

A Project-I Report  
on

**LUNG DISEASE CLASSIFICATION USING X-RAY IMAGES**

Submitted in partial fulfillment of the requirement

For

The award of degree of

**BACHELOR OF TECHNOLOGY**

In

**COMPUTER SCIENCE & ENGINEERING**

**2020-2024**

Submitted By

KEERTHIGARI SHIVANI      20311A05E8

MD ALTAF      21315A0513

POLASI PRANEETH      20311A05G3

Under the Guidance of

MR. DEVAVARAPU SREENIVASARAO

ASSISTANT PROFESSOR



**Department of Computer Science & Engineering**

**SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(AUTONOMOUS)**

Yamnampet (V), Ghatkesar (M), Hyderabad – 501301, T.S.

**AY-2022-2023**

## Department of computer Science and Engineering

### Sreenidhi Institute of Science and Technology



## CERTIFICATE

This is to certify that this Project-I report on “**LUNG DISEASE CLASSIFICATION USING X-RAY IMAGES**”, submitted by **KEERTHIGARI SHIVANI (20315A05E8)** , **MD ALTAF (21315A0513)** , **P. PRANEETH (20315A05G3)** in the year 2023 in the partial fulfillment of the academic requirements of Jawaharlal Nehru Technological University for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide work that has been carried out by them as a part of their **Project-I during Third Year Second Semester**, under our guidance. This report has not been submitted to any other Institute or university for the award of any degree.

<b>Internal guide</b>	<b>Project Coordinator</b>	<b>Head of the Department</b>
Mr. Devavarapu Sreenivasarao	Mrs.K.Padmini	Dr. Aruna Varanasi
Assistant Professor	Assistant Professor	Professor & HOD
Department of CSE	Department of CSE	Department of CSE

**Signature of the External Examiner**

Date:-

## **DECLARATION**

We **KEERTHIGARI SHIVANI (20315A05E8) , MD ALTAF (21315A0513) , POLASI PRANEETH (20315A05G3)** students of **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY, YAMNAMPET, GHATKESAR**, studying III year II semester, **COMPUTER SCIENCE AND ENGINEERING** solemnly declare that the Project-I work, titled “ **LUNG DISEASE CLASSIFICATION USING X-RAY IMAGES** ” is submitted to **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY** for partial fulfillment for the award of degree of Bachelor of technology in **COMPUTER SCIENCE AND ENGINEERING**.

It is declared to the best of our knowledge that the work reported does not form part of any dissertation submitted to any other University or Institute for award of any degree.

## **ACKNOWLEDGEMENT**

We would like to express our gratitude to all the people behind the screen who helped me to transform an idea into a real application.

We would like to express our heart-felt gratitude to our parents without whom we would not have been privileged to achieve and fulfill our dreams. We are grateful to our principal, **DR. T. Ch. Siva Reddy**, who most ably runs the institution and has had the major hand in enabling us to do our project.

We profoundly thank **Dr. Aruna Varanasi**, Head of the Department of Computer Science & Engineering who has been an excellent guide and also a great source of inspiration to our work.

We would like to thank our Coordinator **Mrs.K.Padmini** & my internal guide **Mr. Devavarapu Sreenivasarao** for Project-I for their technical guidance, constant guidelines, encouragement and support in carrying out my project on time at college.

The satisfaction and euphoria that accompany the successful completion of the task would be great but incomplete without the mention of the people who made it possible with their constant guidance and encouragement crowns all the efforts with success. In this context, We would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased my task.

**KEERTHIGARI SHIVANI**

**20311A05E8**

**MD ALTAF**

**21315A0513**

**POLASI PRANEETH**

**20311A05G3**

## ABSTRACT

Lung diseases are a significant cause of morbidity and mortality worldwide, necessitating early and accurate diagnosis for effective treatment. In recent years, advancements in medical imaging, particularly X-ray technology, have provided invaluable insights into the detection and classification of various lung diseases. This project aims to develop a robust and reliable system for the classification of lung diseases using X-ray images.

The proposed methodology employs state-of-the-art deep learning techniques, specifically convolutional neural networks (CNNs), to automatically analyze X-ray images and classify them into different disease categories. The project utilizes a publicly available dataset of labeled X-ray images encompassing a wide range of lung conditions, including pneumonia, tuberculosis, lung cancer, and other respiratory disorders.

The Classification system involves several steps, including data collection, preprocessing of data by removing outliers, noisy data and cleaning, feature extraction using grayscale pixel values, classification of diseases, model training, model evaluation, and deployment. Image Processing is used to preprocess the X-ray images to remove noise and unwanted parts of image and extract relevant features which can be used to classify different types of lung diseases. The performance of the system heavily depends on the quality and quantity of the X-ray images used for training. Machine Learning algorithms such as Convolutional Neural Networks (CNNs), InceptionResnet, EfficientNet models are used which result in classifying X-ray images for diseases such as PNEUMONIA, LUNG OPACITY and COVID-19. After training the model, we evaluate its performance on a test dataset. To evaluate the model's performance, several metrics such as accuracy, precision, recall, and F1 score are employed.

# INDEX

	Page No
<b>1. INTRODUCTION</b>	
1.1 Project Introduction	<b>1-2</b>
1.2 Scope	<b>2</b>
1.3 Project Overview	<b>3</b>
1.4 Objectives	<b>3-4</b>
1.5 Data Set	<b>4</b>
<b>2. LITERATURE SURVEY</b>	
2.1 Existing System	<b>4-5</b>
2.2 Proposed System	<b>5-6</b>
2.3 Related Work	<b>6-7</b>
<b>3. SYSTEM ANALYSIS</b>	
3.1 Functional Requirements	<b>8</b>
3.2 Performance Requirements	<b>8-9</b>
3.3 Software Requirements	<b>9</b>
3.4 Hardware Requirements	<b>9</b>
3.5 Feasibility Study	<b>10</b>
<b>4. SYSTEM DESIGN</b>	
4.1 System Architecture	<b>11-14</b>
4.2 UML Diagrams	<b>14</b>
4.2.1 Use Case Diagram	<b>14</b>
4.2.2 Class Diagram	<b>15</b>
4.2.3 Activity Diagram	<b>15</b>

<b>5. IMPLEMENTATION AND RESULTS</b>	
5.1 Methods / Algorithms Used	<b>16-23</b>
5.2 Modules	<b>24</b>
5.3 Sample Code	<b>24-33</b>
5.4 Results (Accuracy) / Output Screens	<b>34-37</b>
<b>6. CONCLUSION</b>	<b>38</b>
<b>7. FUTURE SCOPE</b>	<b>39</b>
<b>8. BIBLIOGRAPHY</b>	<b>40</b>
<b>ABSTRACT</b>	<b>41</b>
<b>PROJECT CORRELATION WITH PO's AND PSO's</b>	<b>42</b>
<b>NATURE OF THE PROJECT</b>	<b>43</b>
<b>DOMAIN OF THE PROJECT</b>	<b>44</b>

# **1.INTRODUCTION**

## **1.1 PROJECT INTRODUCTION**

Lung diseases are a significant global health concern, affecting millions of people and causing significant morbidity and mortality. Timely and accurate diagnosis is crucial for effective treatment and management of these conditions. X-ray imaging has long been a valuable tool in the diagnosis of lung diseases, providing detailed visual information about the lungs and chest. However, the interpretation of X-ray images is a complex and subjective task that heavily relies on the expertise of radiologists.

In recent years, advancements in Machine Learning and computer vision have opened up new avenues for the accurate and efficient diagnosis of various medical conditions. One area that has seen significant progress is the classification of lung diseases using X-ray images. With the ability to analyze large volumes of medical data and extract meaningful patterns, Machine Learning algorithms can assist healthcare professionals in the early detection and classification of lung diseases, leading to improved patient outcomes and timely interventions. By leveraging the power of Artificial Intelligence and deep learning algorithms, researchers have developed models that can analyze X-ray images and accurately identify various lung conditions. This has the potential to assist radiologists, improve diagnostic accuracy, and expedite treatment decisions.

X-ray imaging has been a cornerstone in the diagnosis of lung diseases for decades. It is a non-invasive and widely available imaging modality that provides a detailed view of the internal structures of the chest, including the lungs. Radiologists play a crucial role in interpreting these images, but the process can be time-consuming and subjective, often leading to variability in diagnoses. This is where the application of machine learning techniques becomes invaluable.

The primary goal of lung disease classification using X-ray images is to develop algorithms that can accurately and automatically identify various lung conditions, such as PNEUMONIA, COVID-19, LUNG CANCER and LUNG OPACITY. By leveraging the power of deep learning and convolutional neural networks (CNNs), researchers have made remarkable strides in this field, achieving levels of accuracy comparable to or even surpassing human radiologists.



However, there are still challenges to overcome in this field. The scarcity of large, diverse, and well-annotated datasets poses a significant obstacle. Additionally, the interpretability of deep learning models and the need for explainability in medical decision-making remain important concerns. Ongoing research efforts are addressing these challenges and exploring new avenues, such as the combination of different imaging modalities and the integration of clinical

## **1.2 SCOPE**

There is a need to research and develop a system that will enable end users to predict chronic diseases without having to visit a physician or doctor for diagnosis. To identify various diseases by observing the symptoms of patients and applying various Machine Learning Models techniques. The scope of our project on lung disease classification using X-ray images extends to various aspects of the classification process. We aim to accurately classify a range of lung diseases, including pneumonia, tuberculosis, lung cancer, and other common respiratory conditions. To achieve this, we will explore the potential of machine learning algorithms, specifically CNN, ResNet, and InceptionResNet, to analyze X-ray images and classify them into relevant disease categories. Additionally, we will incorporate advanced image processing techniques, such as image segmentation and feature extraction, to extract meaningful features that contribute to improved classification performance. Transfer learning techniques will also be investigated, leveraging pre-trained models on large-scale image datasets and fine-tuning them for the specific task of lung disease classification. To enhance the accuracy of the classification process, we will explore the integration of clinical data, such as patient demographics, medical history, and laboratory test results. This additional information can augment the classification process and improve the accuracy of the diagnosis. Moreover, we aim to develop a user-friendly interface or application that enables healthcare professionals to easily input X-ray images and obtain accurate and timely disease classification results. Finally, we will explore the potential for real-time or near-real-time lung disease classification, enabling faster diagnosis and treatment decisions in urgent care scenarios. By encompassing these aspects, our project aims to advance the field of lung disease classification using X-ray images, ultimately improving the accuracy and efficiency of diagnosis in clinical settings.

## **1.3 PROJECT OVERVIEW**

The project on lung disease classification using X-ray images is designed to automate and enhance the process of diagnosing lung diseases. By leveraging machine learning techniques, specifically CNN, ResNet, and InceptionResNet algorithms, we aim to develop a robust system capable of accurately classifying X-ray images into different disease categories. The project follows a modular architecture, starting with the preprocessing of X-ray images, including resizing, normalization, and noise reduction, to standardize the input data. Feature extraction techniques are then applied to capture relevant information from the images, which serves as input to the machine learning models. These models are trained using labeled X-ray images, optimizing parameters and utilizing advanced optimization algorithms. The trained models are then used for the classification of unseen X-ray images, enabling the identification of various lung diseases, such as pneumonia, tuberculosis, and lung cancer. The project's overarching goal is to improve the efficiency and reliability of lung disease diagnosis by reducing human error and providing faster results, ultimately contributing to better patient care and outcomes in the field of healthcare.

