



**Faculty of Computers
& Artificial Intelligence**



Benha University

Pneumonia diagnosis system

A senior project submitted in partial fulfillment of the requirements for the degree of Bachelor of Computers and Artificial Intelligence.

Artificial Intelligence Departement,

Project Team

- 1- Sherif Ashraf Ahmed Rushdy
- 2- Yahia Zakaria Ibrahim
- 3- Sobeh Salah Sobeh
- 4- Mohamed Hosny Mosad Nasser
- 5- Abdelrahman Mohamed Khalil Afkar
- 6- Abdelkhalek Ashraf Shams
- 7- Aya Adel Mohamed Saber
- 8- Abdalla Ibrahim Wahuman

Under Supervision of

Dr. Ibrahim Zaghloul

Benha, **June 2023**

ACKNOWLEDGMENT

ALLAH is the first and the last to be thanked. We want to thank all those who helped us with their knowledge and experience. We will always appreciate their efforts. First and foremost, we would like to express a sincere gratitude to our mentor and advisor Dr. Ibrahim Zaghloul for his advice and support while working on this project. We deeply appreciate his valuable advice. Next, we want also to acknowledge all the support we have received from our faculty “Faculty of Computers & Artificial Intelligence” while working on this project. Finally, it has been a pleasure knowing everyone we worked with during that period: all the people from our department and from the other departments with whom we shared so many good experiences.

DECLARATION

We hereby certify that this material, which we now submit for assessment on the program of study leading to the award of Bachelor of Computers and Artificial Intelligence in Computer Science is entirely our work, and that we have exercised reasonable care to ensure that the work is original, and does not to the best of our knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of our work.

Signed: _____

Date: Wednesday, 21 June 2023.

ABSTRACT

Pneumonia and Covid-19 are two respiratory illnesses that have been spreading rapidly around the world. Diagnosing these diseases has been a challenge due to the similarities in their symptoms. However, chest X-rays have been proven to be effective in detecting these diseases. Recent research has focused on enhancing the accuracy of chest X-ray diagnosis using convolutional neural networks (CNN).

CNN has been designed to analyze medical images and assist in diagnosing diseases accurately. The system works by feeding the machine-learning algorithms with a vast pool of medical image databases to develop an understanding of the patterns of typical and atypical images. The trained model can then be applied to new medical images for diagnosis. Researchers have reported a high accuracy rate for diagnosing Covid-19 and pneumonia using this system, highlighting the potential of CNN in revolutionizing the medical field. With further research, machine learning applications like CNN could significantly enhance the efficiency of the healthcare system by providing accurate and faster diagnoses.

TABLE OF CONTENTS

Table of Contents

List of figures	iii
list of tables	v
LIST OF ACRONYMS/ABBREVIATIONS.....	vi
1 Introduction	7
1.1 Introduction to deep learning.....	7
1.1.1 CNN	7
1.1.2 API.....	9
1.1.3 Data Augmentation	10
1.1.4 Grad-Cam.....	11
1.1.5 Callbacks.....	12
1.1.6 Evaluation metrics:	13
1.1.6.1 Confusion matrix	13
1.1.6.2 Accuracy score.....	14
1.1.6.3 Classification Report.....	15
1.1.7 How deep learning work	17
1.1.7.1 Activation function	18
1.1.7.2 k-fold Cross-Validation.....	18
1.1.8 Transfer learning	19
1.1.8.1 ResNet-50	21
1.1.8.2 DenseNet.....	21
1.1.8.3 MobileNet	22
1.1.8.4 VGG-19	23
1.2 RELATED WORKS	24
2 Analysis and Design	26
2.1 DATASET DESCRIPTION	26
2.2 Block Diagram of the System.....	27
2.3 Use case.....	28
2.4 database diagrams.....	29
2.4.1 Entity Relationship Diagram (ERD)	29
2.4.2 Schema.....	30
2.5 Flowchart of the System.....	31
2.6 data flow of the system.....	31
2.7 Sequence diagram of the system	33
3 Deliverables and Evaluation.....	34
3.1 Introduction	34
3.2 Import requirements libraries	34
3.3 Get and process data.....	34
3.3.1 Splitting data into train and validation data	37
3.3.2 Image Data Generator	39
3.4 create model	41
3.4.1 Cnn1 hidden layer.....	41
3.4.2 Cnn2 hidden layer.....	43

