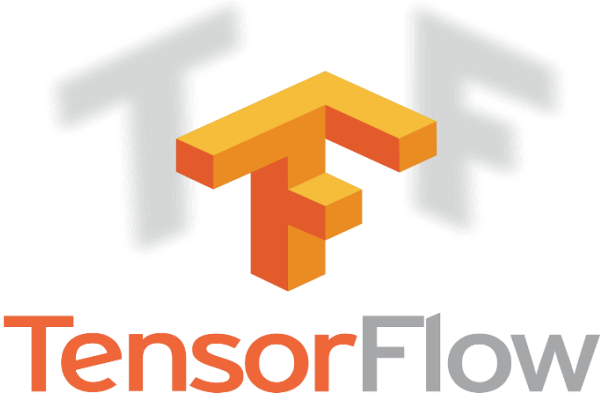
Clustering metrics: The sklearn.metrics module implements several loss, score, and utility functions. For more information see the Clustering performance evaluation section for instance clustering, and Biclustering evaluation for biclustering. When doing supervised learning, a simple sanity check consists of comparing one's estimator against simple rules of thumb.

1.6 TENSORFLOW

The TensorFlow tutorials are written as Jupyter notebooks and run directly in Google Colab—a hosted notebook environment that requires no setup. Click the *Run in Google Colab* button. Explore libraries to build advanced models or methods using TensorFlow, and access domain-specific application packages that extend TensorFlow.

TensorFlow Hub is a repository of trained machine learning models ready for fine- tuning and deployable anywhere. Reuse trained models like BERT and Faster R-CNN with just a few lines of code. **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**.

TensorFlow is an open-source library for fast numerical computing.



It was created and is maintained by Google and released under the Apache 2.0 open source license. The API is nominally for the Python programming language, although there is access to the underlying C++ API.

Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development

and in production systems, not least RankBrain in Google search and the fun DeepDream project.

It can run on single CPU systems, GPUs as well as mobile devices and large scale distributed systems of hundreds of machines.

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of the-art in ML and developers easily build and deploy ML powered applications.

Easy model building

Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging.

TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy.

If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML training tasks, use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition.

Robust ML production anywhere

Easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language you use.

Powerful experimentation for research

A simple and flexible architecture to take new ideas from concept to code, to state- of-the-art models, and to publication faster. Build and train state-of-the-art models without sacrificing speed or performance. TensorFlow gives you the flexibility and control with features like the Keras Functional API and Model Subclassing API for creation of complex topologies. For easy prototyping and fast debugging, use eager execution.

TensorFlow also supports an ecosystem of powerful add-on libraries and models to experiment with, including Ragged Tensors, TensorFlow Probability, Tensor2Tensor and BERT.

1.6.1 Applications

1.6.1.1 Voice and speech recognition

The real challenge put before programmers was that a mere hearing of the words will not be enough. Since, words change meaning with context, a clear understanding of what the word represents with respect to the context is necessary. This is where deep learning plays a significant role. With the help of Artificial Neural Networks or ANNs, such an act has been made possible by performing word recognition, phoneme classification, etc.

1.6.1.2 Image recognition

Apps that use the image recognition technology are probably the ones that popularized deep learning among the masses. The technology was developed with the intention to train and develop computers to see, identify, and analyze the world like how a human would. Today, a number of applications finds these useful — the artificial intelligence enabled camera on your mobile phone, the social networking sites you visit, your telecom operators, to name a few.

1.6.1.3 Time series

The most common application of Time Series is in Recommendations. If you are someone using Facebook, YouTube, Netflix, or any other entertainment platform, then you may be familiar with this concept. For those who do not know, it is a list of videos or articles that the service provider believes suits you the best. TensorFlow Time Services algorithms are what they use to derive meaningful statistics from your history.

1.7 CONVOLUTIONAL NEURAL NETWORK

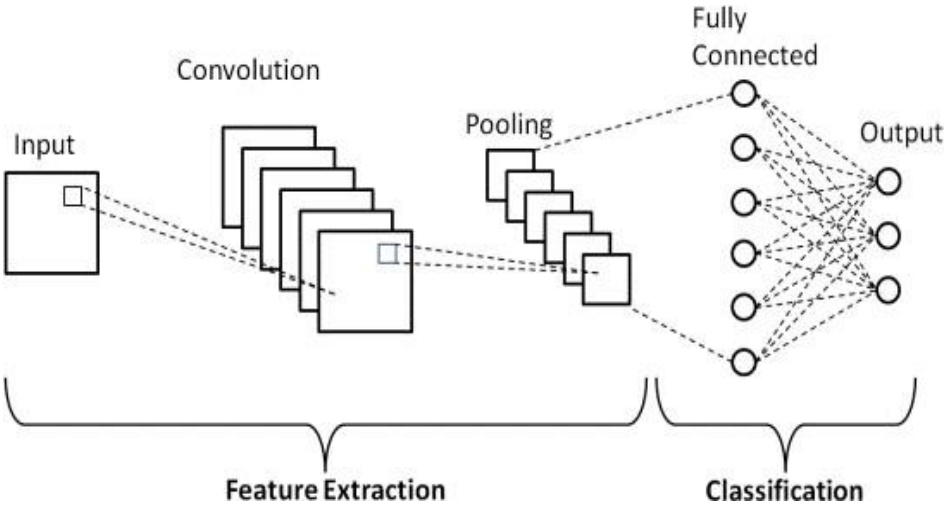
Convolutional Neural Networks (CNN's) are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analysing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing.

The term 'Convolution" in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other. In simple terms, two images which can be represented as matrices are multiplied to give an output that is used to extract features from the image.

There are two main parts to CNN architecture:

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction

• A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.



There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. When these layers are stacked, CNN architecture will be formed.

Convolution layer

A convolution layer is a fundamental component of the CNN architecture that performs feature extraction, which typically consists of a combination of linear and nonlinear operations, i.e., convolution operation and activation function.

The padding argument is set to 'same' to ensure that the output has the same spatial dimensions as the input. The layer uses the rectified linear activation function (relu) to introduce non-linearity.

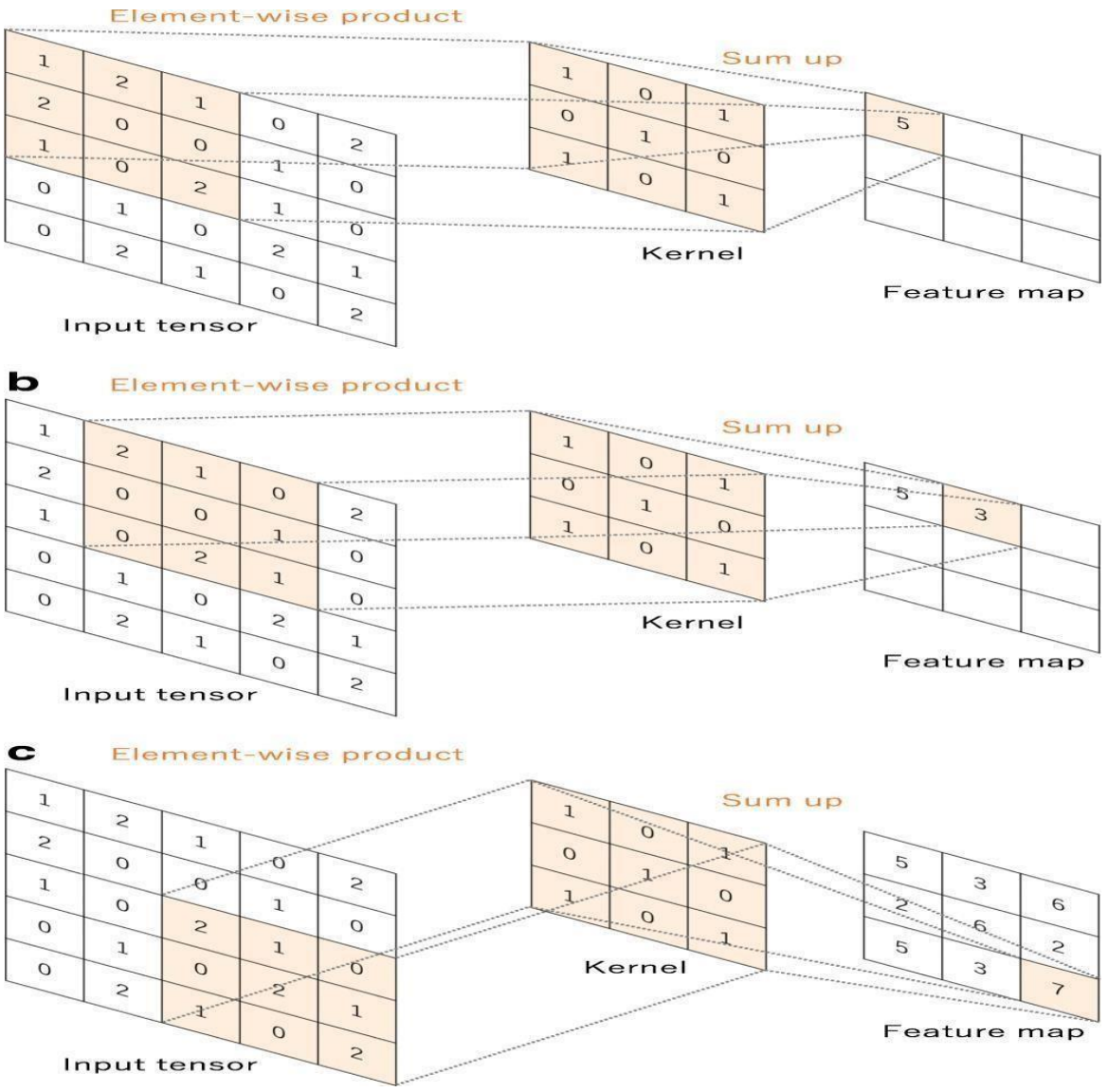
This layer can be used in a deep learning model for COVID-19 detection, along with other layers such as pooling layers, normalization layers, and fully connected layers, to form a complete network architecture.

Convolution:

Convolution is a specialized type of linear operation used for feature extraction, where a small array of numbers, called a kernel, is applied across the input, which is an array of numbers, called a tensor. An element-wise product between each element of the kernel and the input tensor is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor, called a feature map.

This procedure is repeated applying multiple kernels to form an arbitrary number of feature maps, which represent different characteristics of the input tensors; different kernels can, thus, be considered as different feature extractors.Two key hyperparameters that define the convolution operation are size and number of kernels. The former is typically 3×3, but sometimes 5×5 or 7×7. Thelatter is arbitrary, and determines the depth of output feature maps.

We pass this input tensor along with the desired parameters for the convolutional layer to the conv_layer function to define the conv tensor, which can then be used as an input to subsequent layers in the deep learning model.



In the convolution operation described above does not allow the center of each kernel to overlap the outermost element of the input tensor, and reduces the height and width of the output feature map compared to the input tensor. Padding, typically zero padding, is a technique to address this issue, where rows and columns of zeros are added on each side of the input tensor, so as to fit the center of a kernel on the outermost element and keep the same in-plane dimension through the convolution operation (Fig. 4). Modern CNN architectures usually employ zero padding to retain in-plane dimensions in

order to apply more layers. Without zero padding, each successive feature map would get smaller after the convolution operation.

The distance between two successive kernel positions is called a stride, which also defines the convolution operation. The common choice of a stride is 1; however, a stride larger than 1 is sometimes used in order to achieve down sampling of the feature maps. An alternative technique to perform down sampling is a pooling operation, as described below.

The key feature of a convolution operation is weight sharing: kernels are shared across all the image positions. Weight sharing creates the following characteristics of convolution operations:

(1) letting the local feature patterns extracted by kernels translation b invariant as kernels travel across all the image positions and detect learned local patterns

(2) learning spatial hierarchies of feature patterns by down sampling in conjunction with a pooling operation, resulting in capturing an increasingly larger field of view, and

(3) increasing model efficiency by reducing the number of parameters to learn in comparison with fully connected neural networks.

As described later, the process of training a CNN model with regard to the convolution layer is to identify the kernels that work best for a given task based on a given training dataset.

ACTIVATION FUNCTION:

It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).

Types of Activation Functions

- **Sigmoid Function:**

In an ANN, the sigmoid function is a non-linear AF used primarily in feedforward neural networks. It is a differentiable real function, defined for real input values, and containing positive derivatives everywhere with a specific degree of smoothness. The sigmoid function appears in the output layer of the deep learning models and is used for predicting probability-based outputs. The sigmoid function is represented as:

$$f(x) = \left(\frac{1}{(1 + exp^{-x})} \right) - (1.4)$$

Generally, the derivatives of the sigmoid function are applied to learning algorithms. The graph of the sigmoid function is 'S' shaped.