## **1.4 OBJECTIVE**

The objectives of our project on lung disease classification using X-ray images are multi-fold. Firstly, we aim to develop a robust machine learning model that can accurately classify X-ray images into different lung disease categories. By leveraging advanced algorithms such as CNN, ResNet, and InceptionResNet, we seek to improve the accuracy of the classification process compared to traditional methods. Secondly, we aim to enhance the efficiency and reliability of lung disease diagnosis by automating the classification process. This would reduce the dependence on manual interpretation by radiologists, minimizing subjective biases and saving valuable time in the diagnostic workflow. Additionally, we strive to evaluate and compare the performance of different machine learning algorithms to determine the most effective approach for lung disease classification. This analysis would enable us to identify the algorithm that yields the highest accuracy and efficiency for the given dataset and problem domain. Lastly, we aim to assess the potential of machine learning in assisting healthcare professionals in diagnosing lung diseases. By providing accurate and timely results, our project aims to augment the expertise of medical practitioners, leading to improved diagnostic accuracy and treatment decisions. Overall, our objectives revolve around enhancing the accuracy,

efficiency, and reliability of lung disease classification using machine learning techniques, ultimately benefiting both healthcare providers and patients.

## **1.5 DATASET**

- The project utilizes a dataset consisting of X-ray images of patients with various lung diseases.
- The dataset includes images sourced from publicly available databases, such as the National Institutes of Health (NIH) Chest X-ray dataset or the Montgomery County X-ray dataset.
- The dataset comprises a significant number of X-ray images, with each image labeled with the corresponding lung disease category.
- Preprocessing techniques, such as data cleaning, normalization, and augmentation, are applied to the dataset to ensure the quality and diversity of the training data.

## **2.LITERATURE SURVEY**

### **2.1 EXISTING SYSTEM**

The existing system for lung disease classification using X-ray images relies on traditional diagnostic methods and statistical data analysis. These approaches involve manual interpretation of X-ray images by radiologists, which can be time-consuming, subjective, and prone to human errors. The limitations of the existing system include the lack of standardization in the interpretation process, leading to inconsistencies in diagnoses. Moreover, relying solely on statistical data for disease classification may not capture the nuanced features present in X-ray images, potentially resulting in reduced accuracy. The need for a more objective, efficient, and accurate system forms the basis for the proposed system. The existing system often faces challenges in terms of scalability and generalizability. Manual interpretation of X-ray images requires skilled radiologists, and their availability might be limited, especially in remote or underserved areas. This can lead to delays in diagnosis and treatment, affecting patient outcomes. Furthermore, the existing system may struggle to handle large volumes of X-ray images efficiently, as it heavily relies on human resources and can be time-consuming. To address these challenges, the proposed system aims to leverage machine learning techniques and automation.

Drawbacks of existing systems:

1. Time-consuming Process: The manual interpretation of X-ray images is a time-consuming process.
2. Limited Availability of Skilled Radiologists
3. Human Errors

### **2.2 PROPOSED SYSTEM**

The proposed system for lung disease classification using X-ray images aims to overcome the limitations of the existing system by leveraging machine learning techniques, specifically CNN, ResNet, and InceptionResNet algorithms.

The system will automate the classification process, reducing the reliance on manual interpretation and the limited availability of skilled radiologists. By training the machine learning models on large and diverse datasets of labeled X-ray images, the system can learn to accurately identify and classify various lung diseases, including pneumonia, tuberculosis, and lung cancer.

Through the proposed system, healthcare providers will be able to efficiently analyze X-ray images and obtain rapid and reliable disease diagnoses. This automation will not only save time but also improve the consistency and objectivity of diagnoses, minimizing the risk of human errors.

Additionally, the proposed system offers the potential for scalability and generalizability. With machine learning algorithms, the system can handle large volumes of X-ray images efficiently, enabling faster diagnoses and reducing the backlog of cases. Moreover, the models can be trained on diverse datasets, allowing for improved generalization across different populations and improving the accuracy of disease classification.

The proposed system has the potential to revolutionize lung disease classification by providing a more accessible, efficient, and accurate approach. By combining the power of machine learning with X-ray image analysis, it offers the opportunity to enhance patient care, enable early detection of diseases, and facilitate timely treatment interventions.

## **2.3 RELATED WORK**

In the field of lung disease classification using machine learning techniques and medical imaging, there have been several notable related works that have contributed to the advancements in this area. These studies have focused on improving the accuracy, efficiency, and automation of disease classification based on X-ray images. Some of the key related works include:

1. "Machine Learning-Based Classification of Lung Diseases on Chest Radiographs" by Wang et al. (2017): This study proposed a deep learning-based approach using convolutional neural networks (CNNs) for the classification of lung diseases on chest radiographs. The researchers achieved high accuracy in identifying common lung diseases, such as pneumonia, tuberculosis, and lung nodules, demonstrating the potential of deep learning techniques in automated disease classification.
2. "Transfer Learning for Improved Lung Disease Classification on Chest X-rays" by Shin et al. (2016): The study explored the effectiveness of transfer learning, where pre-trained deep learning models are fine-tuned for lung disease classification on chest X-ray images. The

researchers demonstrated that transfer learning can significantly enhance the classification performance and reduce the need for large labeled datasets.

These related works have paved the way for the development of the proposed system in our project. They have provided insights into the application of machine learning algorithms, such as CNNs, in lung disease classification, and have showcased the potential of automated and accurate disease detection and classification using medical imaging data. The proposed system builds upon these advancements, aiming to further improve the accuracy, efficiency, and reliability of lung disease classification using X-ray images.

## **3.SYSTEM ANALYSIS**

### **3.1 FUNCTIONAL REQUIREMENTS**

The functional requirements of the lung disease classification system using X-ray images encompass several key aspects. Firstly, the system should support the input of X-ray images in different formats, enabling flexibility and compatibility with various sources. It should also perform preprocessing tasks on the images, such as resizing, normalization, and noise removal, to enhance their quality and prepare them for accurate analysis. The core functionality lies in the disease classification, where machine learning algorithms like CNN, ResNet, and InceptionResNet are employed to extract meaningful features from the X-ray images and make precise predictions regarding the presence and type of lung diseases. The system should prioritize accuracy and reliability, ensuring rigorous testing and validation to achieve high-performance standards. A user-friendly interface should be provided, allowing seamless image upload, visualization of processed images, and access to classification results. Integration with existing healthcare systems or databases may be required for comprehensive analysis and data exchange. Detailed reports and visualizations should be generated to facilitate interpretation and communication of the classification outcomes. Security and privacy measures should be implemented to safeguard patient data, including encryption and access controls. Maintenance and scalability considerations should be taken into account to ensure the system's continuous operation, adaptability to increasing data volumes, and potential future enhancements. Overall, by fulfilling these functional requirements, the lung disease classification system aims to improve accuracy, efficiency, and effectiveness in diagnosing lung diseases based on X-ray images, contributing to enhanced healthcare outcomes and patient care.

### **3.2 PERFORMANCE REQUIREMENTS**

The performance requirements of the lung disease classification system revolve around achieving high accuracy and efficiency in disease classification based on X-ray images. The system should strive to deliver accurate results by correctly identifying the presence and type of lung diseases. This can be measured using evaluation metrics such as precision, recall, and F1 score, which assess the system's ability to correctly classify positive and negative cases. Additionally, the system should be efficient in terms of processing time and resource utilization. It should be capable of handling large volumes of

X-ray images and performing the classification tasks within acceptable time limits. The performance requirements aim to ensure that the system delivers accurate and timely results, enabling healthcare professionals to make informed decisions and provide appropriate treatment to patients.

### **3.3 SOFTWARE REQUIREMENTS**

- Programming Language: Python
- Machine Learning Libraries: TensorFlow, Keras, PyTorch
- Image Processing Libraries: OpenCV
- Database Management System: MySQL, MongoDB
- Google COLAB
- Efficient Computation and Storage for Large Datasets

### **3.4 HARDWARE REQUIREMENTS**

- Processor: Powerful multi-core processor (e.g., Intel Core i5 or higher)
- Memory (RAM): Sufficient capacity (e.g., 8GB or more)
- Storage: Adequate storage capacity (SSD recommended)
- Graphics Processing Unit (GPU): Dedicated GPU for accelerated deep learning (e.g., NVIDIA GeForce, AMD Radeon)
- Display: High-resolution display with accurate color reproduction
- Network Connectivity: Stable internet connection for accessing external resources and system updates

These hardware requirements ensure that the lung disease classification system has the necessary computational resources, memory capacity, storage capacity, and visual display capabilities to handle the image processing, machine learning, and system operation effectively.



### 3.5 FEASIBILITY STUDY

The feasibility study aims to assess the practicality and viability of implementing the lung disease classification system using X-ray images. It helps in identifying technical, economic, legal, ethical, and operational considerations, allowing for informed decision-making and successful project implementation.

#### **Technical Feasibility:**

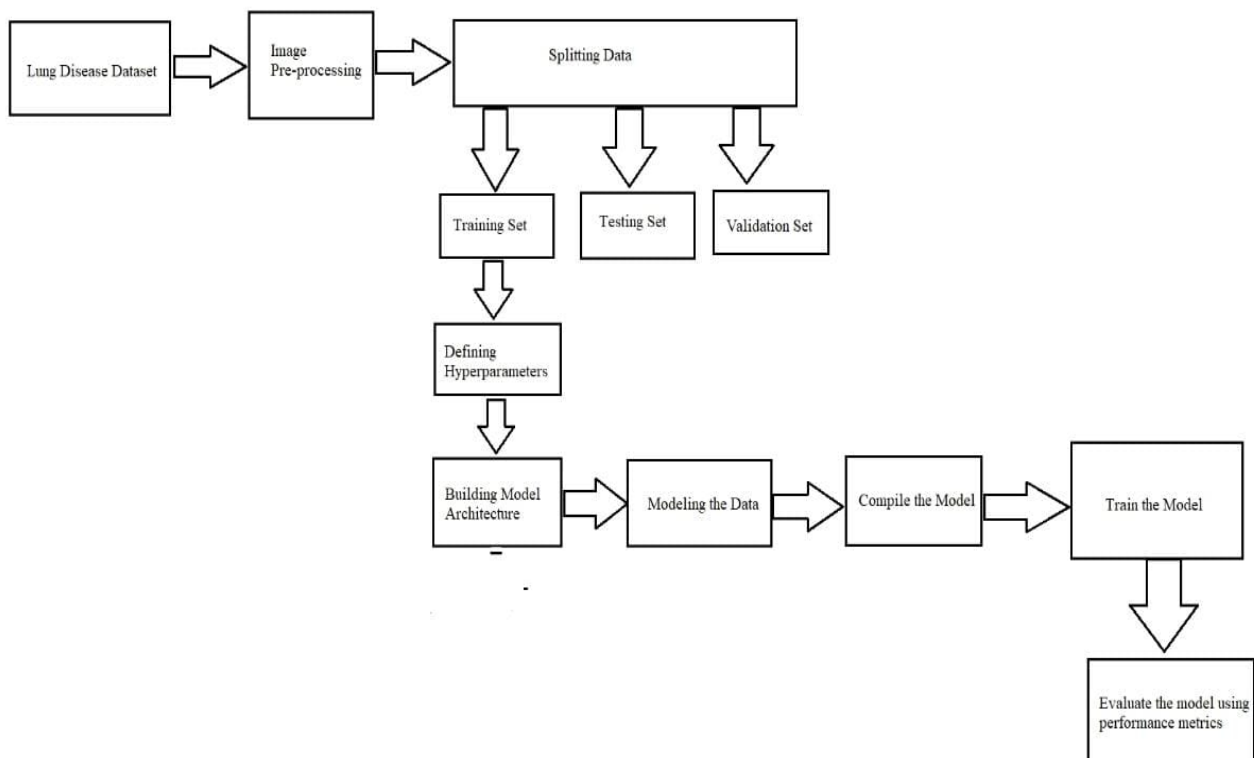
Evaluate the availability and compatibility of image processing tools, machine learning frameworks (such as CNN, ResNet, InceptionResNet), and programming languages suitable for image analysis and classification. Assess the capability to handle and process large volumes of X-ray image data efficiently. Analyze the feasibility of integrating the algorithms (CNN, ResNet, InceptionResNet) into the system architecture.