Table of Contents

3.4.3	Mobilenetv2	43
3.5	Train model	44
3.6	evaluate model.....	47
3.6.1	MobileNetV2	48
3.7	Make predictions	50
3.8	System implementation	52
3.8.1	Home Page.....	53
3.8.2	Sign In Page	54
3.8.3	Sign Up Page.....	55
3.8.4	Update Profile Page	56
3.8.5	Upload Photo Page.....	56
3.8.6	Admin Page.....	57
3.8.7	Diagnosed Cases Page	58
3.8.8	Table Of Patients.....	59
4	Conclusion	60
5	REFERENCES.....	61

LIST OF FIGURES

Figure 1-1:General CNN Architecture	8
Figure 1-2: Geometric Augmentations	10
Figure 1-3: Random Augmentation.....	11
Figure 1-4:Architecture of Grad-CAM	12
Figure 1-5: Example with Grad-CAM	12
Figure 1-6: Confusion Matrix	14
Figure 1-7:Accuracy Score.....	15
Figure 1-8: Evaluation Metrics	16
Figure 1-9: How deep learning works	17
Figure 1-10: Examples of activation functions	18
Figure 1-11: K-Fold	19
Figure 1-12: Stratified K-Fold.....	19
Figure 1-13: CNN vs Transfer Learning	20
Figure 1-14: Traditional ML vs Transfer Learning	20
Figure 1-15: ResNet50 Architecture	21
Figure 1-16: DenseNet Architecture	22
Figure 1-17: MobileNet Architecture	23
Figure 1-18: VGG-19 Architecture	24
Figure 2-1:Samples of Chest X-ray Images in dataset	26
Figure 2-2:Block Diagram of the System.....	27
Figure 2-3:Use Case of the system.....	28
Figure 2-4:Entity Relationship diagram of database	29
Figure 2-5: Schema	30
Figure 2-6:Flowchart of the System.....	31
Figure 2-7: Data Flow of the System	32
Figure 2-8: Sequence Diagram of the system.....	33
Figure 3-1:requirements libraries	34
Figure 3-2:Sample image of pneumonia data.....	34
Figure 3-3:Sample image of covid19 data.....	35
Figure 3-4:Sample image of normal data	35
Figure 3-5:We convert train and test data to the data frame.....	36
Figure 3-6:train data frame of image.....	36
Figure 3-7:the shuffle() function of train data.	37
Figure 3-8:the shuffle () function of test data.....	37
Figure 3-9:The train data and test data are used.....	38
Figure 3-10:Stratified K-Fold.....	38
Figure 3-11:Data augmentation Parameter.....	40
Figure 3-12: Apply Data Augmentation.....	41
Figure 3-13:CNN-2 hiddem layer	43
Figure 3-14:MobileNet model.....	44
Figure 3-15:callback functions.....	45
Figure 3-16:Fit the model.....	46
Figure 3-17:MobileNetV2 loss.....	48
Figure 3-18:MobileNetV2 accuracy.....	48
Figure 3-19:MobileNetV2 classification accuracy.....	48
Figure 3-20:MobileNetV2 CM.....	49
Figure 3-21:Classification report for MobileNetV2	49
Figure 3-22:Make Prediction with Grad-CAM	51
Figure 3-23: Home Page	53
Figure 3-24: Home Page	54
Figure 3-25: Sign-in Page	54
Figure 3-26: Sign-Up Page.....	55
Figure 3-27: Update Profile.....	56
Figure 3-28: Upload Photo Page	56
Figure 3-29: Upload Photo Page	57