Some of the major drawbacks of the sigmoid function include gradient saturation, slow convergence, sharp damp gradients during backpropagation from within deeper hidden layers to the input layers, and non-zero centered output that causes the gradient updates to propagate in varying directions.

- **Hyperbolic Tangent Function(Tanh):**

The hyperbolic tangent function, a.k.a., the tanh function, is another type of AF. It is a smoother, zero-centered function having a range between -1 to 1. As a result, the output of the tanh function is represented by:

$$f(x_i) = \frac{exp\ (x_i)}{\sum_j exp\ (x_j)} \quad -(1.12)$$

The tanh function is much more extensively used than the sigmoid function since it delivers better training performance for multilayer neural networks. The biggest advantage of the tanh function is that it produces a zero-centered output, thereby supporting the backpropagation process. The tanh function has been mostly used in recurrent neural networks for natural language processing and speech recognition tasks.

However, the tanh function, too, has a limitation – just like the sigmoid function, it cannot solve the vanishing gradient problem. Also, the tanh function can only attain a gradient of 1 when the input value is 0 (x is zero). As a result, the function can produce some dead neurons during the computation process.

- **Softmax Function:**

The softmax function is another type of AF used in neural networks to compute probability distribution from a vector of real numbers.
This function generates an output that ranges between values 0 and 1 and with the sum of the probabilities being equal to 1. The softmax function is represented as follows:

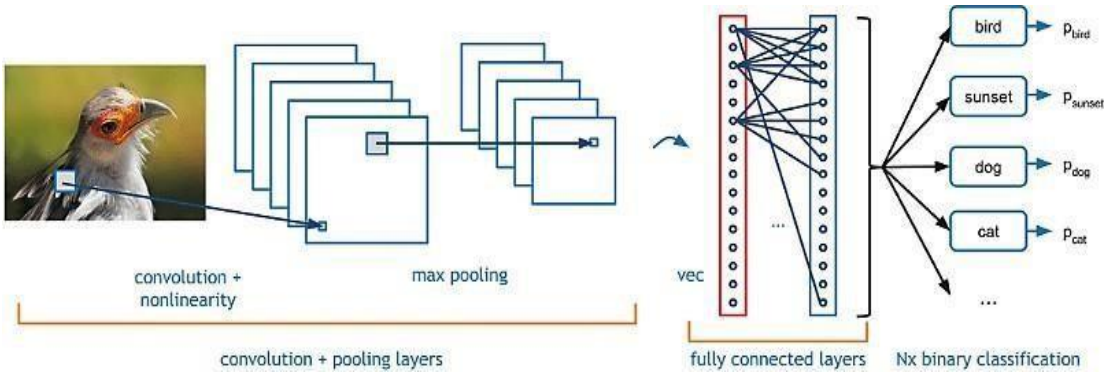
$$f(x_i) = \frac{exp\ (x_i)}{\sum_j exp\ (x_j)} \quad -(1.12)$$

This function is mainly used in multi-class models where it returns probabilities of each class, with the target class having the highest probability. It appears in almost all the output layers of the DL architecture where they are used. The primary difference between the sigmoid and softmax AF is that while the former is used in binary classification, the latter is used for multivariate classification.

DROP OUT LAYER: Dropout works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase. Dropout is implemented per-layer in a neural network. It can be used with most types of layers, such as dense fully connected layers, convolutional layers, and recurrent layers such as the long short-term memory network layer.Dropout may be implemented on any or all hidden layers in the network as well as the visible or input layer. It is not used on the output layer.

Image Classification using CNN:

Convolutional Neural Networks come under the subdomain of Machine Learning which is Deep Learning. Algorithms under Deep Learning process information the same way the human brain does, but obviously on a very small scale, since our brain is too complex (our brain has around 86 billion neurons). Image classification involves the extraction of features from the image to observe some patterns in the dataset. Using an ANN for the purpose of image classification would end up being very costly in terms of computation since the trainable parameters become extremely large.



For example, if we have a 50 X 50 image of a cat, and we want to train our traditional

ANN on that image to classify it into a dog or a cat the trainable parameters become – (50*50) * 100 image pixels multiplied by hidden layer + 100 bias + 2 * 100 output neurons + 2 bias = 2,50,302.

We use filters when using CNNs. Filters exist of many different types according to their purpose. Filters help us exploit the spatial locality of a particular image by enforcing a local connectivity pattern between neurons. Convolution basically means a pointwise multiplication of two functions to produce a third function. Here one function is our image pixels matrix and another is our filter.

We slide the filter over the image and get the dot product of the two matrices. The resulting matrix is called an “Activation Map” or “Feature Map”.

Convolutional Neural Networks have several types of layers:

- **Convolutional layer**: a “filter” passes over the image, scanning a few pixels at a time and creating a feature map that predicts the class to which each feature belongs.

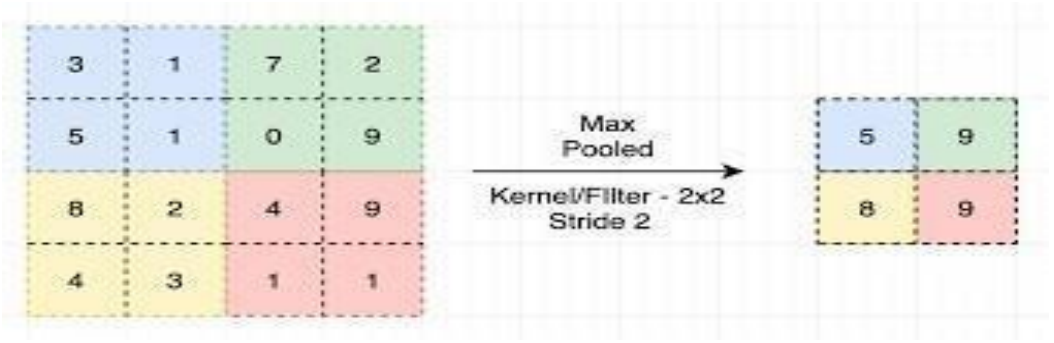
• **Pooling layer (down sampling)**:reduces the amount of information in each feature obtained in the convolutional layer while maintaining the most important information. • **Fully connected input layer (flatten)**:takes the output of the previous layers,
"flattens" them and turns them into a single vector that can be an input for the next stage.

• **The first fully connected layer**:takes the inputs from the feature analysis and applies weights to predict the correct label.

• **Fully connected output layer**:gives the final probabilities for each label.

1.7.1 Pooling

Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarizes the features present in a region of the feature map generated by a convolution layer. So, further operations are performed on summarized features instead of precisely positioned features generated by the convolution layer. This makes the model more robust to variations in the position of the features in the input image.



1.7.2 Fully Connected

The objective of a fully connected layer is to take the results of the convolution/pooling process and use them to classify the image into a label.

The output of convolution/pooling is flattened into a single vector of values, each representing a probability that a certain feature belongs to a label.

The fully connected part of the CNN network goes through its own backpropagation process to determine the most accurate weights. Each neuron receives weights that prioritize the most appropriate label. Finally, the neurons "vote" on each of the labels, and the winner of that vote is the classification decision.

1.7.3 Feed Forward

A feedforward neural network is a biologically inspired classification algorithm. It consists of a (possibly large) number of simple neuron-like processing units, organized in layers. Every unit in a layer is connected with all the units in the previous layer.

These connections are not all equal: each connection may have a different strength or weight. The weights on these connections encode the knowledge of a network. Often the units in a neural network are also called nodes.

Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called feedforward neural networks.

The feedforward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction— forward—from the input nodes, through the hidden.

1.7.4 Back Propagation

Backpropagation is an algorithm commonly used to train neural networks. When the neural network is initialized, weights are set for its individual elements, called neurons.

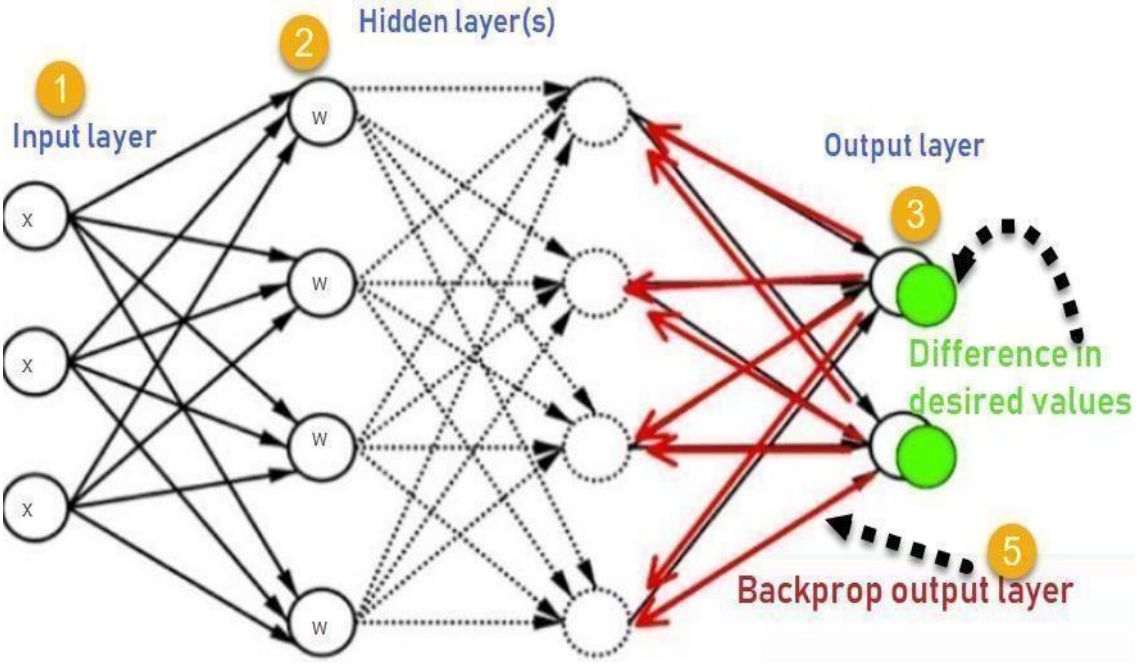
Inputs are loaded, they are passed through the network of neurons, and the network provides an output for each one, given the initial weights. Backpropagation helps to adjust the weights of the neurons so that the result comes closer and closer to the known true result.

Backpropagation, short for "backward propagation of errors," is an algorithm for supervised learning of artificial neural networks using gradient descent. Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural network's weights. It is a generalization of the delta rule for perceptron’s to multilayer feedforward neural networks.

The "backwards" part of the name stems from the fact that calculation of the gradient proceeds backwards through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last. Partial computations of the gradient from one layer are reused in the computation of the gradient for the previous layer. This backwards flow of the error information allows for efficient computation of the gradient at each layer versus the naive approach of calculating the gradient of each layer separately.

Backpropagation's popularity has experienced a recent resurgence given the widespread adoption of deep neural networks for image recognition and

speech recognition. It is considered an efficient algorithm, and modern implementations take advantage of specialized GPUs to further improve performance.



1.7.5 Entropy Loss

Entropy is the number of bits required to transmit a randomly selected event from a probability distribution. A skewed distribution has a low entropy, whereas a distribution where events have equal probability has a larger entropy.

A skewed probability distribution has less “surprise” and in turn a low entropy because likely events dominate. Balanced distribution are more surprising and turn have higher entropy because events are equally likely.

Cross-entropy is commonly used in machine learning as a loss function. Cross-entropy is a measure from the field of information theory, building upon entropy and generally calculating the difference between two probability distributions. It is closely related to but is different from KL divergence that calculates the relative entropy between two probability distributions, whereas cross-entropy can be thought to calculate the total entropy between the distributions.

Cross-entropy is also related to and often confused with logistic loss, called log loss. Although the two measures are derived from a different source, when used as loss functions for classification models, both measures calculate the same quantity and can be used interchangeably. Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross- entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

1.7.6 Optimizer

An optimizer is a method or algorithm to update the various parameters that can reduce the loss in much less effort. Let's look at some popular Deep learning optimizers that deliver acceptable results.

Gradient Descent (GD)

This is the most basic optimizer that directly uses the derivative of the loss function and learning rate to reduce the loss and achieve the minima. This approach is also adopted in backpropagation in neural networks where the

updated parameters are shared between different layers depending upon when the minimum loss is achieved. It is easy to implement and interpret the results, but it has various issues.