**Economic Feasibility:** Estimate the costs associated with acquiring the required hardware resources, such as high-performance processors, GPUs, and sufficient storage capacity. Consider the cost of acquiring or creating a labeled X-ray image dataset for training and validation purposes. Evaluate the potential benefits of improved accuracy and efficiency in lung disease diagnosis and its impact on healthcare costs and patient outcomes.

## 4.SYSTEM DESIGN

### 4.1 SYSTEM ARCHITECTURE

- The system design transforms a logical representation of what a given system is required to do into the physical reality during development. Important design factors such as reliability, response time, throughput of the system, maintainability, expandability etc., should be taken into account. Design constraints like cost, hardware limitations, standard compliance etc should also be dealt with. The task of system design is to take the description and associate with it a specific set of facilities-men, machines (computing and other), accommodation, etc., to provide complete specifications of a workable system.
- This new system must provide for all of the essential data processing and it may also do some of those tasks identified during the work of analysis as optional extras. It must work within the imposed constraints and show improvement over the existing system.. At the outset of design a choice must be made between the main approaches. Talks of preliminary design concerned with identification analysis and selections of the major design options are available for development and implementation of a system. These options are most readily distinguished in terms of the physical facilities to be used for the processing who or what does the work.



## **DATABASE COLLECTION:**

A data set is an ordered collection of data. As we know, a collection of information obtained through observations, measurements, study, or analysis is referred to as [data](#). We have taken COVID-19 Chest X-ray images and Lung masks Database

## **IMAGE PREPROCESSING:**

After the collection of data ,it undergoes data pre-processing techniques which means cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task. Perform image processing tasks, such as removing image noise and performing image-to-image translation, using deep neural networks. Neural networks are computing systems designed to recognize patterns. Their architecture is inspired by the human brain structure, hence the name. They consist of three types of layers: input, hidden layers, and output.

## **SPLITTING DATA:**

The simplest way to split the modeling dataset into training, testing and validation sets. It is to allocate around 70-80% of the data to the training set, 10-15% to the validation set, and the remaining 10-15% to the testing set. These percentages can be adjusted based on factors such as the size of the dataset, the complexity of the problem, and the availability of labeled data. Therefore, we train the model using the training set and then apply the model to the test set. In this way, we can evaluate the performance of our model.

## **DEFINING HYPERPARAMETERS:**

These parameters define the behavior and characteristics of the model and can significantly impact its performance. They are typically set based on prior knowledge, heuristics, or through a process of trial and error. Batch size is one of the hyperparameters which determines the number of samples processed in each training iteration. A larger batch size can lead to faster training but may require more memory. A smaller batch size can provide more accurate gradient updates but can increase the training time. The goal is to find the combination of hyperparameter values that results in the best model performance on a validation set.

## **CNN ALGORITHM:**

Convolutional Neural Networks (CNNs) are a powerful deep learning algorithm commonly used for lung disease classification using X-ray images.

CNNs are specifically designed to effectively extract relevant features from images and learn complex patterns automatically. By leveraging the hierarchical nature of CNNs, these algorithms have demonstrated strong performance in lung disease classification tasks, providing accurate and automated diagnosis assistance to medical professionals.

## **INCEPTION RESNET ALGORITHM:**

The InceptionResNet algorithm is a deep convolutional neural network architecture that combines the principles of the Inception module and the ResNet architecture. InceptionResNet incorporates the concept of "Inception" from the Inception module, which uses multiple parallel convolutional branches with different filter sizes to capture features at various scales. This allows the network to learn both local and global features efficiently. By combining the Inception module and Residual connections, InceptionResNet achieves improved performance and training efficiency compared to previous architectures. It can capture rich and diverse features while facilitating the training of deep networks.

## **EFFICIENT ALGORITHM:**

The EfficientNet algorithm is a state-of-the-art deep learning architecture that aims to achieve high performance while maintaining computational efficiency. It was developed with the goal of balancing model size and accuracy by scaling the network architecture in a principled manner. EfficientNet introduces a compound scaling method that uniformly scales the depth, width, and resolution of the network. This approach allows the model to efficiently use computational resources while achieving improved performance. The scaling is controlled by a single scaling coefficient that determines the network's overall size. EfficientNet has become a popular choice for many computer vision applications due to its impressive performance, efficient architecture, and scalability across different computational resources.

## TRAINING THE MODEL

Training a model in lung disease classification using X-ray images involves the process of teaching the model like CNN, Inception ResNet and EfficientNet to learn patterns and features from the training dataset so that it can accurately classify X-ray images into different lung disease categories.

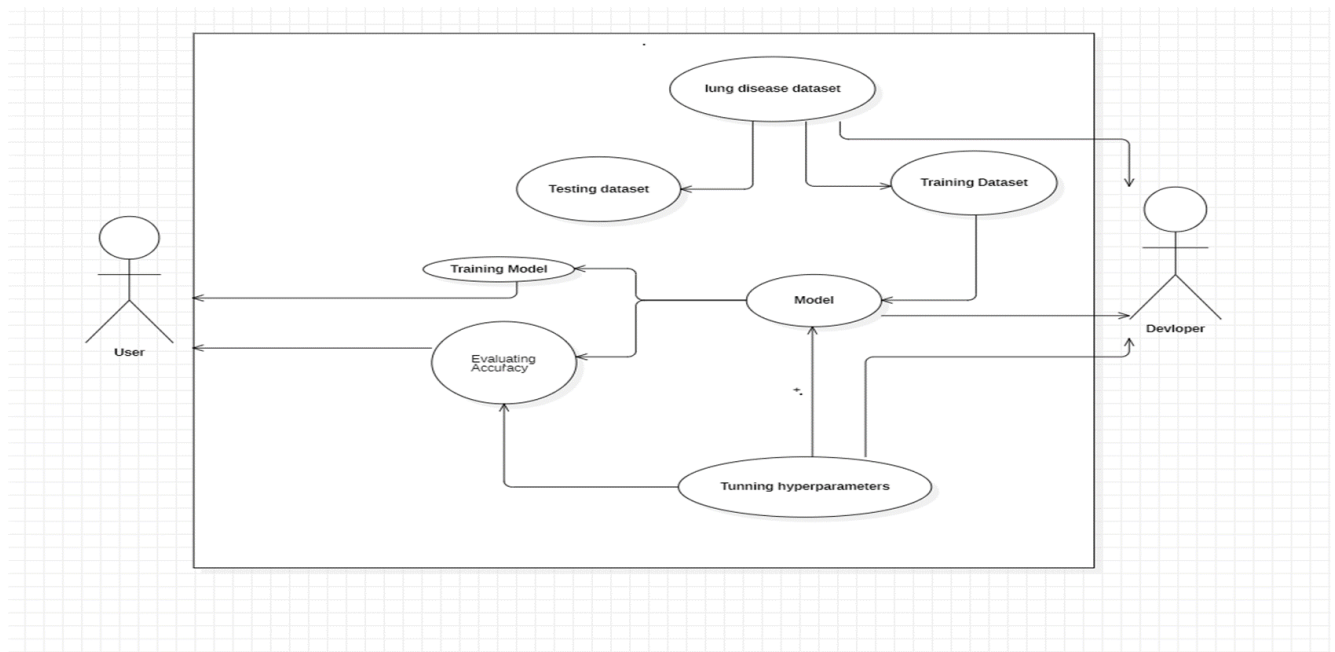
## EVALUATING THE MODEL

Evaluating the model in lung disease classification using X-ray images involves assessing its performance and effectiveness in accurately classifying the images into different lung disease categories. The evaluation process helps determine how well the model has learned from the training data and how it performs on unseen data.

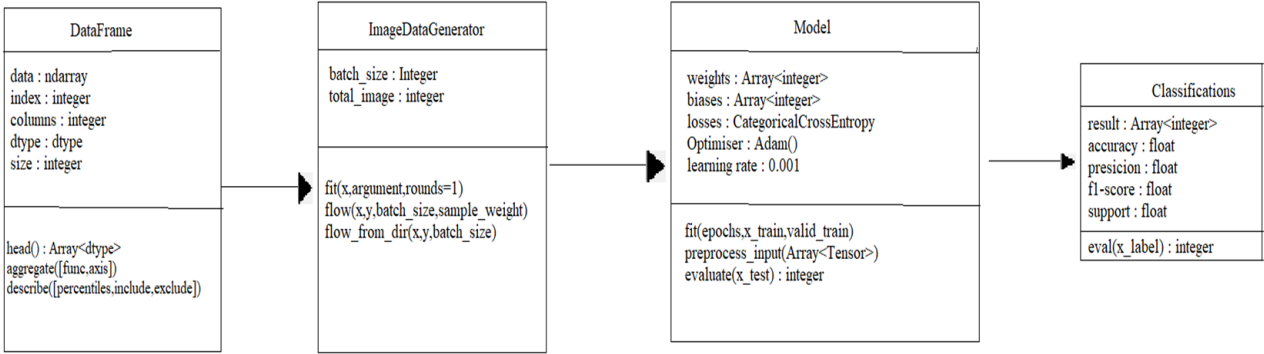
By evaluating the model using performance metrics, accuracy, f1 score, precision. This evaluation process helps determine the model's readiness for deployment and provides insights for potential improvements.

## 4.2 UML DIAGRAMS

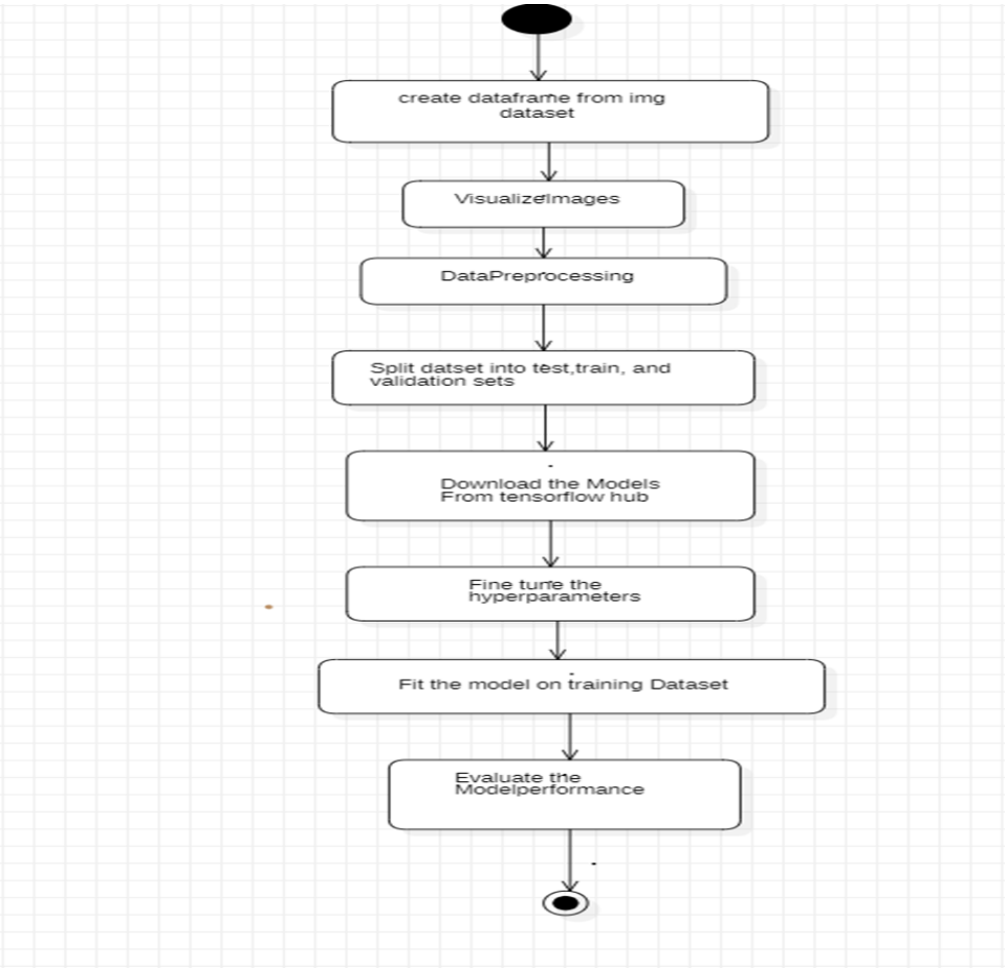
### 4.2.1 USE CASE DIAGRAM



4.2.2 CLASS DIAGRAM



4.2.3 ACTIVITY DIAGRAM



## **5.IMPLEMENTATION AND RESULTS**

### **5.1 ALGORITHMS USED**

#### **5.1.1 CNN**

The Convolutional Neural Network (CNN) algorithm is a deep learning algorithm primarily used for analyzing visual data, such as images and videos. It is inspired by the structure and functioning of the human visual system and has proven to be highly effective in various computer vision tasks. CNNs are designed to automatically learn and extract features from input data. They consist of multiple interconnected layers, each responsible for performing specific operations on the input. The key idea behind CNNs is to capture local patterns and hierarchically learn complex representations of the input.