List of Tables

Figure 3-30: Admin Page	57
Figure 3-31: Diagnosed Cases Page	58
Figure 3-32: Table of Patients	59

LIST OF TABLES

Table 1: Summary of Accuracy	53
Table 2: Table of Patients	80

LIST OF ACRONYMS/ABBREVIATIONS

ML	machine learning
DL	Deep learning
AI	artificial intelligence
CNN	convolutional neural network
DNN	deep neural network
CAD	Computer-Aided Designs
SIFT	Scale Invariant Feature Transform
SURF	Speeded Up Robust Features
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
CM	confusion matrix
VGG	Visual Geometry Group
CV	Computer Vision
Densenet	Densely-Connected Convolutional Networks
API	Application programming interface

Chapter One

1 INTRODUCTION

Pneumonia is a life-threatening respiratory infection caused by bacteria, fungi, or viruses that infect and fill the human lung air sacs with fluid or pus. Pneumonia is a leading cause of death in children and the elderly around the world. Chest X-rays are the most thorough method of detecting pneumonia, and the data must be examined by a physician. Due to an erroneous diagnosis and treatment, a person died because of the inconvenient method of diagnosing pneumonia. It is now possible to construct an autonomous system for identifying pneumonia and treating the condition, especially if the patient is in a distant area with few medical services, thanks to advances in computer technology. As a result, this is the key motivation.

1.1 INTRODUCTION TO DEEP LEARNING

Deep Learning is a data science approach that allows a computer to gain insight into patterns and existing data to forecast the outcomes. With a robust machine learning model, tasks like identifying Pneumonia disease may be automated, efficient, and accurate, as well as identify patterns and qualities in data that humans may forget. To train data input and apply statistical analysis to create specified output, a deep learning algorithm is used. Deep learning uses a variety of algorithms to predict disease. It is possible that humans will not notice. To train data input and apply statistical analysis to create specified output, a deep learning algorithm is used. Deep learning uses a variety of algorithms to anticipate outcomes.

1.1.1 CNN

Throughout the period Computer-Aided Designs (CAD) has emerged as the maximum exploration region in tool getting to know. In using Machine Learning (ML) techniques in scientific pictures, huge features are of uppermost importance. For this cause, most of the previous algorithms used hand-crafted parts for developing CAD systems based primarily on reading photographs. However, the hand-crafted talents with barriers that were diverse and consistent with duties were not capable of presenting the masses with

immense talents. Employment of Deep Learning models located their self-functionality of extracting valuable skills in photograph kind duties. This way of characteristic-extraction desires transfers getting to know techniques wherein pre-professional CNN models study the common talents on largescale datasets like ImageNet, which might be moved to the famous project in addition, the kind used with high-rich extracted talents, not an unusual overall performance in classifying pictures.

The Chest screening subroutines that chest screening might use for sensing lung nodules may be used to diagnose remarkable illnesses that embody Pneumonia, effusion, cardiomegaly, etc. Among these, Pneumonia is an infectious and deadly infection that moves over masses of people, majorly people aged above sixty-five and tormented by chronic ailments like bronchial allergies or diabetes. In diagnosing Pneumonia, chest X-Rays are considered due to the truth; the best approach is to determine the amount and area of the septic region withinside the lungs. However, reading chest radiographs is not a leisurely project for radiotherapists. In the chest X-ray pictures, the appearance of Pneumonia can be hazy and can be misapprehended with unique diagnoses. The evaluation of chest X-Ray, particularly in the case of Pneumonia, can be misleading because of the truth, many unique troubles like congestive coronary heart failure, lung scarring, etc., can mimic Pneumonia. Thus, the project is challenging, and the improvement of a set of pointers for detecting thoracic illnesses like Pneumonia also needs to increase the accessibility of medical settings in an extended manner off regions. In this check, we evaluated the overall conventional everyday famous not unusual, usual overall performance of extreme variations of pre-informed CNN fashions determined via one-of-a-kind classifiers for classifying everyday and regular chest X-Rays.