The weights are updated when the whole dataset gradient is calculated, which slows down the process. It also requires a large amount of memory to store this temporary data, making it a resource-hungry process. Though the idea behind this algorithm is well suited, it needs to be tweaked.

1.8 PROBLEM STATEMENT

Detection of COVID-19 Using Deep Learning CT Image Classification using convolutional neural networks: Improving Testing with Deep Learning and Computer Vision. In this project we shall discuss, how testing is done for the covid-19 and how deep learning tools will be useful for medical imaging can help us in improving the testing quality of COVID-19.

We aim to research how AI can be used to tackle the coronavirus and implement how Deep Learning is being used to develop better ways for testing COVID 19. We will be building a classifier that contains the information of COVID as well as non-COVID patients which can differentiate the pneumonia of patients based on which we declare the results of the patients.

We propose to build a deep learning neural network to detect COVID- 19 using chest CT scans of community- acquired pneumonia and other non- pneumonia lung disease will be excluded for future works.

CHAPTER 2
LITERATURE SURVEY

2.1 Paper-1: Deep Learning System to Screen Coronavirus Disease 2019 Pneumonia

Objectives: Aimed to establish an early screening model to distinguish COVID-19 pneumonia from Influenza-A viral pneumonia and healthy cases with pulmonary CT images using deep learning techniques.

Keywords: COVID-19, Location-attention classification model, Computed tomography

The diversity severity gives accurate results from mild to critical conditions of a covid-19 patient. This method is implemented by taking a pre-trained ResNet network to diagnose the severity. CT Scan images and five-fold cross-validation of 408 patients were considered. This model predicts the disease severity accuracy and also progression using a CT scan image.

The CT imaging of COVID-19 presents several distinct manifestations, according to studies. These focal ground-glass shadows mainly distributed along the pleura, multiple consolidation shadows accompanied by the “halo sign” of the surrounding ground-glass shadow, multiple consolidations of different sizes, and grid-shaped high-density shadows.

The Deep Learning CT model contains five clinic-radiological features that serve as an efficient tool for prediction than individual CT parameters. In this study, deep learning technology was used to design a classification network

for distinguishing COVID-19 from IAVP. In terms of the network structure, the classical ResNet was used for feature extraction. A comparison was made between models with and without an added location-attention mechanism. The experiment showed that the aforementioned mechanism could better distinguish COVID-19 cases from others. Furthermore, multiple enhancement methods were involved in our study, such as image patch vote and Noisy-OR Bayesian function, in order to determinate the dominating infection types. All these efforts produced a consistent improvement in the average f_1-score and accuracy rate.

This study has some limitations. First, the manifestations of COVID-19 may have some overlap with the manifestations of other pneumonias such as IAVP, organizing pneumonia, and eosinophilic pneumonia. We only compared the CT manifestation of COVID-19 with that of IAVP. A clinical diagnosis of COVID-19 still needs to combine the patient's contact history, travel history, first symptoms, and laboratory examination. Second, the number of model samples was limited in this study. The number of training and test samples should be expanded to improve the accuracy in the future. More multi-center clinical studies should be conducted to cope with the complex clinical situation. Moreover, efforts should be made to improve the segmentation and classification model. A better exclusive model could be designed for training, the segmentation and classification accuracy of the model.

In this case study, they presented a novel method that can screen CT images of COVID-19 automatically by means of deep learning technologies. Models with a location-attention mechanism can classify COVID-19, IAVP, and healthy cases with an overall accuracy rate of 86.7%.

2.2 Paper-2: COVID-19 Virus Detected using Real Time RT-PCR

Objectives: To optimize a plasmonic PCR system and perform rapid, one step RT- PCR for the samples.

Keywords: SARS, Coronavirus, Real-Time PCR, RT-PCR, TaqMan, cytokine

Real time RT–PCR is a nuclear-derived method for detecting the presence of specific genetic material in any pathogen, including a virus. Originally, the method used radioactive isotope markers to detect targeted genetic materials, but subsequent refining has led to the replacement of isotopic labelling with special markers, most frequently fluorescent dyes. This technique allows scientists to see the results almost immediately while the process is still ongoing, whereas conventional RT–PCR only provides results at the end of the process. Real time RT–PCR is one of the most widely used laboratory methods for detecting the COVID-19 virus. While many countries have used real time RT–PCR for diagnosing other diseases, such as Ebola virus and Zika virus, many need support in adapting this method for the COVID-19 virus, as well as in increasing their national testing capacities.

The real time RT–PCR technique is highly sensitive and specific and can deliver a reliable diagnosis in as little as three hours, though laboratories take on average between six and eight hours. Compared to other available virus isolation methods, real time RT–PCR is significantly faster and has a lower potential for contamination or errors, as the entire process can be carried out within a closed tube. It continues to be the most accurate method available for the detection of the COVID-19 virus.

However, real time RT–PCR cannot be used to detect past infections, which is important for understanding the development and spread of the virus, as viruses are only present in the body for a specific window of time. Other methods are necessary to detect, track and study past infections, particularly those which may have developed and spread without symptoms.

The RNA is reverse transcribed to DNA using a specific enzyme. Scientists then add additional short fragments of DNA that are complementary to specific parts of the transcribed viral DNA. If the virus is present in a sample, these fragments attach themselves to target sections of the viral DNA.

In conclusion, real-time RT-PCR assay permitted rapid, sensitive, and specific detection of SARS-CoV in clinical specimens and provided needed diagnostic support during the recent SARS outbreak. Widely deploying this assay through the LRN will enhance our ability to provide a rapid response in the event of the possible return of SARS.

2.3 Paper-3: Bidirectional elastic registration algorithm using chest CT images

Keywords: B-spline registration, affine registration, spline affine registration, iterative closest point, sparse linear equations

Lung boundary was segmented using a bidirectional elastic registration algorithm. The middle regions of the lungs were used to detect the virus. As a spatial normalization procedure for two images, which could be acquired at different times or from different modalities, the objective of an image registration operation is to find an optimal transformation to align two given images in the same coordinate system. The motivation is typically to compare the morphological variations of two images. There have been numerous

algorithms developed for this purpose in the past, particularly in the area of medical image analysis.

In this study, they developed a new registration scheme termed as B-spline affine transformation (BSAT). The innovation of this scheme lies in two aspects: (1) defining an affine transformation instead of the traditional translation at each control point, and (2) defining a bi-directional instead of the traditional unidirectional objective / cost function. Mathematically, the developed B-spline affine transformation (BSAT) is a generalized model of the traditional B-Spline transformation (BST). The primary difference between the B-spline model with affine transformations and the B-spline model with displacements or translation are their regularizations.

The regularization of the former model as in Equation (7) gives the transformation more freedom so that the transformation will have a zero cost on the regularization when displacing, rotating, rescaling, and shearing. On the contrary, the regularization of the models with displacements gives the transformation much fewer freedoms, where the transformation has a zero cost on the regularization only with translation.

Although the B-spline model also has zero cost in the model constraint when using the bending energy (second order constraint), the second order constraint might be more sensitive than the first order constraint used in the B-spline affine model. As our experiments demonstrate, the affine transformation defined at the B-spline control points is capable of obtaining smaller registration errors than the traditional displacement. The traditional ICP methods use a unidirectional cost function that always pushes the moving feature points to the fixed feature points. In contrast, the proposed bidirectional cost function not only pushes the moving feature points to the fixed feature

points but also pull the moving feature points away from the fixed feature points at the same time, depending on which direction will lead to a smaller cost. In our experiments, bidirectional cost energy decreases faster than the unidirectional cost energy and converges to smaller values.

In this study, a new registration scheme is proposed to align two images based on feature points. The affine transformation is used to replace the traditional displacement at each B-spline control point, and a bidirectional distance cost function is used to replace the traditional unidirectional distance cost function.

Detecting abnormalities in the lung region is done by using a threshold. This research concludes with an accuracy of 81.3% of severity and non-severity.

2.7 SUMMARY OF RESEARCH PAPERS

They, introduced COVID-Net, a deep convolutional neural network design for the detection of COVID-19 cases from CXR images that is open source and available to the general public. We also introduce COVIDx, an open access benchmark dataset that is comprised of 13,975 CXR images across 13,870 patient cases from five open access data repositories.

Moreover, we investigated how COVID-Net makes predictions using an explain ability method in an attempt to gain deeper insights into critical factors associated with COVID cases, which can aid clinicians in improved screening as well as improve trust and transparency when leveraging COVID-Net for accelerated computer-aided screening.

By no means a production-ready solution, the hope is that the promising results achieved by COVID-Net on the Covid test dataset, along with the fact that it is available in open source format alongside the description on constructing the open source dataset, will lead it to be leveraged and build upon by both researchers and citizen data scientists alike to accelerate the development of highly accurate yet practical deep learning solutions for detecting COVID-19 cases from CXR images and accelerate treatment of those who need it the most. Future directions include continuing to improve sensitivity and PPV to COVID-19 infections as new data is collected, as well as extend the proposed COVID-Net to risk stratification for survival analysis, predicting risk status of patients, and predicting hospitalization duration which would be useful for triaging, patient population management, and individualized care planning.

CT imaging is crucial for diagnosis, assessment and staging COVID-19 infection. Says this Follow-up scans every 3-5 days are often recommended for disease
progression. It has been reported that bilateral and peripheral ground glass opacification (GGO) with or without consolidation are predominant CT findings in COVID-19 patients.

However, due to lack of computerized quantification tools, only qualitative impression and rough description of infected areas are currently used in radiological reports. In this paper, a deep learning (DL)-based segmentation system is developed to automatically quantify infection regions of interest (ROIs) and their volumetric ratios with respect to the lung.

The performance of the system was evaluated by comparing the automatically segmented infection regions with the manually- delineated ones

on 300 chest CT scans of 300 COVID-19 patients. For fast manual delineation of training samples and possible manual intervention of automatic results, a human-in-the-loop (HITL) strategy has been adopted to assist radiologists for infection region segmentation, which dramatically reduced the total segmentation time to 4 minutes after 3 iterations of model updating.

For, a dataset of X-Ray images from patients with common pneumonia, Covid-19, and normal incidents was utilized for the automatic detection of the Coronavirus. The aim of the study is to evaluate the performance of state-of-the-art Convolutional Neural Network architectures proposed over recent years for medical image classification. Specifically, the procedure called transfer learning was adopted. With transfer learning, the detection of various abnormalities in small medical image datasets is an achievable target, often yielding remarkable results.

The dataset utilized in this experiment is a collection of 1427 X-Ray images. 224 images with confirmed Covid- 19, 700 images with confirmed common pneumonia, and 504 images of normal conditions are included. The data was collected from the available X-Ray images on public medical repositories. With transfer learning, an overall accuracy of 97.82% in the detection of Covid-19 is achieved.

The purpose of this research is to evaluate the effectiveness of state-of-the- art pre- trained convolutional neural networks proposed by the scientific community, regarding their expertise in the automatic diagnosis of Covid-19 from thoracic X-rays. To achieve this, a collection of 1427 thoracic X-ray scans is processed and utilized to train and test the CNNs.

CHAPTER 3
METHODOLOGY

Based on the above literature survey, we finally decided to implement this project by making use of an efficient algorithm. In this project, we are trying to detect covid-19 using deep learning CT image classification.

These are the methods by using which we will be building a system to detect covid-19 from a chest CT scan. It aims to reduce false negatives and false positives and increase true positive and true negatives by creating a neural network from scratch and training our model to get effective results.

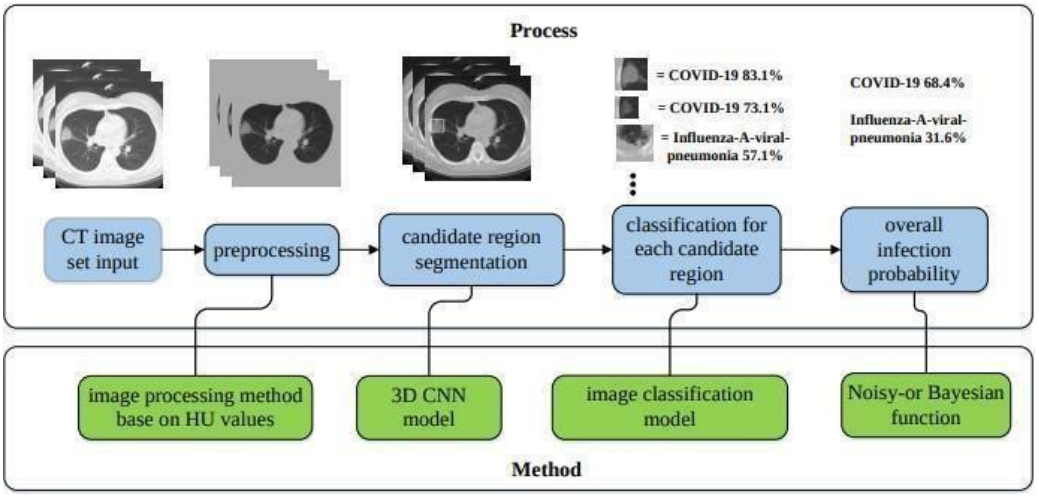
3.1 EXISTING METHODOLOGY

OBJECTIVE: This study aimed to establish an early screening model to distinguish COVID-19 pneumonia from Influenza-A viral pneumonia and healthy cases with pulmonary CT images using deep learning techniques.