CNN is another type of neural network that can uncover key information in both time series and image data. For this reason, it is highly valuable for image-related tasks, such as image recognition, object classification and pattern recognition. To identify patterns within an image, a CNN leverages principles from linear algebra, such as matrix multiplication. CNNs can also classify audio and signal data. A CNN's architecture is analogous to the connectivity pattern of the human brain. Just like the brain consists of billions of neurons, CNNs also have neurons arranged in a specific way. In fact, CNN's neurons are arranged like the brain's frontal lobe, the area responsible for processing visual stimuli. This arrangement ensures that the entire visual field is covered, thus avoiding the piecemeal image processing problem of traditional neural networks, which must be fed images in reduced-resolution pieces. Compared to the older networks, a CNN delivers better performance with image inputs, and also with speech or audio signal inputs.

##### **5.1.1.1 WORKING OF CNN**

The working of a Convolutional Neural Network (CNN) involves several steps, from receiving the input data to generating the final output. Here's a high-level overview of the working of a CNN:

1. **Input Data:** CNNs are primarily used for processing visual data, such as images. The input data consists of one or more images represented as matrices of pixel values. Each image typically has multiple channels (e.g., RGB images have three channels for red, green, and blue).

2. **Convolutional Layers:** The input data is passed through one or more convolutional layers. Each convolutional layer applies a set of learnable filters to the input. The filters convolve over the input using element-wise multiplication and summation, producing feature maps. Each filter is responsible for detecting specific features or patterns in the input.
3. **Activation Function:** After each convolutional layer, a non-linear activation function, such as ReLU (Rectified Linear Unit), is applied element-wise to the feature maps. Activation functions introduce non-linearities, allowing the model to capture complex relationships in the data.
4. **Pooling Layers:** Following the activation function, pooling layers are often used to reduce the spatial dimensions of the feature maps while retaining the most important information. The most common pooling operation is max pooling, which selects the maximum value within a sliding window and discards the rest.
5. **Fully Connected Layers:** Towards the end of the CNN architecture, one or more fully connected layers are added. These layers take the high-level features learned by the previous layers and map them to the desired output. Each neuron in a fully connected layer is connected to every neuron in the previous layer, similar to traditional neural networks.
6. **Output Layer:** The final fully connected layer is connected to an output layer, which produces the desired output based on the task at hand. For example, in image classification, the output layer might have neurons corresponding to different classes, and the network's prediction is determined by the neuron with the highest activation.
7. **Loss Function and Training:** The predicted output from the CNN is compared with the true label using a loss function. The CNN is trained using the backpropagation algorithm, which calculates the gradients of the model parameters with respect to the loss function.
8. **Inference:** Once the CNN is trained, it can be used for inference on new, unseen data. The input data is fed into the trained network, and the output is generated based on the learned representations and weights.

By stacking convolutional, pooling, and fully connected layers together, CNNs can automatically learn hierarchical representations of visual data. The earlier layers capture low-level features like edges and textures, while the deeper layers learn high-level features and semantic information, enabling the network to make accurate predictions or classifications.



### **5.1.1.2 ADVANTAGES OF CNN**

1. Efficient image processing – One of the key advantages of CNNs is their ability to process images efficiently. This is because they use a technique called convolution, which involves applying a filter to an image to extract features that are relevant to the task at hand.
2. High accuracy rates – Another advantage of CNNs is their ability to achieve high accuracy rates. This is because they can learn to recognize complex patterns in images by analyzing large datasets. This means that they can be trained to recognize specific objects or features with a high degree of accuracy, which makes them ideal for tasks like facial recognition or object detection.
3. Robust to noise – CNNs are also robust to noise, which means that they can still recognize patterns in images even if they are distorted or corrupted. This is because they use multiple layers of filters to extract features from images, which makes them more resilient to noise than other types of algorithms.
4. Transfer learning – CNNs also support transfer learning, which means that they can be trained on one task and then used to perform another task with little or no additional training. This is because the features that are extracted by CNNs are often generic enough to be used for a wide range of tasks, which makes them a versatile tool for many different applications.
5. Automated feature extraction – Finally, CNNs automate the feature extraction process, which means that they can learn to recognize patterns in images without the need for manual feature engineering.

### **5.1.2 INCEPTION RESNET**

The Inception-ResNet algorithm is a deep learning architecture that combines the ideas from two influential models: Inception and ResNet. It was developed to improve the performance and efficiency of convolutional neural networks (CNNs) in various computer vision tasks, such as image classification and object detection.

The Inception architecture, introduced in the original Inception paper by Szegedy et al., is known for its ability to capture multi-scale features by employing parallel convolutional operations with different filter sizes. This parallel operation helps the model to effectively learn both local and

global features, leading to improved performance. However, deeper versions of the Inception architecture suffer from vanishing gradients and increased computational complexity.

On the other hand, the ResNet (Residual Network) architecture, proposed by He et al., addresses the vanishing gradient problem in deep networks. ResNet introduces skip connections, also known as residual connections, which allow the network to learn residual mappings. These connections enable the gradient to flow directly from earlier layers to later layers, making it easier to train very deep networks. ResNet achieved outstanding performance by effectively handling the optimization challenges of deep networks.

Inception-ResNet has been widely used in different domains, including image classification, object detection, and image segmentation. Its effectiveness in handling complex visual data and its ability to achieve high accuracy have made it a popular choice for many computer vision applications.

### **5.1.2.1 WORKING OF INCEPTION RESNET**

The working of the Inception-ResNet algorithm involves several key components and steps.

1. **Input Data:** The algorithm takes input data in the form of images or feature maps. The input is typically a multi-channel image representation, such as an RGB image.
2. **Inception Modules:** The core building blocks of the Inception-ResNet algorithm are the Inception modules. Each Inception module consists of parallel branches of convolutional operations with different filter sizes. The outputs of these branches are concatenated to create a rich set of features capturing both local and global information.
3. **Residual Connections:** In addition to the parallel branches, Inception-ResNet introduces residual connections within the Inception modules. These connections allow the model to learn residual mappings by directly passing the input from an earlier layer to a later layer.
4. **Stacking of Modules:** Multiple Inception modules are stacked on top of each other to create a deep neural network architecture. The stacking of modules allows for the hierarchical learning of features, with each module capturing increasingly complex representations.
5. **Classification or Task-specific Layers:** Towards the end of the Inception-ResNet architecture, one or more fully connected layers or task-specific layers are added. These layers map the high-level features learned by the earlier layers to the desired output.

6. **Training:** Inception-ResNet is typically trained using labeled data in a supervised manner. The model's parameters, including the weights and biases of the convolutional layers and fully connected layers, are optimized during the training process. This optimization is achieved by minimizing a loss function that measures the discrepancy between the predicted output and the true label.
7. **Inference:** Once trained, the Inception-ResNet algorithm can be used for inference on new, unseen data. The input data is fed into the trained network, and the output is generated based on the learned representations and weights.

By combining the multi-scale feature extraction capabilities of the Inception architecture and the gradient flow optimization of the ResNet architecture, the Inception-ResNet algorithm achieves high accuracy and efficiency in various computer vision tasks.

#### **5.1.2.2 ADVANTAGES OF INCEPTION RESNET**

The Inception-ResNet algorithm offers several advantages that contribute to its effectiveness in computer vision tasks. Here are some key advantages of the Inception-ResNet algorithm:

1. **Improved Accuracy:** The Inception-ResNet algorithm achieves state-of-the-art accuracy on various computer vision benchmarks. By combining the strengths of both the Inception and ResNet architectures, it captures multi-scale features and effectively handles optimization challenges, leading to improved performance.
2. **Efficient Parameter Usage:** The Inception-ResNet algorithm efficiently utilizes network parameters. The Inception modules employ parallel convolutional operations with different filter sizes, enabling the model to capture features at different scales without dramatically increasing the number of parameters.
3. **Scalability:** The Inception-ResNet algorithm is highly scalable. It can be adapted to different model depths by stacking multiple Inception modules. This scalability allows the algorithm to handle varying levels of complexity and adapt to the specific requirements of different tasks or datasets.
4. **Robust Feature Extraction:** The parallel branches of convolutional operations within the Inception-ResNet algorithm capture diverse and complementary features. This multi-scale feature extraction capability enables the model to effectively represent both local and global information present in the input data.

### 5.1.3 EFFICIENT NET

EfficientNet is a deep learning architecture that aims to achieve state-of-the-art performance while being computationally efficient. It was introduced by Tan and Le in their paper titled "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." The algorithm addresses the challenge of scaling neural network models to achieve better accuracy without significantly increasing computational cost.

The EfficientNet algorithm introduces a novel compound scaling method that uniformly scales the depth, width, and resolution of the network. By carefully balancing these dimensions, EfficientNet achieves improved performance across various computer vision tasks, including image classification and object detection.

EfficientNet addresses the challenge of designing models that are both accurate and efficient. Typically, increasing the model size or depth improves accuracy but comes at the cost of increased computational resources. On the other hand, reducing the model size may compromise accuracy. EfficientNet seeks to find the optimal trade-off by scaling the network's dimensions in a principled manner.

#### 5.1.3.1 WORKING OF EFFICIENT NET

The working of the EfficientNet algorithm involves several key steps and components. Here's an overview of how it operates:

1. **Scaling Coefficients:** The EfficientNet algorithm introduces scaling coefficients to uniformly scale the depth, width, and resolution of the network. These coefficients determine the size of each dimension and play a crucial role in achieving a balance between accuracy and computational efficiency.
2. **Base Architecture:** EfficientNet starts with a base architecture, typically composed of efficient building blocks like MobileNetV2 or MBConv. These building blocks are lightweight and computationally efficient, making them suitable for resource-constrained environments.
3. **Compound Scaling:** EfficientNet applies compound scaling to the base architecture by uniformly scaling the depth, width, and resolution. The scaling coefficients determine the size

of each dimension, allowing the network to be efficiently scaled up or down. The compound scaling method ensures that all dimensions are adjusted proportionally, maintaining a balance between model capacity and computational cost.