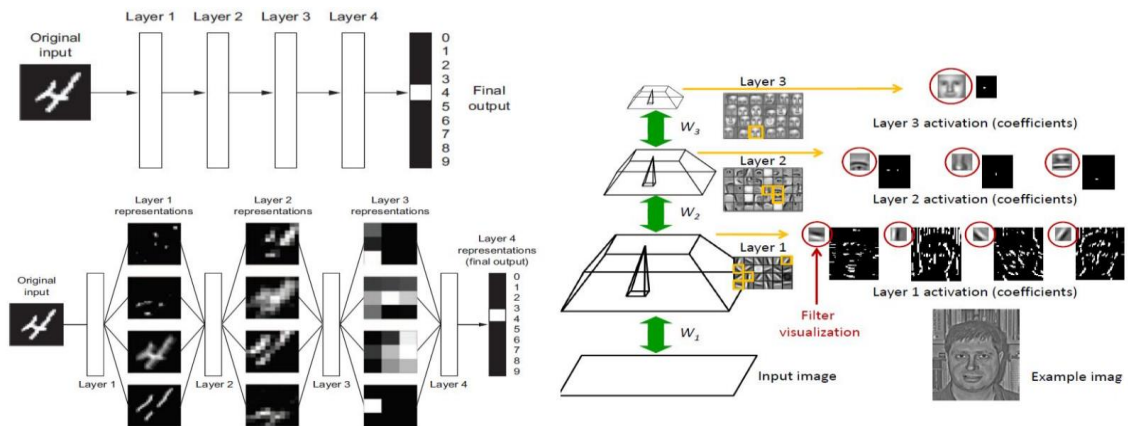


Figure 1-1: General CNN Architecture

1.1.2 API

APIs are one of the key building blocks of modern web applications, enabling communication and data transfer between disparate systems. Flask is a popular Python-based web application framework that allows developers to build web applications quickly and easily. The integration of APIs with Flask allows developers to develop and deploy powerful, scalable, and flexible applications that can meet with a wide range of external services. When integrating an API with Flask, developers need to pay careful attention to the architecture of their application, including the design of endpoints and handling of requests and responses. Additionally, security is a critical concern when developing APIs, so developers must consider using secure authentication methods, encryption protocols, firewalls, and other security measures to prevent unauthorized access or data breaches. Overall, Flask's ability to integrate with APIs provides developers with an exceptional opportunity to deliver high-quality software solutions quickly while maintaining strict security protocols throughout the development process.

APIs, or Application Programming Interfaces, are crucial components of modern software development. Flask, an open-source Python web framework, offers a simple and lightweight way to create APIs that can be integrated into web applications. The importance of using Flask with APIs lies in its ability to streamline the development process by supplying efficient and scalable tools for building and deploying robust APIs. Flask's ability to simplify coding tasks allows developers to focus more on product development rather than spending too much time on boilerplate code.

Flask's flexibility makes it an excellent choice for creating RESTful APIs that can be used across various platforms such as iOS and Android devices. With Flask's strong support for microservices & containerization technologies like Docker; it is easy to deploy APIs that are resilient and fault-tolerant even under heavy loads. Flask allows for quick integration with other third-party services via plugins and extensions, which enables developers to leverage the power of Multiple libraries like SQL Alchemy database toolkit for database access.

The simplicity offered by Flask also supports the agile method of programming where apps are built incrementally through smaller iterations ensuring faster delivery through working software increments sprints.

With Cracking down on monolithic codebases and organizations breaking down larger projects into smaller microservices advantages from Flask have allowed many

1.1.3 Data Augmentation

Data augmentation is a technique used in machine learning to create more training data by transforming existing data and adding variations that help improve the model's accuracy. The idea behind this method is that if we have more data, we can train our models more effectively. Data augmentation can be used for several types of data such as images, audio, and text.