Key words: coronavirus disease 2019 pneumonia, COVID-19, deep learning, computed tomography, convolution neural network, location-attention network.

The CT imaging of COVID-19 present several distinct manifestations according to previous studies. The manifestations include focal ground glass shadows mainly distributed in bilateral lungs, multiple consolidation shadows accompanied by the "halo sign" of surrounding ground glass shadow in both lungs, mesh shadows and bronchiectasis and inflating signs inside the lesions, and multiple consolidation of different sizes and grid-shaped high-density

shadows. However, it is not objective and accurate to distinguish COVID-19 from other diseases only with human eyes. In comparison, deep learning system-based screen models revealed more specific and reliable results by digitizing and standardizing the image information. Hence, they can assist physicians to make a quick clinical decision more accurately, which would benefit on management of suspected patients.



In this study, the deep learning technology was used to design a classification network for distinguishing the COVID-19 from Influenza-A viral pneumonia. In terms of the network structure, the classical ResNet was used for feature extraction.

The manifestation of COVID-19 may have some overlap with the manifestations of other pneumonias such as Influenza-A viral pneumonia, organic pneumonia and eosinophilic pneumonia. The clinical diagnosis of COVID-19 needs. To combine the patients', contact history, travel history, first symptoms and laboratory examination. In this study, the number of model samples was limited. Hence, the training and test the number of samples should

be expand to improve the accuracy in the future. More multi-centre clinical studies should be conducted to cope with the complex clinical situation.

Moreover, efforts should be made to improve the segmentation and classification model. A better exclusive models should be designed for training, the segmentation and classification accuracy of the model should be improved, and the generalization performance of this algorithm should be verified with a larger data set.

In this multi-center case study, they had presented a novel method that could screen COVID-19 fully automatically by deep learning technologies. Models with location- attention mechanism could more accurately classify COVID-19 at chest radiography with the overall accuracy rate of 86.7 % and could be a promising supplementary diagnostic method for frontline clinical doctors.

3.2 PROPOSED SYSTEM

As a non-invasive imaging approach, CT scan can depict specific characteristics manifestations in the lung associated with COVID-19. Therefore, CT scan could serve as the most accurate way for early diagnosis of COVID-19.

We used CT Scan images to predict the virus by identifying the distortions in the images. Generally, distortions caused by pneumonia have fluid build-up in the lung, which appears as opacities in the lung. The research paper on applying deep learning to diagnose COVID-19 using CT scans points out some unique characteristic features found in CT scans of lungs infected by COVID-19 compared to other types of pneumonia, resulting from different causes.

In this project we are using **RESIDUAL NEURAL NETWORK (ResNet)** used for the detection of the novel corona virus disease – are also known as the COVID-19 through the chest scanning images.

Chest X-rays and CT scans: These imaging modalities can be used to detect characteristic features of COVID-19 such as ground-glass opacities and consolidations in the lungs. They are not as accurate as laboratory tests but can provide rapid results.

For the proposed system we followed the below mentioned procedure to build the classifier:

1. Data Set Acquisition:

In our paper, we will be using a large dataset of CT scans publicly available to identify covid-19. The SARS-Cov-2 Ct Scan Dataset. The dataset contains 1252 CT scans of patients infected by the SARS-CoV-2 virus and 1230 CT scans of non-infected SARS-CoV-2 patients with covid-19 diseases.

To train a robust classifier, we need to have information about the non-covid patients who contain other diseases that might affect the lungs and cause opacities. An **Adaptive bilateral filter** is a type of image filtering technique that is commonly used in medical imaging, including CT (computed tomography) scans. The goal of the filter is to reduce noise and improve image quality, making it easier for medical professionals to analyze and interpret the images.

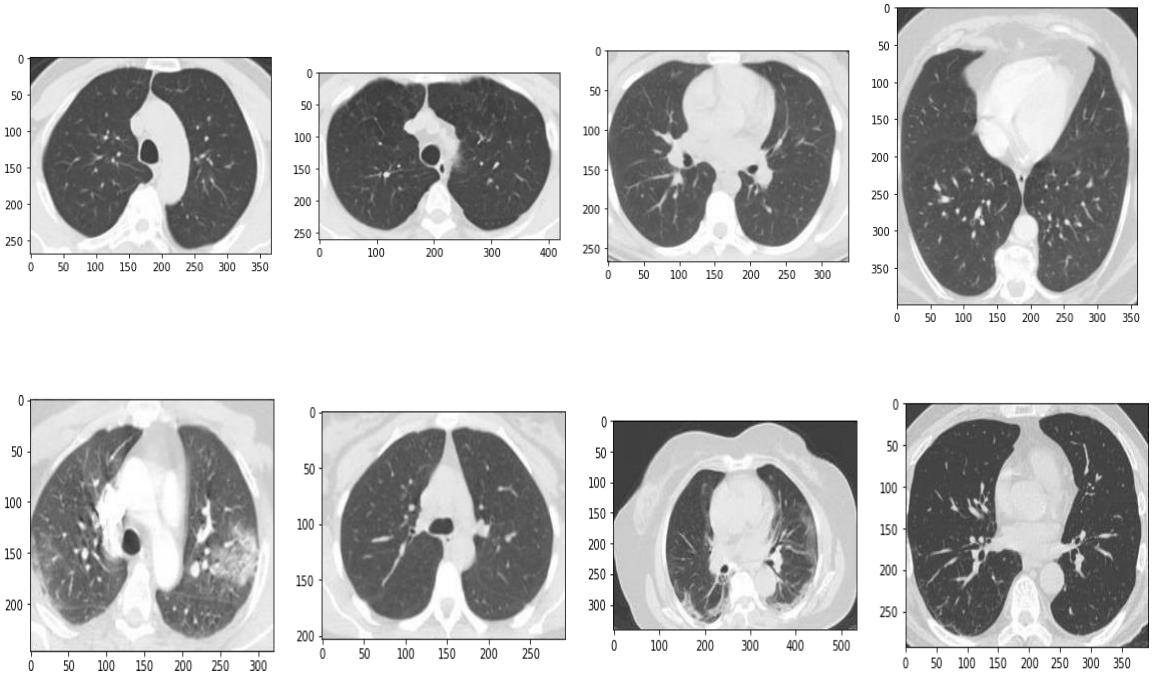


Fig 3.2: Sample images

Luckily, this dataset is not skewed toward any of the classes. All images are of different shapes, which implies the necessity of reshaping. This study takes us to the next module, preprocessing.

2. Performing Data Preprocessing:

All the CT scan images undergo preprocessing that standardize the images, and then they are used to train and test the deep learning model for feature extraction. The convolutional neural networks will process these extracted features and classify the CT image as covid or non-covid.

a) The input images are preprocessed to make them compatible with a pre-trained network, i.e., an image which is of size 224 x 224 x 3 converted into 64 x 64 x 3 to maintain uniformity.

b) We labeled all the CT images from train data to 0 (no covid) or 1 (covid).

c) We performed normalization on the image to have a mean of 0 and a standard deviation of 1. After completing data preprocessing, we shall be sending the images to the image classification model, which we build using convolutional neural networks.

3.2.1 System Architecture

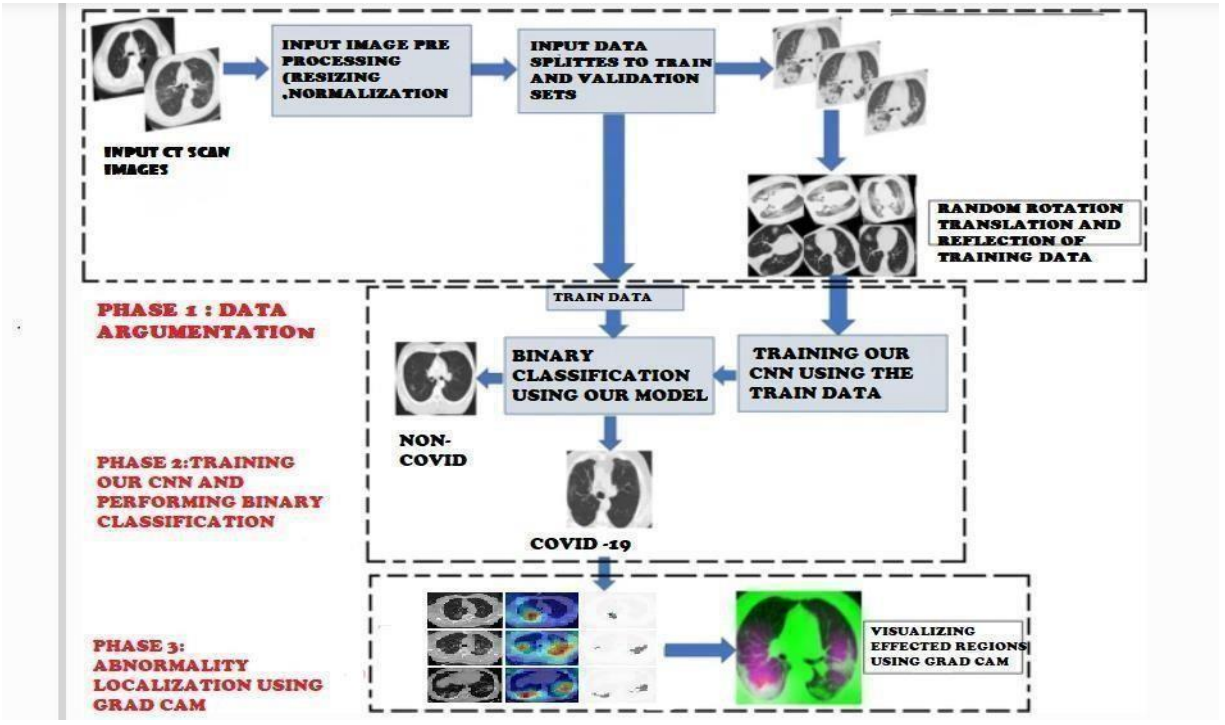


Fig 3.2.1 System Architecture

3. Image Classification by Extracting Features:

From the existing systems, we have observed that the Deep Learning models provide better accuracy and results. A Deep Learning-based model is used in this system. Convolutional neural networks function just like a human brain.

A Neural Network has neurons arranged in layers; input passes through the layers from the input layer to the output layer and generates the output. We use convolutional neural networks to deal with inputs having 2D and 3D formats and extract various spatial features.

Instead of extracting the features manually, unlike in Traditional Machine Learning techniques, the features are automatically extracted in Neural Networks by the hidden layers. These can also perform well even if the dataset is too large. Neural Networks are generally trained using the Back Propagation algorithm.

We built a customized Convolutional neural network from scratch and performed binary classification (covid or non-covid).

The COVID-19 is detected using CT scan images decomposed to three-level using stationary wavelet.

A three-phase detection model is proposed to improve the detection accuracy and the procedures are as follows;

Phase1- Data augmentation

Phase2- COVID-19 detection using CNN learning model and

Phase3- Abnormality localization in CT scan images

DATA AUGMENTATION:

CT scan images compatible with the learning-based model.

For this, pre-processing steps:

Change in input image data type, resizing of the input images, and normalization of input images. To resolve this, the pre-trained transfer learning-based CNN model with data augmentation is performed.

The input images are pre-processed to make it compatible with a pre-trained residual network, i.e., an image of size 224x224x3 converted into 64 x64 x3 to maintain the uniformity.

CNN LEARNING MODEL:

The pre-trained learning-based COVID-19 detection model classifies a CT scan of lungs into binary classes: a) COVID-19 and, b) Non-COVID. To classify new images, retrain a pre-trained model by updating fully connected layers according to the input augmented dataset.

During this process, images sizes are again adjusted but the uniformity in size of input CT scan used by the pre-trained layers is retained due to pre-processing step. Then resized CT scan images are normalized within the confined range.