4. **Neural Architecture Search (NAS):** EfficientNet utilizes neural architecture search techniques to automatically search for the optimal architecture within the scaled dimensions. The goal is to find the architecture that achieves the best trade-off between accuracy and computational cost. The architecture search process explores different combinations of building blocks, layer depths, widths, and resolutions to find the most effective configuration.
5. **Depth Scaling:** EfficientNet scales the depth of the network by repeating the building blocks multiple times. Deeper networks capture more complex features and representations but come at the cost of increased computational complexity. The scaling coefficient for depth determines the optimal number of repetitions for the building blocks.
6. **Width Scaling:** EfficientNet scales the width of the network by adjusting the number of channels or filters in each layer. Wider networks capture more diverse and richer features, leading to improved performance. The scaling coefficient for width determines the optimal number of channels for the convolutional layers.
7. **Resolution Scaling:** EfficientNet scales the resolution of the network by resizing the input images. Higher-resolution inputs provide more detailed information, enabling the model to capture fine-grained features. The scaling coefficient for resolution determines the optimal input image size.
8. **Training and Fine-tuning:** EfficientNet is typically trained using labeled data in a supervised manner. The model's parameters, including the weights and biases of the convolutional layers and fully connected layers, are optimized during the training process. The training involves feeding the input data through the network, calculating the loss, and updating the model's parameters using gradient-based optimization algorithms like stochastic gradient descent.
9. **Inference:** Once trained, the EfficientNet algorithm can be used for inference on new, unseen data. The input data is fed into the trained network, and the output is generated based on the learned representations and weights.

By leveraging compound scaling, efficient building blocks, and neural architecture search, the EfficientNet algorithm achieves state-of-the-art accuracy while being more computationally efficient compared to previous models.

### 5.1.3.2 ADVANTAGES OF EFFICIENT NET

The EfficientNet algorithm offers several advantages that contribute to its effectiveness and popularity in the field of computer vision. Here are some key advantages of the EfficientNet algorithm:

1. **Improved Accuracy:** EfficientNet achieves state-of-the-art accuracy on various computer vision tasks, including image classification and object detection. By employing compound scaling and efficient building blocks, it effectively captures and represents complex features in the data, leading to higher accuracy compared to previous models.
2. **Computational Efficiency:** One of the significant advantages of EfficientNet is its computational efficiency. The compound scaling method allows for fine-grained control over the model's depth, width, and resolution, ensuring a balance between accuracy and computational cost. This makes EfficientNet suitable for deployment on resource-constrained devices and accelerates inference time.
3. **Scalability:** EfficientNet offers scalability by providing different model variants labeled as EfficientNet-B0, EfficientNet-B1, EfficientNet-B2, and so on. These variants allow users to choose the appropriate model size based on their computational resources and accuracy requirements. The scalability of EfficientNet makes it adaptable to a wide range of applications and computing environments.
4. **Broad Applicability:** EfficientNet is versatile and can be applied to various computer vision tasks, including image classification, object detection, semantic segmentation, and more. Its state-of-the-art performance and efficiency make it applicable in a wide range of domains, such as healthcare, autonomous driving, retail, and surveillance.

Overall, the advantages of the EfficientNet algorithm, including improved accuracy, computational efficiency, scalability, transfer learning capabilities, broad applicability, and the use of lightweight building blocks, make it a highly effective and popular choice for various computer vision tasks. Its performance and efficiency contribute to advancements in the field and facilitate the development of practical and impactful computer vision applications.

## 5.2 MODULES

**Input:** X-Ray Images

**Output:** Classification of lung diseases

**Module 1 :** Import the required libraries

**Module 2 :** Explore the data to figure out what they look like

**Module 3 :** Pre-processing the images using data generators.

**Module 4 :** Splitting the data into a training and testing set.

**Module 5 :** Defining Hyperparameters

**Module 6 :** Building the model architecture.

**Module 7 :** Modeling the data using three algorithms.

**Module 8 :** Compiling the model

**Module 9 :** Training the model

**Module 10 :** Evaluate the results of the algorithm using performance metrics

## 5.3 SAMPLE CODE

### 5.3.1 Code for CNN model

#### MODULE 1 : DATA COLLECTION AND IMPORT DATA SET

```
+ Code + Text
Skipping, found downloaded files in "../covid19-radiography-dataset" (use force=True to force download)

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
from tensorflow.keras import keras
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Dense, Activation, Dropout, Conv2D, MaxPooling2D, BatchNormalization, Flatten
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras import regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model, load_model, Sequential
import numpy as np
import pandas as pd
import shutil
import time
import cv2 as cv2
from tqdm import tqdm
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
import os
import seaborn as sns
sns.set_style('darkgrid')
from PIL import Image
from sklearn.metrics import confusion_matrix, classification_report
from IPython.core.display import display, HTML
```

```

+ Code + Text

!pip install opendatasets
import opendatasets as od
od.download('https://www.kaggle.com/datasets/preetviradiya/covid19-radiography-dataset')

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: opendatasets in /usr/local/lib/python3.10/dist-packages (0.1.22)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from opendatasets) (4.65.0)
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (from opendatasets) (1.5.13)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from opendatasets) (8.1.3)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2022.12.7)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.27.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (8.0.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (1.26.15)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle->opendatasets) (1.3)
Requirement already satisfied: charset-normalizer~2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatasets) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatasets) (3.4)
Skipping, found downloaded files in "./covid19-radiography-dataset" (use force=True to force download)

```

## MODULE 2 : EXPLORE THE DATA TO FIGURE OUT WHAT THEY LOOK LIKE

```

+ Code + Text

[ ] sdir=r'covid19-radiography-dataset/COVID-19_Radiography_Dataset/COVID-19_Radiography_Dataset'

filepaths=[]
labels=[]
classlist=os.listdir(sdir)
for klass in classlist:
    classpath=os.path.join(sdir,klass)
    if os.path.isdir(classpath):
        flist=os.listdir(classpath)
        for f in flist:
            fpath=os.path.join(classpath,f)
            filepaths.append(fpath)
            labels.append(klass)
Fseries= pd.Series(filepaths, name='filepaths')
Lseries=pd.Series(labels, name='labels')
df=pd.concat([Fseries, Lseries], axis=1)
print (df.head())
print (df['labels'].value_counts())

      filepaths      labels
0 covid19-radiography-dataset/COVID-19_Radiograp...  Viral Pneumonia
1 covid19-radiography-dataset/COVID-19_Radiograp...  Viral Pneumonia
2 covid19-radiography-dataset/COVID-19_Radiograp...  Viral Pneumonia
3 covid19-radiography-dataset/COVID-19_Radiograp...  Viral Pneumonia
4 covid19-radiography-dataset/COVID-19_Radiograp...  Viral Pneumonia
Normal      10192
Lung Opacity      6012
COVID      3616
Viral Pneumonia      1345
Name: labels, dtype: int64

```

## MODULE 3 : SPLITTING THE DATA INTO A TRAINING AND TESTING SET.

```

+ Code + Text

[ ] train_split=.9
test_split=.05
dummy_split=test_split/(1-train_split)
train_df, dummy_df=train_test_split(df, train_size=train_split, shuffle=True, random_state=123)
test_df, valid_df=train_test_split(dummy_df, train_size=dummy_split, shuffle=True, random_state=123)
print ('train_df length: ', len(train_df), ' test_df length: ', len(test_df), ' valid_df length: ', len(valid_df))

train_df length: 19048 test_df length: 1058 valid_df length: 1059

```



## MODULE 4 : PRE-PROCESSING THE IMAGES USING DATA GENERATORS.

```
+ Code + Text

[ ] height=128
width=128
channels=3
batch_size=40
img_shape=(height, width, channels)
img_size=(height, width)
length=len(test_df)
test_batch_size=sorted([int(length/n) for n in range(1,length+1) if length % n ==0 and length/n<=80],reverse=True)[0]
test_steps=int(length/test_batch_size)
print ( 'test batch size: ',test_batch_size, ' test steps: ', test_steps)
def scalar(img):
    return img/127.5-1 # scale pixel between -1 and +1
gen=ImageDataGenerator(preprocessing_function=scalar)
train_gen=gen.flow_from_dataframe( train_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
                                color_mode='rgb', shuffle=True, batch_size=batch_size)
test_gen=gen.flow_from_dataframe( test_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
                                color_mode='rgb', shuffle=False, batch_size=test_batch_size)
valid_gen=gen.flow_from_dataframe( valid_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
                                color_mode='rgb', shuffle=True, batch_size=batch_size)
classes=list(train_gen.class_indices.keys())
print (classes)
class_count=len(classes)
train_steps=int(len(train_gen.labels)/batch_size)

test batch size: 46 test steps: 23
Found 19048 validated image filenames belonging to 4 classes.
Found 1058 validated image filenames belonging to 4 classes.
Found 1059 validated image filenames belonging to 4 classes.
['COVID', 'Lung_Opacity', 'Normal', 'Viral_Pneumonia']
```

## MODULE 5 : DEFINING HYPERPARAMETERS

## MODULE 6 : BUILDING THE MODE ARCHITECTURE.

## MODULE 7 : COMPILING THE MODEL

```
+ Code + Text

[ ] # Define hyperparameters
input_shape = (128, 128, 3)
batch_size = 40
epochs = 10
num_classes = 4

[ ] # Define the model architecture
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=input_shape),
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2,2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])

[ ] # Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

## MODULE 8 : TRAINING THE MODEL

```
+ Code + Text

[ ] # Train the model
model.fit(train_gen, epochs=epochs, validation_data=valid_gen)

Epoch 1/10
477/477 [=====] - 57s 110ms/step - loss: 0.6705 - accuracy: 0.7370 - val_loss: 0.5162 - val_accuracy: 0.7998
Epoch 2/10
477/477 [=====] - 49s 104ms/step - loss: 0.4670 - accuracy: 0.8266 - val_loss: 0.3747 - val_accuracy: 0.8602
Epoch 3/10
477/477 [=====] - 49s 102ms/step - loss: 0.3793 - accuracy: 0.8611 - val_loss: 0.3356 - val_accuracy: 0.8659
Epoch 4/10
477/477 [=====] - 49s 103ms/step - loss: 0.3240 - accuracy: 0.8817 - val_loss: 0.3286 - val_accuracy: 0.8754
Epoch 5/10
477/477 [=====] - 50s 104ms/step - loss: 0.2835 - accuracy: 0.8943 - val_loss: 0.3016 - val_accuracy: 0.8905
Epoch 6/10
477/477 [=====] - 54s 112ms/step - loss: 0.2568 - accuracy: 0.9027 - val_loss: 0.2853 - val_accuracy: 0.8942
Epoch 7/10
477/477 [=====] - 49s 103ms/step - loss: 0.2266 - accuracy: 0.9163 - val_loss: 0.3091 - val_accuracy: 0.8876
Epoch 8/10
477/477 [=====] - 49s 103ms/step - loss: 0.2000 - accuracy: 0.9241 - val_loss: 0.3189 - val_accuracy: 0.8980
Epoch 9/10
477/477 [=====] - 51s 108ms/step - loss: 0.1764 - accuracy: 0.9336 - val_loss: 0.3543 - val_accuracy: 0.8933
Epoch 10/10
477/477 [=====] - 49s 102ms/step - loss: 0.1599 - accuracy: 0.9393 - val_loss: 0.3661 - val_accuracy: 0.8924
<keras.callbacks.History at 0x7fae936a8d00>
```

## MODULE 9 : EVALUATE THE MODEL ON TEST SET

```
+ Code + Text

[ ] # Evaluate the model on the test set
test_loss, test_acc = model.evaluate(test_gen)
print('Test accuracy:', test_acc)

23/23 [=====] - 3s 124ms/step - loss: 0.3938 - accuracy: 0.8922
Test accuracy: 0.8922495245933533
```

### 5.3.2 Code for InceptionResNet model

## MODULE 1 : DATA COLLECTION AND IMPORT DATA SET

```
+ Code + Text

# Import required libraries
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Dense, Activation, Dropout, Conv2D, MaxPooling2D, BatchNormalization, Flatten
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras import regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model, load_model, Sequential
import numpy as np
import pandas as pd
import shutil
import time
import cv2 as cv2
from tqdm import tqdm
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
import os
import seaborn as sns
sns.set_style('darkgrid')
from PIL import Image
from sklearn.metrics import confusion_matrix, classification_report
from IPython.core.display import display, HTML
```