To understand the concept of data augmentation, let us take the example of image recognition. In image recognition, we may have limited labeled images to train a model on. By using techniques like rotating the image, changing its brightness or contrast level, or flipping it horizontally/vertically; we can create several different versions of each original image with different angles and light levels. This creates a larger dataset and improves the model's ability to classify new images.

Text augmentation also involves creating new samples from existing ones by changing some parts such as synonyms replacement or removing/adding words. Audio augmentation includes manipulating sound effects including pitch shifting, noise addition, or removal which can be particularly useful if there are not enough training samples available for ambient sounds.

Overall, data augmentation helps you overcome two main challenges which tend to hinder deep learning concepts underfitting, training stability, and lack of generalizability due to the limited amount of accurate

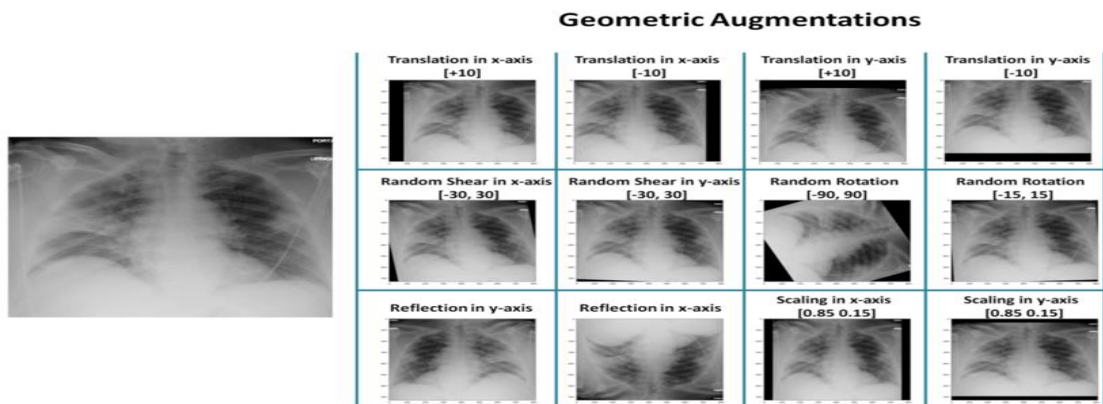


Figure 1-2: Geometric Augmentations

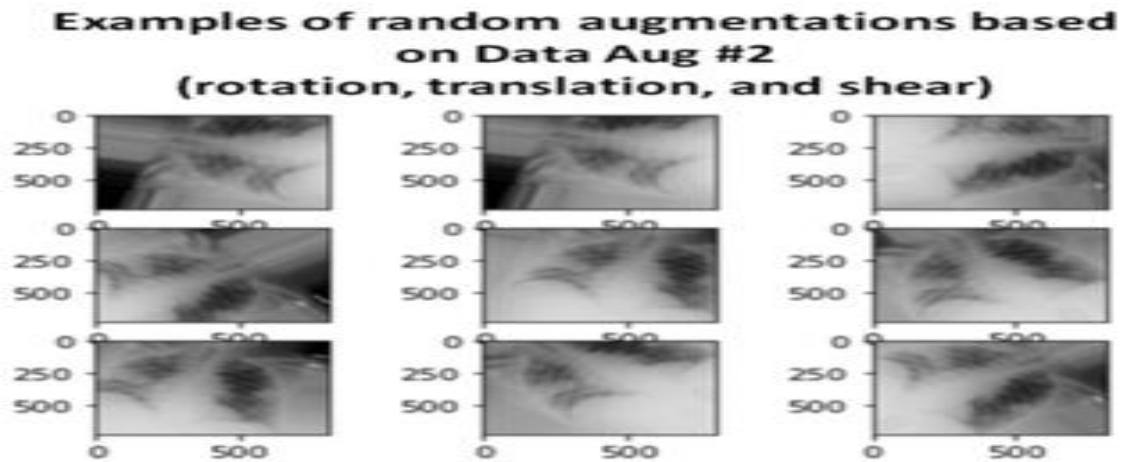


Figure 1-3: Random Augmentation

1.1.4 Grad-Cam

Grad-cam is a tool that helps to visualize how deep neural networks make decisions. It allows us to see which parts of an image the network focused on most when making its prediction. This is incredibly useful for understanding how these algorithms work and can help identify potential biases or weaknesses in the model.