The training parameters opted for the learning-based CNN model are “rmsprop” optimizer is used, batch size is 64, the training is performed up to 30 epochs. The performance of different networks is examined based on the following parameters: Accuracy, Precision, Recall. The role of the convolution layer is to provide low-level features like color, edges, and gradient operation.

ABNORMALITY LOCALIZATION USING GRAD CAM:

Convolutional Neural Networks (CNNs) and other deep learning networks have enabled unprecedented breakthroughs in a variety of computer vision tasks from image classification to object detection, semantic segmentation, image captioning and more recently visual question answering.

While these networks enable superior performance, their lack of decomposability into intuitive and understandable components makes them hard to interpret. Consequently, when today's intelligent systems fail, they fail spectacularly disgracefully without warning or explanation, leaving a user staring at an incoherent output, wondering why. Interpretability of Deep Learning models matters to build trust and move towards their successful integration in our daily lives. To achieve this goal the model transparency is useful to explain why they predict what they predict.

A very popular technique known as Class Activation Map or CAM used to solve the problem to some extent. Though CAM is a good technique to demystify the working of CNN and build customer trust in the applications developed, they suffer from some limitations. One of the drawbacks of CAM is that it requires feature maps to directly precede the SoftMax layers, so it is applicable to a particular kind of

CNN architectures that perform global average pooling over convolutional maps immediately before prediction. (i.e conv feature maps → global average pooling →SoftMax layer). Such architectures may achieve in inferior accuracies compared to general networks on some tasks or simply be inapplicable to new tasks.

In this post, we discuss a generalization of CAM known as Grad-Cam. Grad-Cam published in 2017, aims to improve the shortcomings of CAM and claims to be compatible with any kind of architecture.

The technique does not require any modifications to the existing model architecture and this allows it to apply to any CNN based architecture, including those for image captioning and visual question answering. For fully-convolutional architecture, the Grad-Cam reduces to CAM.

APPROACH:

Several previous works have asserted that deeper representations in a CNN capture the best high-level constructs. Furthermore, CNN's naturally retrain spatial information which is lost in fully-connected layers, so we can expect the last convolutional layer to have the best tradeoff between high-level semantics and detailed spatial information. Grad-Cam, unlike CAM, uses the gradient information flowing into the last convolutional layer of the CNN to understand each neuron for a decision of interest. To obtain the class discriminative localization map of width u and height v for any class c, we first compute the gradient of the score for the class c, yc (before the SoftMax) with respect to feature maps Ak of a convolutional layer.

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

Calculating weights ak

After calculating ak for the target class c, we perform a weighted combination of activation maps and follow it by ReLU.

$$L^{c}_{\text{Grad-CAM}} = ReLU\underbrace{\left(\sum_{k} \alpha^{c}_{k} A^{k}\right)}_{\text{linear combination}}$$

Linear Combination followed by ReLU resulting in final class discriminative map

Without ReLU, the class activation map highlights more than that is required and hence achieve low localization performance.

Each layer of a pre-trained CNN network contains channels (consists of many 2-D arrays).

3.2.2 Our CNN Architecture

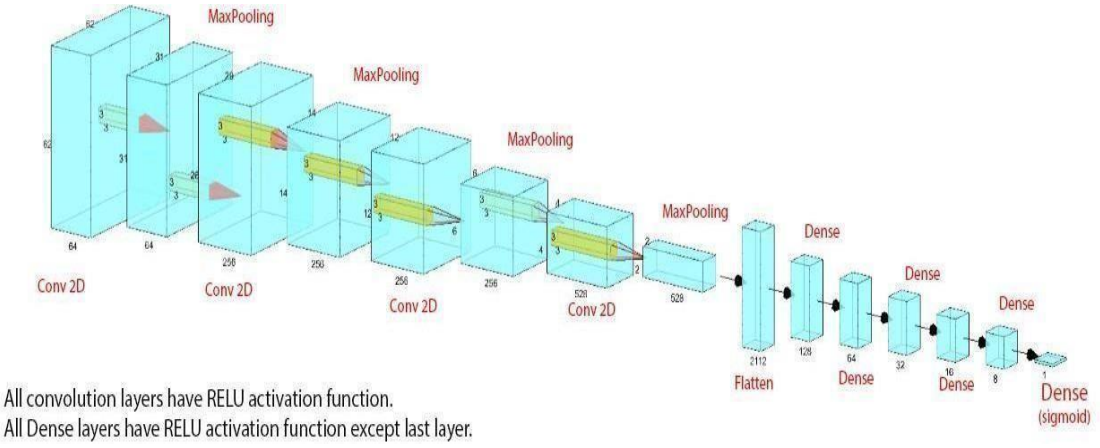


Fig 3.2.2: Our CNN Model

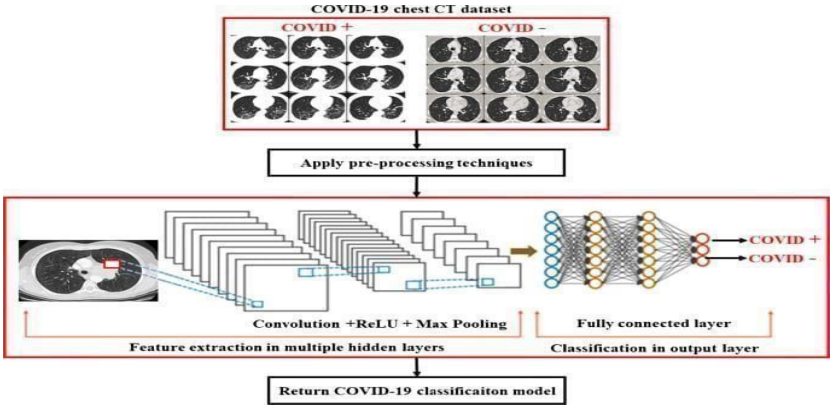
a) Convolutional layers: A convolution layer is a fundamental component of the CNN architecture that performs feature extraction, which typically consists of a combination of linear and nonlinear operations, i.e., convolution operation and activation function. In convolutional layer we used “ReLu” activation function.

We described the kernel values, stride value while constructing this layer.

b) Pooling layer: A pooling layer provides a typical down sampling operation which reduces the in-plane dimensionality of the feature maps in order to introduce a translation invariance to small shifts and distortions, and decrease the number of subsequent learnable parameters.

c) Dense Layers: These are the important layers in which classification takes place As it’s a **binary classification problem**, we will use sigmoid as the activation function for the output.

3.2.3 Visual Representation Of Training Our Model



Building the model:

We build a 15-layer sequential network containing four convolutional layers, which employed stochastic gradient descent with momentum to update all network parameters. Each followed by a max-pooling layer; finally, a flatten layer is followed by six dense layers for generating the diagnosis.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 64)	1792
max_pooling2d (MaxPooling2D)	(None, 31, 31, 64)	0
conv2d_1 (Conv2D)	(None, 29, 29, 256)	147712
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 256)	0
conv2d_2 (Conv2D)	(None, 12, 12, 256)	590080
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 256)	0
conv2d_3 (Conv2D)	(None, 4, 4, 528)	1217040
max_pooling2d_3 (MaxPooling2	(None, 2, 2, 528)	0
flatten (Flatten)	(None, 2112)	0
dense (Dense)	(None, 128)	270464
dense_1 (Dense)	(None, 64)	8256
dense_2 (Dense)	(None, 32)	2080
dense_3 (Dense)	(None, 16)	528
dense_4 (Dense)	(None, 8)	136
dense_5 (Dense)	(None, 1)	9

Total params: 2,238,097
Trainable params: 2,238,097
Non-trainable params: 0

a) We shuffle the data and split all the available images into train data (80%) and test data (20%).

b) We set the batch size to 64 for train data which means 64 images are processed as chunks and epochs to 30.

c) We used the sigmoid activation function in the last layer to perform binary classification so the output could be classified into two categories. For this case, greater than or equal to 0.5 implies COVID-19 negative. And less than 0.5 corresponds to COVID-19 positive.

d) With 'rmsprop' optimization, the model is compiled with the default learning rate.
The loss function used was binary cross-entropy.

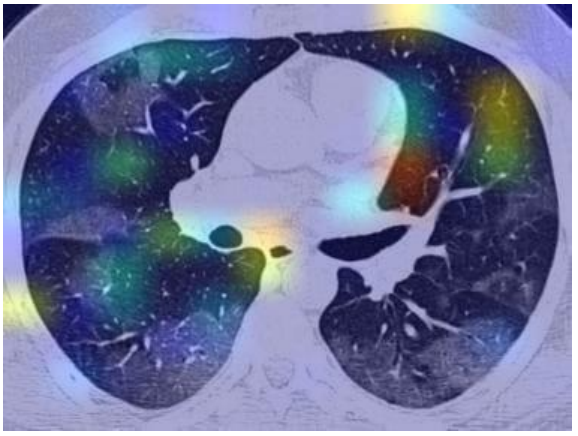
5. Grad-cam to Visualize Abnormalities:

After training our model, it is time to test how well it is performing.

Testing is used to evaluate the generalizing ability of the model by giving unseen data to it. The test data is given to the model in this phase.

Now that we have built an image classification model, to know how well our model is predicting, build trust, and integrate the model in our life, we need to have more model transparency to understand why it predicts.

For that, we will be using an algorithm known as the gradient-based class activation map technique (GRAD CAM) to visualize the regions of abnormalities for the target class as predicted by our model.



Highlighted regions indicate abnormality

This algorithm uses the gradient information flowing into the last convolutional layer of the CNN to visualize and understand each neuron for a decision of interest and obtain the class discriminative localization map.

6. Gradient Mutated Leader Algorithm

Here are the steps involved in using the GML algorithm for COVID-19 detection: Preprocess the CT images of the lungs: Before applying the GML algorithm, the CT images need to be preprocessed to enhance the features that are relevant for COVID-19 detection. This may involve noise removal, image normalization, and segmentation to identify the regions of interest.

CHAPTER 4

MODULE

In our project we have 3 modules

1.Preprocessing and Data augmentation Module

2.Image Classification Module

3. Evaluation Module

4.1 PREPROCESSING DATA AUGMENTATION MODULE

We are using a publicly available SARS-CoV-2 CT scan dataset, containing 1252 CT scans that are positive for SARS-CoV-2 infection (COVID-19) and 1230 CT scans for patients non-infected by SARS-CoV-2, 2482 CT scans in total. These data have been collected from real patients in hospitals from Sao Paulo, Brazil.

Luckily, this dataset is not skewed toward any of the classes. All images are of different shapes, which implies the necessity of reshaping

We labeled all the Ct images from train data to 0(no covid) or 1(covid). We performed normalization on the image to have a mean 0 and standard deviation of 1. After performing data preprocessing, we shall be sending the images to the image classification model which we build using convolutional neural networks.

4.2 IMAGE CLASSIFICATION MODULE

Image Classification Module contains our customized convolution neural network model which we have built from scratch.
Model contains 4 convolution layers of varying filter and stride sizes followed by Max pooling layers.

All the convolution layers are regulated with “Rectified Linear Unit” activation function.

Then we have 6 dense layers with varying filter sizes, final dense layer contains sigmoid activation function since we need binary output.

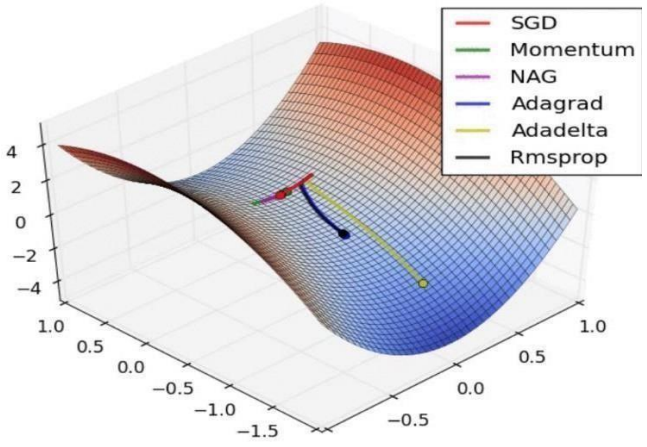
The optimizer we used is **“rmsprop”** and followed by “**binary cross entropy”** for loss parameter.
Input size of the image is 64 X 64 X 3 px.

Since we need to identify the abnormalities in lungs with are basically smaller portions in the lungs CT, we needed to extract a greater number of parameters. Most of the convolutional layers have 256 number of filters and 3x3 kernel size.

After we receive the image from preprocessing module, we pass the image to the classification module. This model classifies the image as covid or non-covid depending upon its own intelligence.