```

+ Code + Text

[ ] !pip install opendatasets
import opendatasets as od
od.download('https://www.kaggle.com/datasets/preetviradiya/covid19-radiography-dataset')

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting opendatasets
  Downloading opendatasets-0.1.22-py3-none-any.whl (15 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from opendatasets) (4.65.0)
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (from opendatasets) (1.5.13)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from opendatasets) (8.1.3)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2022.12.7)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.27.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (8.0.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (1.26.15)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle->opendatasets) (1.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatasets) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatasets) (3.4)
Installing collected packages: opendatasets
Successfully installed opendatasets-0.1.22
Please provide your Kaggle credentials to download this dataset. Learn more: http://bit.ly/kaggle-creds
Your Kaggle username: shivanikeerthigari
Your Kaggle Key: .....
Downloading covid19-radiography-dataset.zip to ./covid19-radiography-dataset
100% |██████████| 747M/747M [00:07<00:00, 105MB/s]

```

## MODULE 2 : EXPLORE THE DATA TO FIGURE OUT WHAT THEY LOOK LIKE

```

+ Code + Text

[ ] sdir=r'covid19-radiography-dataset/COVID-19_Radiography_Dataset/COVID-19_Radiography_Dataset'

filepaths=[]
labels=[]
classlist=os.listdir(sdir)
for klass in classlist:
    classpath=os.path.join(sdir,klass)
    if os.path.isdir(classpath):
        flist=os.listdir(classpath)
        for f in flist:
            fpath=os.path.join(classpath,f)
            filepaths.append(fpath)
            labels.append(klass)
Fseries= pd.Series(filepaths, name='filepaths')
Lseries=pd.Series(labels, name='labels')
df=pd.concat([Fseries, Lseries], axis=1)
print (df.head())
print (df['labels'].value_counts())

```

	filepaths	labels
0	covid19-radiography-dataset/COVID-19_Radiograp...	Viral Pneumonia
1	covid19-radiography-dataset/COVID-19_Radiograp...	Viral Pneumonia
2	covid19-radiography-dataset/COVID-19_Radiograp...	Viral Pneumonia
3	covid19-radiography-dataset/COVID-19_Radiograp...	Viral Pneumonia
4	covid19-radiography-dataset/COVID-19_Radiograp...	Viral Pneumonia
	Normal	10192
	Lung Opacity	6012
	COVID	3616
	Viral Pneumonia	1345
	Name: labels, dtype: int64	

## MODULE 3 : SPLITTING THE DATA INTO A TRAINING AND TESTING SET.

```

+ Code + Text

[ ] train_split=.9
test_split=.05
dummy_split=test_split/(1-train_split)
train_df, dummy_df=train_test_split(df, train_size=train_split, shuffle=True, random_state=123)
test_df, valid_df=train_test_split(dummy_df, train_size=dummy_split, shuffle=True, random_state=123)
print ('train_df length: ', len(train_df), ' test_df length: ', len(test_df), ' valid_df length: ', len(valid_df))

```

train\_df length: 19048    test\_df length: 1058    valid\_df length: 1059

## MODULE 4 : PREPROCESSING THE IMAGES USING DATA GENERATORS

## MODULE 5 : DEFINING HYPERPARAMETERS

```
+ Code + Text

height=128
width=128
channels=3
batch_size=40
img_shape=(height, width, channels)
img_size=(height, width)
length=len(test_df)
test_batch_size=sorted([int(length/n) for n in range(1,length+1) if length % n ==0 and length/n<=80],reverse=True)[0]
test_steps=int(length/test_batch_size)
print ( 'test batch size: ',test_batch_size, ' test steps: ', test_steps)
def scalar(img):
    return img/127.5-1 # scale pixel between -1 and +1
gen=ImageDataGenerator(preprocessing_function=scalar)
train_gen=gen.flow_from_dataframe( train_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
    color_mode='rgb', shuffle=True, batch_size=batch_size)
test_gen=gen.flow_from_dataframe( test_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
    color_mode='rgb', shuffle=False, batch_size=test_batch_size)
valid_gen=gen.flow_from_dataframe( valid_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
    color_mode='rgb', shuffle=True, batch_size=batch_size)

classes=list(train_gen.class_indices.keys())
print (classes)
class_count=len(classes)
train_steps=int(len(train_gen.labels)/batch_size)

test batch size: 46 test steps: 23
Found 19048 validated image filenames belonging to 4 classes.
Found 1058 validated image filenames belonging to 4 classes.
Found 1059 validated image filenames belonging to 4 classes.
['COVID', 'Lung Opacity', 'Normal', 'Viral Pneumonia']

[ ] input_shape = (128, 128, 3)
    batch_size = 40
    epochs = 10
    num_classes = 4
```

## MODULE 6 : BUILDING THE MODEL ARCHITECTURE.

## MODULE 7 : COMPILING THE MODEL

```
+ Code + Text

[ ] # Build the model architecture
    model = tf.keras.models.Sequential([
        inception_resnet_v2,
        tf.keras.layers.GlobalAveragePooling2D(),
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(num_classes, activation='softmax')
    ])

[ ] # Compile the model
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

## MODULE 8 : TRAINING THE MODEL

## MODULE 9 : EVALUATING THE MODEL

```
+ Code + Text

[ ] # Train the model
model.fit(train_gen, epochs=epochs, validation_data=valid_gen)

Epoch 1/10
477/477 [=====] - 81s 123ms/step - loss: 0.7957 - accuracy: 0.7202 - val_loss: 0.5332 - val_accuracy: 0.8055
Epoch 2/10
477/477 [=====] - 55s 115ms/step - loss: 0.6118 - accuracy: 0.7665 - val_loss: 0.4749 - val_accuracy: 0.8253
Epoch 3/10
477/477 [=====] - 57s 118ms/step - loss: 0.5884 - accuracy: 0.7764 - val_loss: 0.4608 - val_accuracy: 0.8263
Epoch 4/10
477/477 [=====] - 54s 114ms/step - loss: 0.5661 - accuracy: 0.7828 - val_loss: 0.4666 - val_accuracy: 0.8366
Epoch 5/10
477/477 [=====] - 56s 116ms/step - loss: 0.5532 - accuracy: 0.7902 - val_loss: 0.4539 - val_accuracy: 0.8357
Epoch 6/10
477/477 [=====] - 53s 111ms/step - loss: 0.5494 - accuracy: 0.7931 - val_loss: 0.4562 - val_accuracy: 0.8281
Epoch 7/10
477/477 [=====] - 53s 112ms/step - loss: 0.5397 - accuracy: 0.7964 - val_loss: 0.4406 - val_accuracy: 0.8357
Epoch 8/10
477/477 [=====] - 54s 112ms/step - loss: 0.5124 - accuracy: 0.8058 - val_loss: 0.4601 - val_accuracy: 0.8338
Epoch 9/10
477/477 [=====] - 54s 112ms/step - loss: 0.5118 - accuracy: 0.8057 - val_loss: 0.4435 - val_accuracy: 0.8395
Epoch 10/10
477/477 [=====] - 53s 112ms/step - loss: 0.5084 - accuracy: 0.8068 - val_loss: 0.4584 - val_accuracy: 0.8300
<keras.callbacks.History at 0x7f39d01bad40>

[ ] # Evaluate the model on the test set
test_loss, test_acc = model.evaluate(test_gen)
print('Test accuracy:', test_acc)

23/23 [=====] - 5s 140ms/step - loss: 0.4906 - accuracy: 0.8233
Test accuracy: 0.8232514262199402
```

### 5.3.3 Code for EfficientNet model

## MODULE 1 : DATA COLLECTION AND IMPORT DATA SET

```
EfficientNet - Lung disease prediction.ipynb ☆
File Edit View Insert Runtime Tools Help Last edited on May 10

+ Code + Text

[ ] # Import Data Science Libraries
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
import itertools
import random

# Import visualization libraries
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import cv2

# Tensorflow Libraries
from tensorflow import keras
from tensorflow.keras import layers, models
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import Callback, EarlyStopping, ModelCheckpoint
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras import Model
from tensorflow.keras.layers.experimental import preprocessing

# System libraries
from pathlib import Path
import os.path

# Metrics
from sklearn.metrics import classification_report, confusion_matrix
```

```

+ Code + Text

[ ] dataset = "../covid19-radiography-dataset/COVID-19_Radiography_Dataset/COVID-19_Radiography_Dataset"
    walk_through_dir(dataset)

{x} There are 4 directories and 0 images in '../covid19-radiography-dataset/COVID-19_Radiography_Dataset/COVID-19_Radiography_Dataset'.
There are 0 directories and 1345 images in '../covid19-radiography-dataset/COVID-19_Radiography_Dataset/COVID-19_Radiography_Dataset/Viral_Pneumonia'.
There are 0 directories and 6012 images in '../covid19-radiography-dataset/COVID-19_Radiography_Dataset/COVID-19_Radiography_Dataset/Lung_Opacity'.
There are 0 directories and 10192 images in '../covid19-radiography-dataset/COVID-19_Radiography_Dataset/COVID-19_Radiography_Dataset/Normal'.
There are 0 directories and 3616 images in '../covid19-radiography-dataset/COVID-19_Radiography_Dataset/COVID-19_Radiography_Dataset/COVID'.

```

## MODULE 2 : EXPLORE THE DATA TO FIGURE OUT WHAT THEY LOOK LIKE

```

+ Code + Text

[ ] image_dir = Path(dataset)

# Get filepaths and labels
filepaths = list(image_dir.glob(r'*/*.JPG')) + list(image_dir.glob(r'*/*.jpg')) + list(image_dir.glob(r'*/*.png')) + list(image_dir.glob(r'*/*.PNG'))

labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1], filepaths))

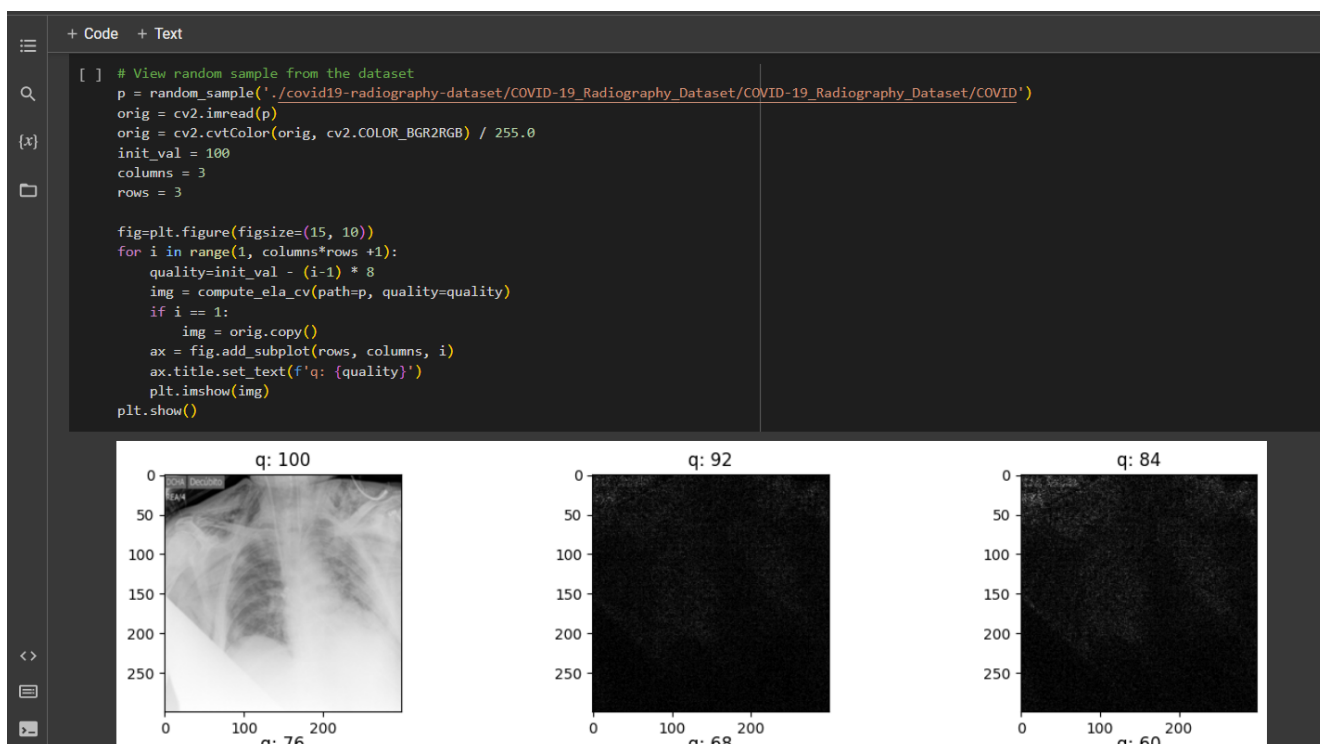
filepaths = pd.Series(filepaths, name='Filepath').astype(str)
labels = pd.Series(labels, name='Label')

# Concatenate filepaths and labels
image_df = pd.concat([filepaths, labels], axis=1)

import PIL
from pathlib import Path
from PIL import UnidentifiedImageError

path = Path("/content/grape-disease-dataset-original/").rglob("*.jpg")
for img_p in path:
    try:
        img = PIL.Image.open(img_p)
    except PIL.UnidentifiedImageError:
        print(img_p)