4.2.1 RMSprop Optimizer

Optimizers are a crucial part of the neural network, understanding how they work would help you to choose which one to use for your application.



The RMSprop optimizer is similar to the gradient descent algorithm with momentum. The RMSprop optimizer restricts the oscillations in the vertical direction. Therefore, we can increase our learning rate and our algorithm could take larger steps in the horizontal direction converging faster.

The difference between RMSprop and gradient descent is on how the gradients are calculated. The following equations show how the gradients are calculated for the RMSprop and gradient descent with momentum. The value of momentum is denoted by beta and is usually set to 0.9.
The gist of RMSprop is to maintain a moving (discounted) average of the square of gradients. Divide the gradient by the root of this average.

Optimizers are a crucial part of the neural network, understanding how they work would help you to choose which one to use for your application. I hope this article was helpful in making that decision.

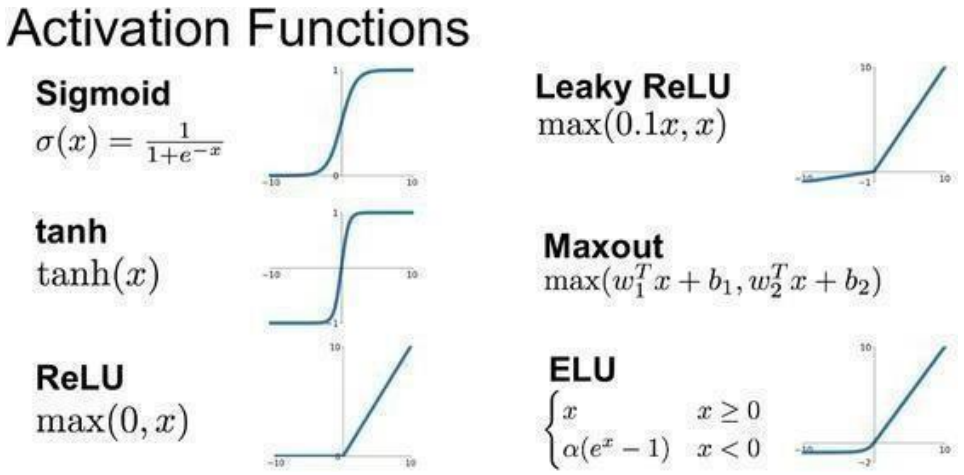
4.2.2 Binary Cross Entropy

Binary cross entropy compares each of the predicted probabilities to actual class output which can be either 0 or 1. It then calculates the score that penalizes the probabilities based on the distance from the expected value. That means how close or far from the actual value. Binary cross entropy is a loss function that is used in binary classification tasks. These are tasks that answer a question with only two choices (yes or no, A or B, 0 or 1, left or right). Several independent such questions can be answered at the same time, as in multi-label classification or in binary image segmentation.

Formally, this loss is equal to the average of the categorical cross entropy loss on many two-category tasks. The binary cross entropy is very convenient to train a model to solve many classification problems at the same time, if each classification can be reduced to a binary choice (i.e. yes or no, A or B, 0 or 1).

It is Also called **Sigmoid Cross-Entropy loss**. It is a **Sigmoid activation** plus a **Cross-Entropy loss**. Unlike **Soft max loss** it is independent for each vector component (class), meaning that the loss computed for every CNN output vector component is not affected by other component values. That's why it is used for **multi- label classification**, were the insight of an element belonging to a certain class should not influence the decision for another class. It's called **Binary Cross-Entropy Loss** because it sets up a binary classification problem between 2 classes for every class in same.

4.2.3 ReLU Activation Function



The **rectified linear activation function** or **ReLU** for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

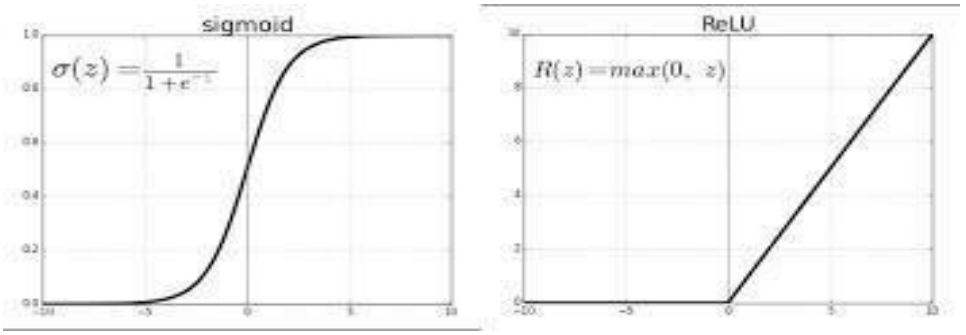
In order to use stochastic gradient descent with backpropagation of errors to train deep neural networks, an activation function is needed that looks and acts like a linear function, but is, in fact, a nonlinear function allowing complex relationships in the data to be learned.

The solution had been bouncing around in the field for some time, although was not highlighted until papers in 2009 and 2011 shone a light on it.
The solution is to use the rectified linear activation function, or ReL for short.
A node or unit that implements this activation function is referred to as a **rectified linear activation unit**, or ReLU for short. Often, networks that use the rectifier function for the hidden layers are referred to as rectified networks.

Adoption of ReLU may easily be considered one of the few milestones in the deep learning revolution, e.g. the techniques that now permit the routine development of very deep neural networks. ReLU function is its derivative both are monotonic. The function returns 0 if it receives any negative input, but for any positive value x, it returns that value back. Thus it gives an output that has a range from 0 to infinity.

4.2.4 Sigmoid Function

A sigmoid function is a bounded, differentiable, real function that is defined for all real input values and has a non-negative derivative at each point[1] and exactly one inflection point. A sigmoid "function" and a sigmoid "curve" refer to the same object.



In general, a sigmoid function is monotonic, and has a first derivative which is bell shaped. Conversely, the integral of any continuous, non-negative, bell-shaped function (with one local maximum and no local minimum, unless

degenerate) will be sigmoidal. Thus the cumulative distribution functions for many common probability distributions are sigmoidal. One such example is the error function, which is related to the cumulative distribution function of a normal distribution.

A sigmoid function is convex for values less than 0, and it is concave for values greater than 0.

4.3 EVALUATION MODULE

While evaluating the results of any coronavirus diagnosis method, the accuracy is not enough. This is because of all the people we test, only a few of them are going to have the virus. Hardly any country in the world has had a positive hit rate of more than 20 percent (positives cases out of all tests done).

In such a case, let's say we develop a solution that just calls everything negative. From an accuracy point of view, the solution would still have 80 percent accuracy, despite being a totally useless classifier.

We therefore need to focus on other metrics, such as:

Sensitivity, or the true positive rate. This is the proportion of true positives out of the total number of positive samples, or simply put, the number of coronavirus infected patients that we correctly classify as positive.

Having a sensitivity which is too low would mean that there are many people who have the virus which our algorithm classified as negative. This is a particularly alarming shortcoming to have, since it would allow many infected people to go home and perhaps spread the virus.

Specificity, or the true negative rate. This is the proportion of the true negatives to the total number of negative samples, or simply put, the number of non-infected people that we correctly classify as negative.

Having a specificity which is too low would mean that we will be telling many people who don't have the virus that they do. While not as alarming as a low sensitivity, it could put undue pressure on health systems if we get too many false positives in our system.

We also use metrics like **Precision** (of all the patients we have diagnosed as positive, how many people actually had the disease; this is useful for measuring how resourceful our test is) and the **F1 Score** (which combines Precision and Sensitivity).

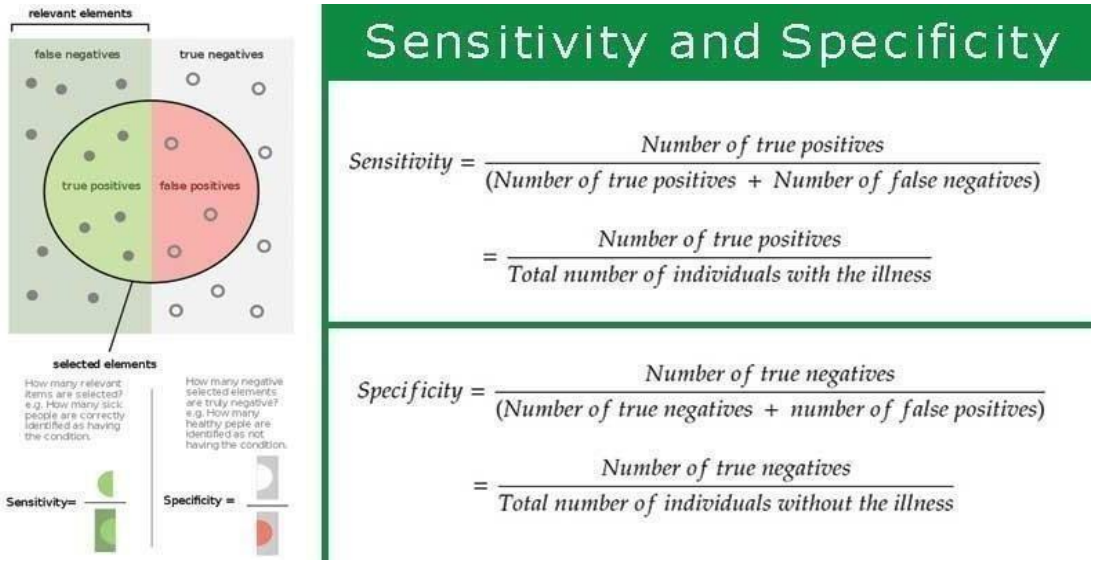
Finally, we got
f1 score = 0.925
Precision = 0.949
Specificity = 0.952
Sensitivity = 0.902

True positive (TP) = the number of cases correctly identified as patient

False positive (FP) = the number of cases incorrectly identified as patient

True negative (TN) = the number of cases correctly identified as healthy

False negative (FN) = the number of cases incorrectly identified as healthy



Accuracy:

The accuracy of a test is its ability to differentiate the patient and healthy cases correctly. To estimate the accuracy of a test, we should calculate the proportion of true positive and true negative in all evaluated cases. Classification accuracy is the ratio of number of correct predictions to all predictions made.

Accuracy = No. of correct predictions / Total no. of Input samples.

Mathematically, this can be stated as:

$$Accuracy=TP+TNTP+TN+FP+FN$$

Sensitivity:

The sensitivity of a test is its ability to determine the patient cases correctly. Sensitivity (True Positive rate) measures the proportion of positives that are correctly identified (i.e. the proportion of those who have some condition

(affected) who are correctly identified as having the condition).To estimate it, we should calculate the proportion of true positive in patient cases.

Mathematically, this can be stated as:

$$\text{Sensitivity}=\frac{TP}{TP+FN}$$

Specificity:

The specificity of a test is its ability to determine the healthy cases correctly. (True Negative rate) measures the proportion of negatives that are correctly identified (i.e. the proportion of those who do not have the condition (unaffected) who are correctly identified as not having the condition).To estimate it, we should calculate the proportion of true negative in healthy cases.

Mathematically, this can be stated as:

$$\text{Specificity}=\frac{TN}{TN+FP}$$

Precision:

Precision is used with recall, the percent of *all* relevant documents that is returned by the search. The two measures are sometimes used together in the F_1 Score (or f-measure) to provide a single measurement for a system. usage of "precision" in the field of information retrieval differs from the definition of accuracy and precision within other branches of science and technology. Precision is defined as the ratio of number of TP to the number of TP plus the number of FP.

Mathematically, this can be stated as:

$$\text{Precision}= TP/\ TP+FP$$

Recall:

In binary classification, recall is called sensitivity. It can be viewed as the probability that a relevant document is retrieved by the query.

It is trivial to achieve recall of 100% by returning all documents in response to any query. Therefore, recall alone is not enough but one needs to measure the number of non-relevant documents also, for example by also computing the precision.

Recall is defined as the ratio of number of TP to the number of TP plus the number of FN.

Mathematically, this can be stated as:

$$\text{Recall} = TP/TP+FN$$

Confusion Matrix

A confusion matrix is a table used to describe the performance of a classification model on test data. This uses the actual labels and predicted labels and how many are correctly and wrongly predicted.

What can we learn from this matrix?

- There are two possible predicted classes: "yes" and "no". If we were predicting the presence of a disease, for example, "yes" would mean they have the disease, and "no" would mean they don't have the disease.

- The classifier made a total of 165 predictions (e.g., 165 patients were being tested for the presence of that disease).

- Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times.

- In reality, 105 patients in the sample have the disease, and 60 patients do not. Let's now define the most basic terms, which are whole numbers (not rate)

- **True positives (TP):** These are cases in which we predicted yes (they have the disease), and they do have the disease.

- **True negatives (TN):** We predicted no, and they don't have the disease.

- **False positives (FP):** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")

- **False negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

CHAPTER 5
EXPERIMENTAL ANALYSIS AND RESULT

5.1 SYSTEM CONFIGURATION

5.1.1 Hardware Configurations

These are the minimum Hardware configurations that are required.

- CPU-1.7 GHz
- GPU-Yes
- RAM: 8 GB
- ROM: 100GB or higher
- Monitor: 1024 x 768 minimum screen resolution.

5.1.2 Software Configurations

These are the Software Configurations that are required.

- Operating System: Linux, Windows, Mac OS
- Language: Python 3, JavaScript
- IDE: Google Colab, Visual Studio.

5.2 SYSTEM REQUIREMENTS

5.2.1 Software Requirements

- Python
- cv2
- Tensorflow
- keras
- Pandas
- NumPy
- Matplot Lib

5.3 Sample Code

5.3.1 Importing all necessary libraries

```
import os import gc
import cv2 import math
import numpy as np
import random
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm
import pandas as pd
from keras.utils import
plot_model from keras import
models from keras import layers
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import make_scorer, accuracy_score, roc_auc_score,
roc_curve
```

5.3.2 Importing dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

5.3.3 Defining helper functions for reading dataset

```
def read_image(filepath): return cv2.imread(os.path.join(data_dir, filepath))
def resize_image(image, image_size):
return cv2.resize(image.copy(), image_size, interpolation =
cv2.INTER_AREA)
```

5.3.4 Having two separate folders for covid and non- covid data set

```
disease_types=['COVID','non-COVID']
data_dir = '/content/drive/MyDrive/COVID'
train_dir = os.path.join(data_dir)
```

5.3.5 Displaying Random Image From Dataset

```
SUBDIR_POS = '/content/drive/MyDrive/COVID/COVID'
SUBDIR_NEG = '/content/drive/MyDrive/COVID/non-COVID'
print(f'Positive samples: {len(os.listdir(SUBDIR_POS))}.')
print(f'Negative samples: {len(os.listdir(SUBDIR_NEG))}.')
```

5.3.6 Model Code

```
model=models.Sequential()
model.add(layers.Conv2D(64,(3,3),activation='relu',input_shape=(64,64,3)))
model.add(layers.MaxPool2D((2, 2)))
model.add(layers.Conv2D(256,(3,3),activation='relu'))
model.add(layers.MaxPool2D((2, 2)))
model.add(layers.Conv2D(256,(3,3),activation='relu'))
model.add(layers.MaxPool2D((2, 2)))
model.add(layers.Conv2D(528, (3, 3), activation='relu'))
model.add(layers.MaxPool2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(128,activation='relu'))
model.add(layers.Dense(64,activation='relu'))
model.add(layers.Dense(32,activation='relu'))
model.add(layers.Dense(16,activation='relu'))
model.add(layers.Dense(8,activation='relu'))
model.add(layers.Dense(1,activation='sigmoid'))
return model
```

5.3.7 Model Training Code

```
model.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['accuracy','Recall','Precision']) history = model.fit(x_train, y_train,
batch_size=64, epochs=30, verbose=1,validation_split=0.1)
```

5.3.8 Plotting Graph For Each Metrics

For accuracy

```
from matplotlib import pyplot as plt
plt.title('Accuracy')
plt.plot(history.history['accuracy'],label = 'training accuracy')
plt.plot(history.history['val_accuracy'],label='validation accuracy')
plt.xlabel('epochs') plt.ylabel('accuracy') plt.legend()
plt.show()
```

For loss

```
plt.title('Loss') plt.plot(history.history['loss'],label='training
loss') plt.plot(history.history['val_loss'],label='validation
loss') plt.xlabel('epochs') plt.ylabel('loss')
plt.legend()
plt.show()
```

For recall

```
plt.title('Recall') plt.plot(history.history['recall'])
plt.plot(history.history['val_recall'])
```

```
plt.ylabel('recall') plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

For precision

```
plt.title('Precision')
plt.plot(history.history['precision'])
plt.plot(history.history['val_precision'])
plt.ylabel('recall')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

5.3.9 Receiver operating characteristic curve

```
import numpy as np
import matplotlib.pyplot asplt from
tqdm.notebook import tqdm import
pandas as pd
fpr = dict() tpr = dict() roc_auc = dict() for i in range(2): fpr[i],
tpr[i], _ = roc_curve(y_test, y_pred) roc_auc[i] = auc(fpr[i],
tpr[i]) plt.figure() plt.plot(fpr[1], tpr[1]) plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate') plt.title('Receiver operating
characteristic') plt.show()
```

5.3.10 Final metric values

```
y_pred = model.predict_classes(x_test)
final_loss, final_accuracy,final_recall,final_precision =
model.evaluate(x_test, y_test) print('Final Loss: {}, Final Accuracy: {}
,Final recall: {},final precision: {} '.format(fi nal_loss,
final_accuracy,final_recall, final_precision))
```

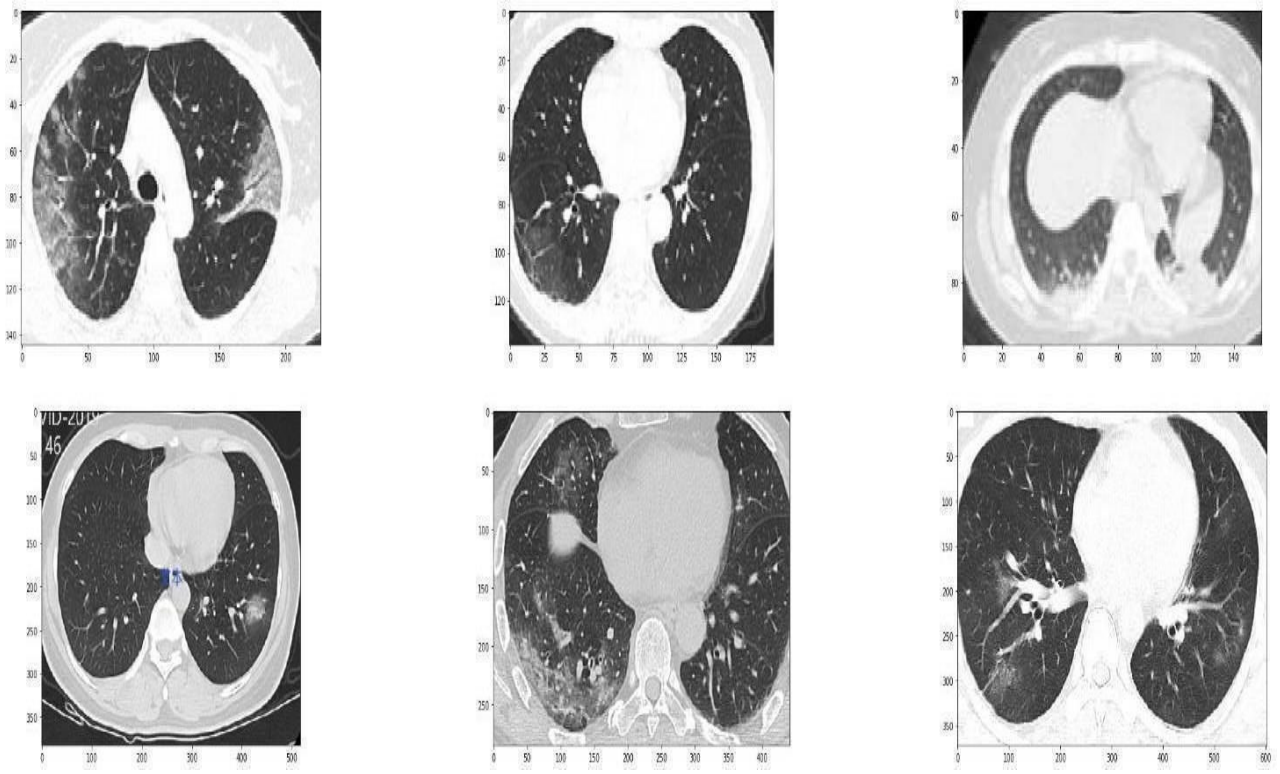
5.3.11 Confusion Matrix

```
from sklearn.metrics import f1_score, roc_curve, auc, precision_score,
accuracy_score , confusion_matrix
ture_negitive, false_Positive, false_negitive, true_Positive =
confusion_matrix(y_test, y_pred).ravel()

import sklearn.metrics
cf_matrix=sklearn.metrics.confusion_matrix(y_test, y_pred)
print(cf_matrix)
import seaborn as sns
sns.set(font_scale=1.5) a="True
neg\n"+str(cf_matrix[0][0])
b="False
pos\n"+str(cf_matrix[0][1])
c="False
neg\n"+str(cf_matrix[1][0])
d="True
Pos\n"+str(cf_matrix[1][1])
```

5.3.12 GRAD CAM

```
model_builder = keras.applications.xception.Xception img_size = (64, 64)
preprocess_input=keras.applications.xception.preprocess_input
decode_predictions=keras.applications.xception.decode_predictions
last_conv_layer_name = "conv2d_2"

# The local path to our target image
img_path =
'/content/drive/MyDrive/ct.jpg'
display(Image(img_path))
```