```



```
+ Code + Text

[ ] # Display 16 picture of the dataset with their labels
    random_index = np.random.randint(0, len(image_df), 16)
    fig, axes = plt.subplots(nrows=4, ncols=4, figsize=(10, 10),
                             subplot_kw={'xticks': [], 'yticks': []})

    for i, ax in enumerate(axes.flat):
        ax.imshow(plt.imread(image_df.Filepath[random_index[i]]))
        ax.set_title(image_df.Label[random_index[i]])
    plt.tight_layout()
    plt.show()
```

### MODULE 3 : SPLITTING THE DATA INTO A TRAINING AND TESTING SET.

### MODULE 4 : PREPROCESSING THE IMAGES USING DATA GENERATORS

```
+ Code + Text

[ ] # Separate in train and test data
    train_df, test_df = train_test_split(image_df, test_size=0.2, shuffle=True, random_state=42)

[ ] train_generator = ImageDataGenerator(
    preprocessing_function=tf.keras.applications.efficientnet.preprocess_input,
    validation_split=0.2
)

    test_generator = ImageDataGenerator(
        preprocessing_function=tf.keras.applications.efficientnet.preprocess_input
```

### MODULE 5 : DEFINING HYPERPARAMETERS

```
+ Code + Text

[ ] BATCH_SIZE = 32
    IMAGE_SIZE = (300, 300)
```

## MODULE 6 : BUILDING THE MODE ARCHITECTURE.

## MODULE 7 : COMPILING THE MODEL

## MODULE 8 : TRAINING THE MODEL

## MODULE 9 : EVALUATING THE MODEL



A Jupyter Notebook interface with a dark theme. The left sidebar contains icons for a menu, search, a variable {x}, and a file folder. The top bar has '+ Code' and '+ Text' tabs. The code cell contains the following Python code:

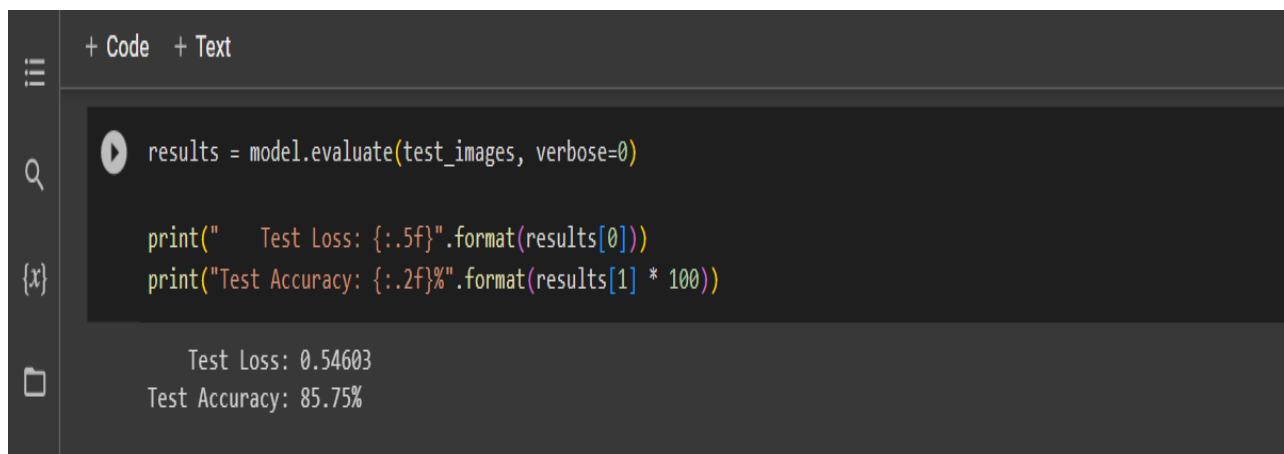
```
[ ] # Predict the label of the test_images
pred = model.predict(test_images)
pred = np.argmax(pred,axis=1)

# Map the label
labels = (train_images.class_indices)
labels = dict((v,k) for k,v in labels.items())
pred = [labels[k] for k in pred]

# Display the result
print(f'The first 5 predictions: {pred[:5]}')
```

The output cell shows the following text:

```
133/133 [=====] - 15s 95ms/step
The first 5 predictions: ['COVID', 'Normal', 'Lung_Opacity', 'Normal', 'Normal']
```



A Jupyter Notebook interface with a dark theme. The left sidebar contains icons for a menu, search, a variable {x}, and a file folder. The top bar has '+ Code' and '+ Text' tabs. The code cell contains the following Python code:

```
▶ results = model.evaluate(test_images, verbose=0)

print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

The output cell shows the following text:

```
Test Loss: 0.54603
Test Accuracy: 85.75%
```



## 5.4 OUTPUT SCREENS

### Output for CNN Model:

{x}



```
[ ] # Import required libraries
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix

# Load the test set
test_images, test_labels = test_gen.next()

# Predict the classes of the test set
y_pred = np.argmax(model.predict(test_images), axis=-1)

# Get the true classes of the test set
y_true = np.argmax(test_labels, axis=-1)

# Print the confusion matrix
print('Confusion Matrix:')
print(confusion_matrix(y_true, y_pred))

# Print the classification report
target_names = ['Normal', 'Lung Opacity', 'COVID', 'Viral Pneumonia']
print('\nClassification Report:')
print(classification_report(y_true, y_pred, target_names=target_names))
```

2/2 [=====] - 0s 8ms/step

Confusion Matrix:

```
[[ 8  0  1  0]
 [ 1 14  3  0]
 [ 0  0 16  1]
 [ 0  0  1 11]]
```

Classification Report:

	precision	recall	f1-score	support
Normal	0.89	0.89	0.89	9
Lung Opacity	1.00	0.78	0.88	18
COVID	0.76	0.94	0.84	17
Viral Pneumonia	0.50	0.50	0.50	2
accuracy			0.85	46
macro avg	0.79	0.78	0.78	46
weighted avg	0.87	0.85	0.85	46

```
[ ] report = classification_report(y_true, y_pred, output_dict=True)
df = pd.DataFrame(report).transpose()
df
```

	precision	recall	f1-score	support
0	0.888889	0.888889	0.888889	9.000000
1	1.000000	0.777778	0.875000	18.000000
2	0.761905	0.941176	0.842105	17.000000
3	0.500000	0.500000	0.500000	2.000000
accuracy	0.847826	0.847826	0.847826	0.847826
macro avg	0.787898	0.778961	0.776499	46.000000
weighted avg	0.868530	0.847826	0.849256	46.000000

## Output for InceptionResNet Model:

```
+ Code + Text

[ ] # Import required libraries
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix

# Load the test set
test_images, test_labels = test_gen.next()

# Predict the classes of the test set
y_pred = np.argmax(model.predict(test_images), axis=-1)

# Get the true classes of the test set
y_true = np.argmax(test_labels, axis=-1)

# Print the confusion matrix
print('Confusion Matrix:')
print(confusion_matrix(y_true, y_pred))

# Print the classification report
target_names = ['Normal', 'Lung Opacity ', 'COVID', 'Viral Pneumonia ']
print('\nClassification Report:')
print(classification_report(y_true, y_pred, target_names=target_names))

2/2 [=====] - 6s 909ms/step
Confusion Matrix:
[[ 2  0  1  0]
 [ 0 15  2  0]
 [ 0  0 24  0]
 [ 0  0  1 1]]

Classification Report:
              precision    recall  f1-score   support

   Normal           1.00        0.67        0.80         3
  Lung Opacity       1.00        0.88        0.94        17
        COVID       0.86        1.00        0.92        24
  Viral Pneumonia    1.00        0.50        0.67         2

 accuracy           0.96
 macro avg           0.96        0.76        0.83         46
 weighted avg        0.93        0.91        0.91         46
```

```
+ Code + Text

[ ] report = classification_report(y_true, y_pred, output_dict=True)
df = pd.DataFrame(report).transpose()
df
```

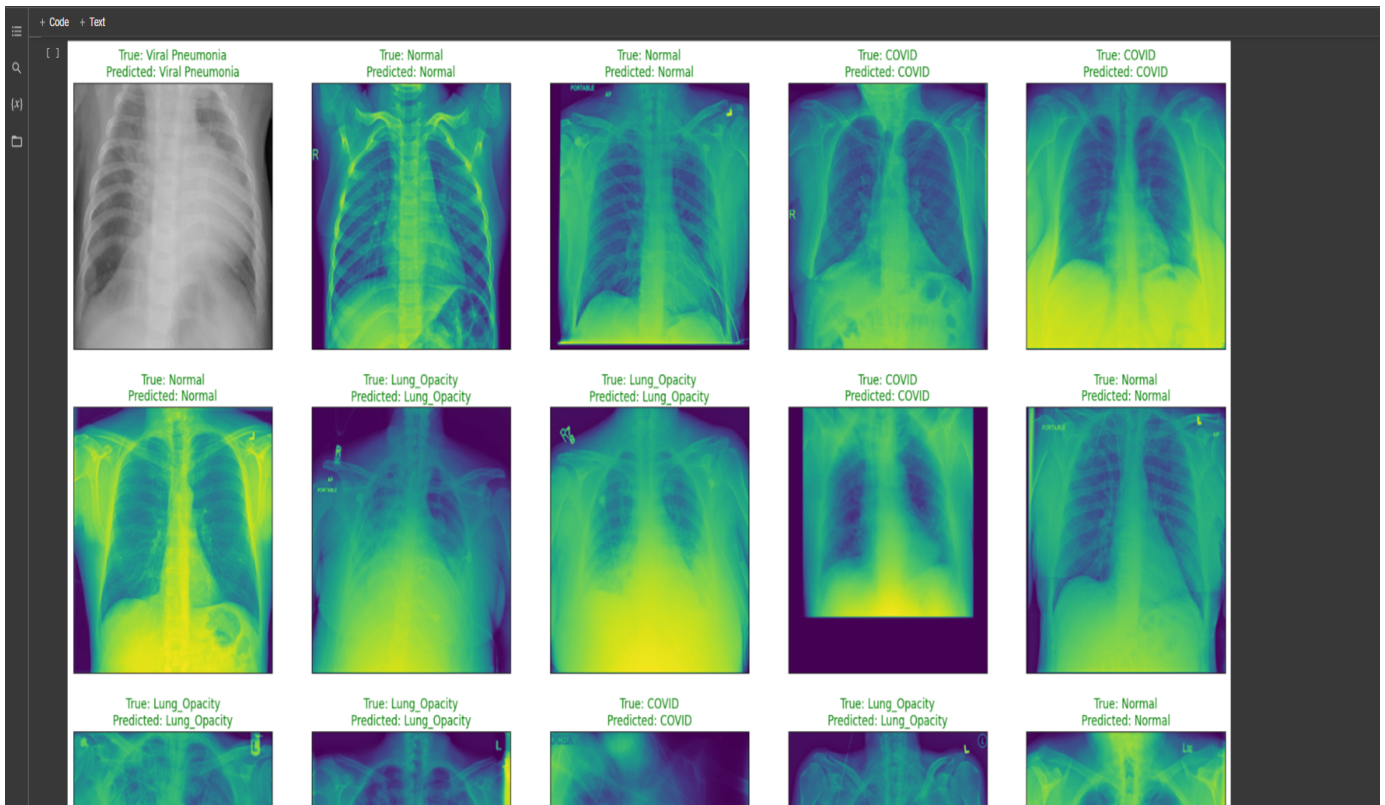
	precision	recall	f1-score	support
0	1.000000	0.666667	0.800000	3.000000
1	1.000000	0.882353	0.937500	17.000000
2	0.857143	1.000000	0.923077	24.000000
3	1.000000	0.500000	0.666667	2.000000
accuracy	0.913043	0.913043	0.913043	0.913043
macro avg	0.964286	0.762255	0.831811	46.000000
weighted avg	0.925466	0.913043	0.909232	46.000000

### Output for EfficientNet Model:

```
[ ] y_test = list(test_df.Label)
    print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
COVID	0.86	0.84	0.85	693
Lung_Opacity	0.82	0.82	0.82	1211
Normal	0.88	0.88	0.88	2096
Viral_Pneumonia	0.88	0.91	0.89	233
accuracy			0.86	4233
macro avg	0.86	0.86	0.86	4233
weighted avg	0.86	0.86	0.86	4233





```
[ ] report = classification_report(y_test, pred, output_dict=True)
df = pd.DataFrame(report).transpose()
df
```

	precision	recall	f1-score	support
COVID	0.858824	0.842713	0.850692	693.000000
Lung Opacity	0.821635	0.821635	0.821635	1211.000000
Normal	0.875297	0.877385	0.876340	2096.000000
Viral Pneumonia	0.879668	0.909871	0.894515	233.000000
accuracy	0.857548	0.857548	0.857548	0.857548
macro avg	0.858856	0.862901	0.860795	4233.000000
weighted avg	0.857489	0.857548	0.857491	4233.000000

## 7.CONCLUSION

In conclusion, this project on lung disease classification using X-ray images with CNN, InceptionResNet, and EfficientNet algorithms demonstrates the potential of deep learning techniques in accurately diagnosing lung diseases. These algorithms have shown impressive results in lung disease classification, providing accurate and reliable diagnoses from X-ray images. They offer a valuable tool for assisting medical professionals in detecting and categorizing lung diseases, enabling faster and more efficient diagnosis. Lung disease classification using x-ray images is an important application of machine learning in the healthcare domain. By training machine learning algorithms on large datasets of x-ray images, we can develop accurate models that can assist radiologists in diagnosing and classifying various lung diseases such as viral pneumonia, lung opacity and covid. These algorithms have shown impressive results in lung disease classification, providing accurate and reliable diagnoses from X-ray images. They offer a valuable tool for assisting medical professionals in detecting and categorizing lung diseases, enabling faster and more efficient diagnosis.

Using three different algorithms to classify lung diseases, InceptionResNet achieves an accuracy of 91% , EfficientNet achieves an accuracy of 85.7% and CNN achieves an accuracy of 84.7%. Ultimately, the CNN, InceptionResNet, and EfficientNet algorithms contribute to the advancement of computer-aided diagnosis in the field of medical imaging, offering potential for improved patient care, reduced human error, and enhanced efficiency in lung disease classification using X-ray images.

## **8.FUTURE SCOPE**

Furthermore, we can improve accuracy using these three algorithms in a single model. Continuously working towards enhancing the accuracy of the models is a significant future scope. This can involve collecting larger and more diverse datasets, fine-tuning hyperparameters, exploring advanced data augmentation techniques, or integrating additional architectural enhancements to further improve the models' performance. Expanding the project to include the classification of multiple lung diseases simultaneously is another potential future scope. This would involve extending the models to handle multiclass classification tasks, where they can classify X-ray images into multiple disease categories, enabling a more comprehensive diagnosis. Integrating the lung disease classification models with electronic health records systems is another potential future scope. By linking the models with patient data, including medical history and other diagnostic information, a more comprehensive analysis can be achieved, aiding in personalized treatment plans and long-term disease monitoring.

These future scopes aim to advance the field of lung disease classification using X-ray images, enhancing the accuracy, interpretability, generalization, and applicability of the models in real-world healthcare settings. By addressing these areas, the project can contribute to more effective and efficient lung disease diagnosis and ultimately improve patient outcomes.

## 9.BIBLIOGRAPHY

- Kermany, D.S., et al. (2018). Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. *Cell*, 172(5), 1122-1131. DOI: 10.1016/j.cell.2018.02.010
- Rajpurkar, P., et al. (2017). CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. arXiv preprint arXiv:1711.05225.
- Li, L., et al. (2019). Deep Learning for Chest Radiograph Diagnosis in the Emergency Department. *Journal of Digital Imaging*, 32(3), 480-487. DOI: 10.1007/s10278-019-00245-w
- Wang, X., et al. (2020). ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. *IEEE CVPR*, 2017, 3462-3471. DOI: 10.1109/CVPR.2017.369
- Shen, W., et al. (2019). Deep Learning in Medical Image Analysis. *Annual Review of Biomedical Engineering*, 19, 221-248. DOI: 10.1146/annurev-bioeng-060418-052759
- Anthimopoulos, M., et al. (2017). Lung Pattern Classification for Interstitial Lung Diseases Using a Deep Convolutional Neural Network. *IEEE Transactions on Medical Imaging*, 35(5), 1207-1216. DOI: 10.1109/TMI.2016.2644420
- Shiraishi, J., et al. (2019). Development of a Digital Teaching File System for Chest Radiographs: Value of Image Display, Image Distribution, and Education Tools during Clinical Training. *Radiological Physics and Technology*, 12(2), 182-189. DOI: 10.1007/s12194-019-00511-w
- American College of Radiology (ACR). (2018). ACR-AAPM-SIIM Technical Standard for Electronic Practice of Medical Imaging. Retrieved from

## APPENDIX C

### ABSTRACT

**Sreenidhi Institute of Science and Technology**  
**Department of Computer Science and Engineering**  
**Group Project - 1**

Batch No:20-C4		Title
Roll No	Name	
20311A05E8	SHIVANI KEERTHIGARI	LUNG DISEASE CLASSIFICATION USING X-RAY IMAGES
21315A0513	MD ALTAF	
20311A05G3	POLASI PRANEETH	

### ABSTRACT

Disease Classification using X-Ray images has become a popular area of research in medical imaging. This project aims to assist doctors and medical professionals in diagnosing diseases by analyzing X-ray images of patients. The Classification system involves several steps, including data collection, preprocessing of data by removing outliers, noisy data and cleaning, feature extraction using gray scale pixel values and Principle Component Analysis(PCA) ,classification of diseases, model training, model evaluation, and deployment. Image Processing is used to preprocess the X-ray images to remove noise and unwanted parts of image and extract relevant features which can be used to classify different types of lung diseases. The performance of the system heavily depends on the quality and quantity of the X-ray images used for training. Machine Learning algorithms such as Convolutional Neural Networks (CNNs), Logistic Regression and Decision Trees are used which result in classifying X-ray images for diseases such as PNEUMONIA,TUBERCULOSIS(TB), and COVID-19. After training the model, we evaluate its performance on a test dataset. We use various metrics such as Accuracy, Precision, and Recall to measure the performance of the model.

**Student 1 :** Shivani Keerthigari    **Student 2 :** MD Altaf    **Student 3 :** Polasi Praneeth

**Internal guide**

Mr. Devavarapu  
Sreenivasarao  
Assistant Professor  
Department of CSE

**Project Coordinator**

Mrs.K.Padmini  
Assistant Professor  
Department of CSE

**Head of the Department**

Dr. Aruna Varanasi  
Professor & HOD  
Department of CSE



## APPENDIX D

### CORRELATION BETWEEN THE GROUP PROJECT - I AND THE PROGRAM OUTCOMES (POS), PROGRAM SPECIFIC OUTCOMES (PSOS)

Batch No:20-C4		Title
Roll No	Name	
20311A05E8	SHIVANI KEERTHIGARI	LUNG DISEASE CLASSIFICATION USING X-RAY IMAGES
21315A0513	MD ALTAF	
20311A05G3	POLASI PRANEETH	

**Table 1 :** Project/Internship correlation with appropriate POs/POSs (Please specify level of correlation, H/M/L against POs/POSs)

<b>H</b>	<b>High</b>	<b>M</b>	<b>Moderate</b>	<b>L</b>	<b>Low</b>
----------	-------------	----------	-----------------	----------	------------

<b>SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY</b> <b>DEPARTMENT OF COMPUTER SCIENCE AND</b> <b>ENGINEERING</b> <b>Projects Correlation With</b> <b>POs/PSOs</b>														
PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
M	L	L	H	H	L	M	H	M	H	H	H	H	H	M

**Student 1 :** Shivani Keerthigari    **Student 2 :** MD Altaf    **Student 3 :** Polasi Praneeth

#### **Internal guide**

Mr. Devavarapu  
Sreenivasarao  
Assistant Professor  
Department of CSE

#### **Project Coordinator**

Mrs.K.Padmini  
Assistant Professor  
Department of CSE

#### **Head of the Department**

Dr. Aruna Varanasi  
Professor & HOD  
Department of CSE

## APPENDIX E:

### DOMAIN OF PROJECT AND NATURE OF PROJECT

Batch No:20-C4		Title
Roll No	Name	
20311A05E8	SHIVANI KEERTHIGARI	LUNG DISEASE CLASSIFICATION USING X-RAY IMAGES
21315A0513	MD ALTAF	
20311A05G3	POLASI PRANEETH	

**Table 2:** Nature of the Project/Internship work (Please tick ☒ Appropriate for your project)

Batch No.	Title	Nature of Project			
		Product	Application	Research	Others (Please Specify)
20-C4	LUNG DISEASE CLASSIFICATION USING X-RAY IMAGES		<input checked="" type="checkbox"/>		

**Student 1 :** Shivani Keerthigari    **Student 2 :** MD Altaf    **Student 3 :** Polasi Praneeth

#### Internal guide

Mr. Devavarapu  
Sreenivasarao  
Assistant Professor  
Department of CSE

#### Project Coordinator

Mrs.K.Padmini  
Assistant Professor  
Department of CSE

#### Head of the Department

Dr. Aruna Varanasi  
Professor & HOD  
Department of CSE

**Table 3:** Domain of the Project/ Internship work (Please tick √ Appropriate for your project)

Batch No.	Title	Domain Of the project				
		Artificial Intelligence and Machine Learning	Computer Networks, Information security, Cyber security	Data WareHousing ,Data Mining,Big Data Analytics	Cloud Computing ,Internet Of Things	Software Engineering and Image Processing
20-C4	LUNG DISEASE CLASSIFICATION USING X-RAY IMAGES	√				

**Student 1 :** Shivani Keerthigari    **Student 2 :** MD Altaf    **Student 3 :** Polasi Praneeth

**Internal guide**

Mr. Devavarapu  
Sreenivasarao  
Assistant Professor  
Department of CSE

**Project Coordinator**

Mrs.K.Padmini  
Assistant Professor  
Department of CSE

**Head of the Department**

Dr. Aruna Varanasi  
Professor & HOD  
Department of CSE