5.4 INPUTS



5.5 OUTPUTS

5.5.1 Metrics

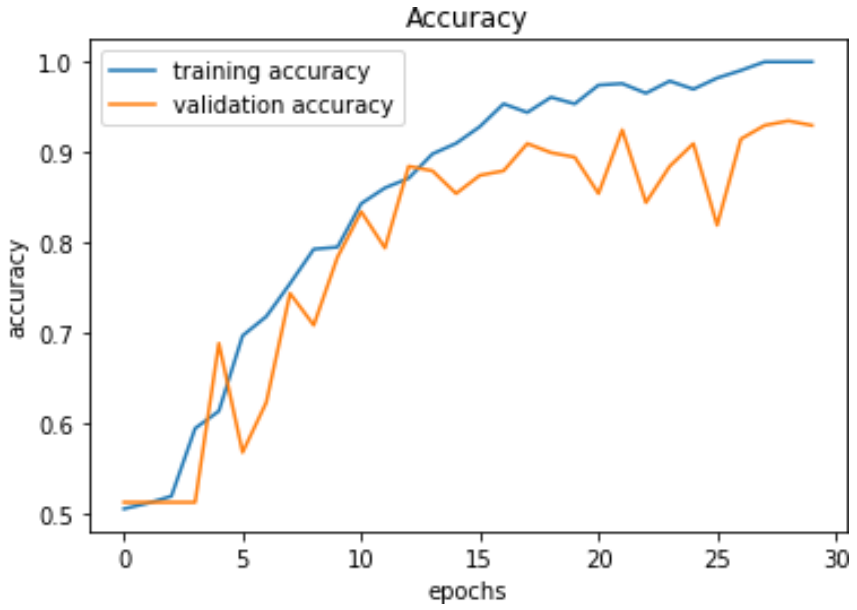


Fig 5.5.1.1 Accuracy Graph

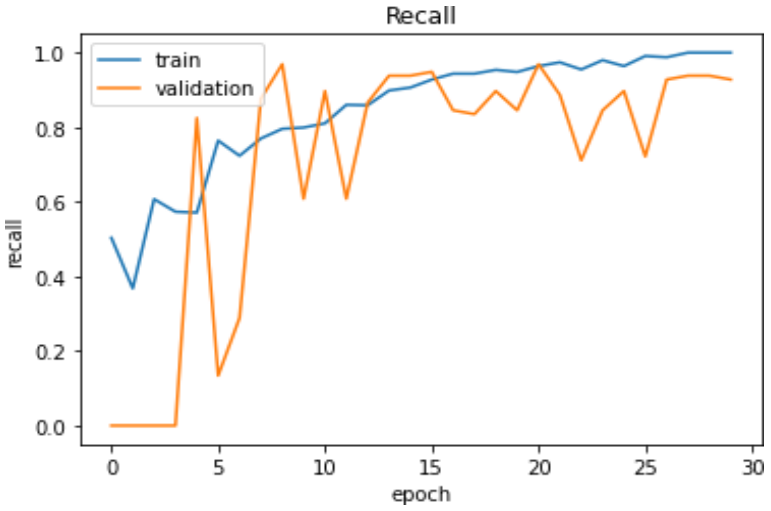


Fig 5.5.1.2 Recall Graph

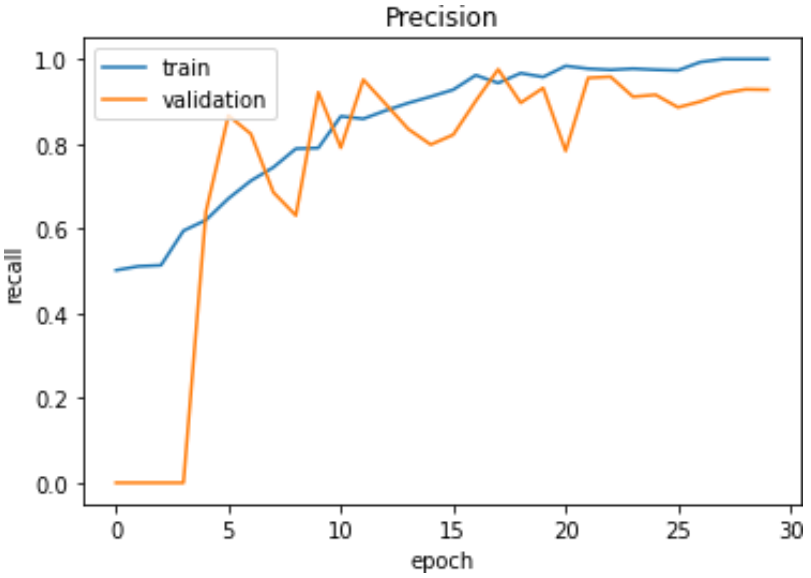


Fig 5.5.1.3 Precision Graph

5.5.2 Receiver operating characteristic curve

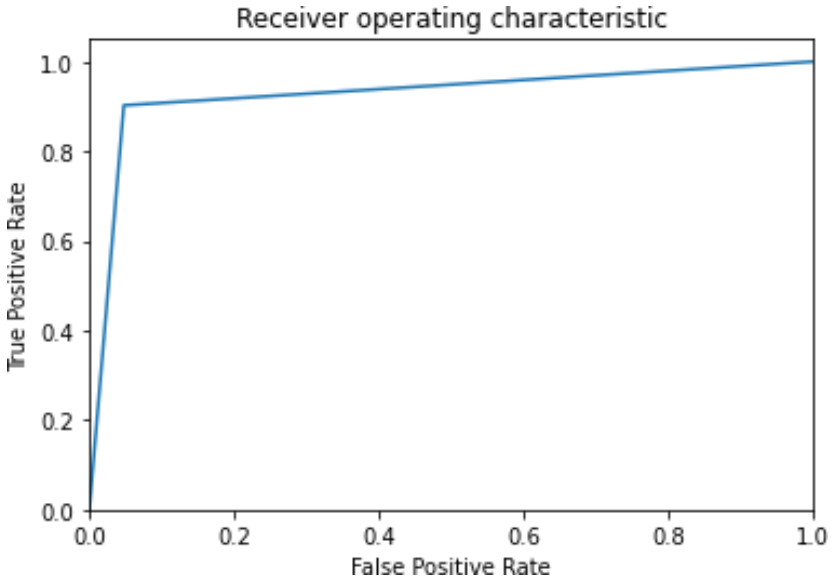


Fig 5.5.2 Receiver operating characteristic Graph

5.5.3 Confusion Matrix

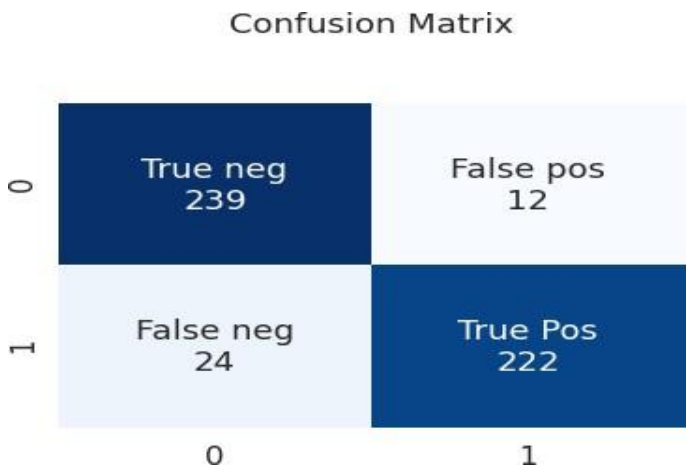


Fig 5.5.3: Confusion Matrix

5.5.4 Grad Cam applied on covid positive image

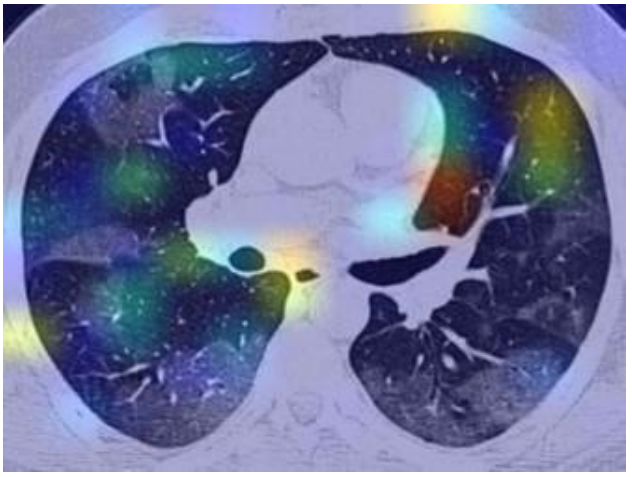
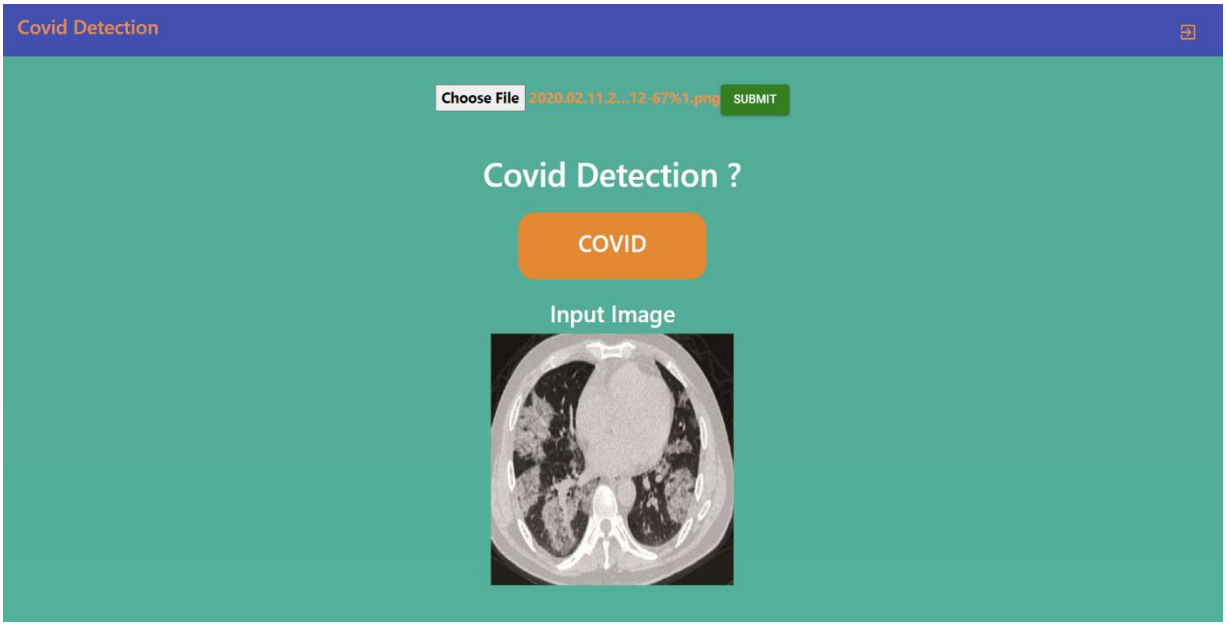


Fig 5.5.4 Highlighed regions in this indicate abnormality

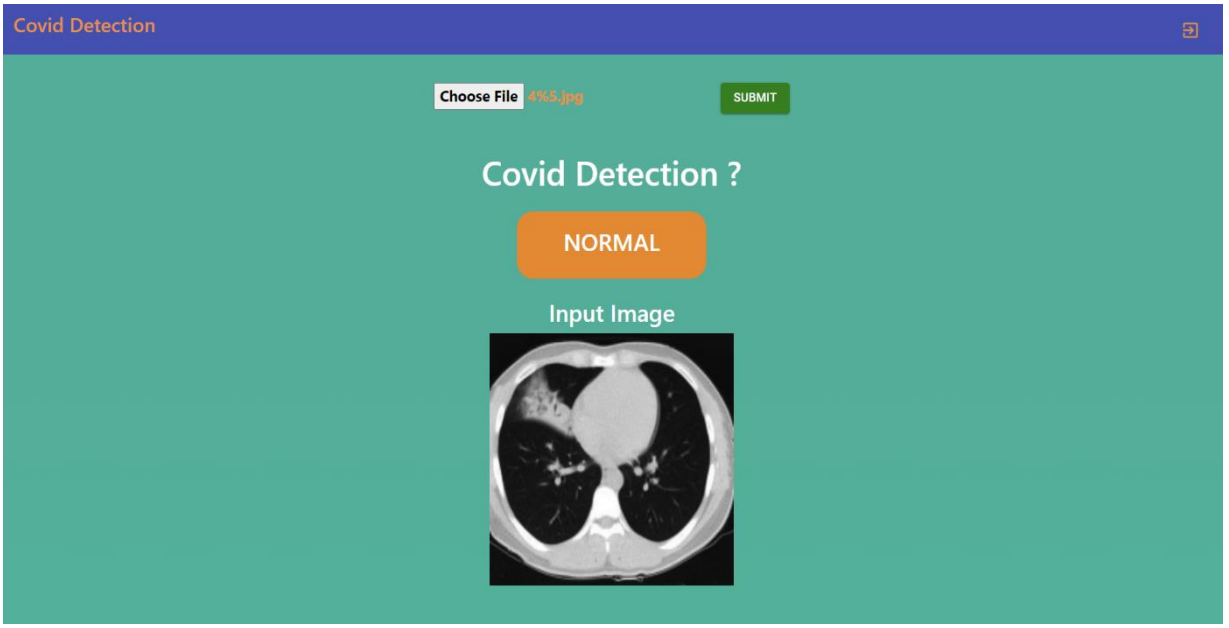
5.6 SCREENSHOTS

Screenshots 1:



COVID (Positive)

Screenshot 2:



NORMAL (Negative)

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

This project is developed to accurately predict covid-19 using lung CT scans. We proposed to investigate and implement a customized deep learning model which can be used to thoroughly analysis and classify lung CT scans.

We successfully managed to yield a high accuracy of 98% for the test data. This model can be used as an efficient tool for early detection and diagnosis of covid-19 which is boon for the medical field. Using the deep learning models could open new insights for detecting and improving testing and treatment for covid-19 patients.

6.2 FUTURE SCOPE

As a future enhancement we can extend our project to perform accurate detection and quick diagnosis of other diseases in various organs using CT scans. It would be a great enhancement in the medical field.

7. REFERENCES

1. Deep Learning System to Screen Coronavirus Disease 2019 Pneumonia
2. https://www.iaea.org/newscenter/news/how-is-the-covid-19-virus- detected-using- real-time-rt-pcr , 28 May 2020.
3. https://www.mayoclinic.org/tests-procedures/covid-19-antibody- testing/about/pac- 20489696 , By mayo clinic staff, 6 May 2021.
4. Deep Learning-Based Model Using Computed Tomography Imaging for Predicting Disease Severity of Coronavirus Disease 2019: https://www.nature.com/articles/s41598-020-76282-0 Jun Chen, Lianlian Wu, Jun Zhang, Liang Zhang, 5 November 2020.
5. Automated quantification of COVID-19 using chest CT images: https://link.springer.com/article/10.1007/s00330-020-07156-2 ,Jiantao Pu, Joseph K. Leader, Andriy Bandos, Shi Ke, Jing Wang, 13 August, 2020.
6. Machine Learning for covid-19 from chest X-rays: https://www.sciencedirect.com/science/article/pii/S1877050920321621 LucaBrunese, FabioMartinelli, 2 October, 2020.
7. Machine learning classification of texture features of portable chest xray covid-19 lung infection: https://pubmed.ncbi.nlm.nih.gov/33239006/ Lal Hussain, Tony Nguyen , Haifang Li , 25 November 2020.
8. https://towardsdatascience.com/detection-of-covid-19-presence-fromchest-x-ray- scans-using-cnn-class-activation-maps-c1ab0d7c294b