To use Grad-cam, we start by training our model on some datasets. Once trained, we select an image we want to analyze and feed it through the network. Grad-cam then looks at the final layer of the network and calculates a heatmap showing which areas of the input most contributed to the output decision. We can overlay this heatmap onto the original image, giving us a visual representation of what parts of the input were most important for the network's decision.

This type of analysis is particularly useful for applications like medical imaging or autonomous cars, where we need to understand exactly why a given prediction was made to trust it or improve upon it. Grad-cam has quickly become an essential tool in many machine learning workflows, helping researchers gain insight into complex models and improving their ability to create more advanced algorithms going forward. So, let me explain in simple terms how Grad-CAM works. Grad-CAM stands for Gradient-weighted Class Activation Mapping, and it is a technique used to visualize the regions of an image that were important in making a classification decision by a Convolutional Neural Network (CNN). When a CNN processes an image, it performs

convolutions and pooling operations on the image's pixels. The output of these layers is called feature maps. Grad-CAM computes the gradient of the target class activation concerning each feature map and uses this information to weight each feature map based on its contribution to the target class activation. Then, it generates a heatmap by combining these weighted feature maps and visualizes the regions where the neural network focused on in making its prediction. This technique is useful for understanding what features of an image are relevant for a certain classification task, which can help improve model interpretability and identify biases or errors.

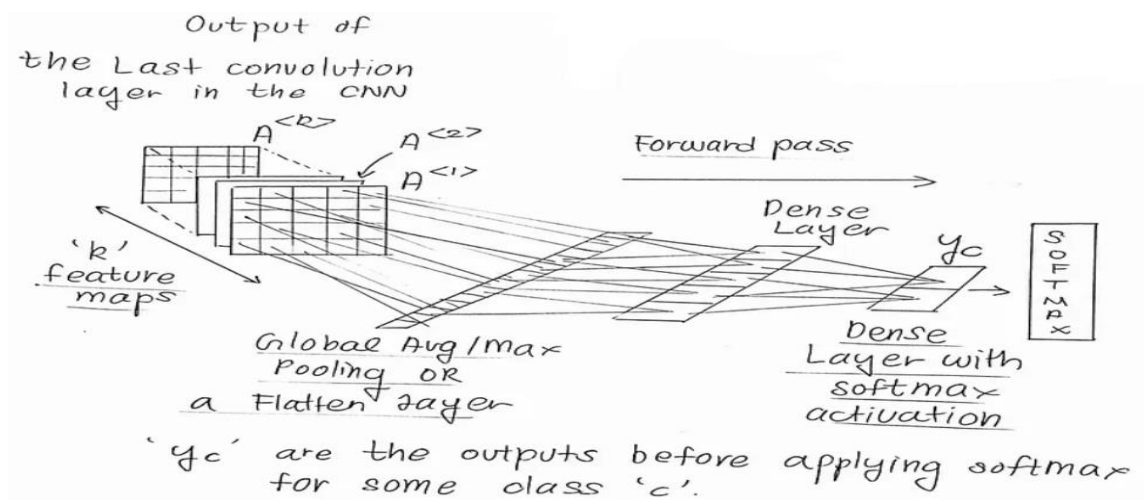


Figure 1-4: Architecture of Grad-CAM



Figure 1-5: Example with Grad-CAM

1.1.5 Callbacks

So, let us talk about callbacks in Keras. These little functions can be a real lifesaver when it comes to training your machine learning models effectively. Essentially, callbacks are

methods that are called during the training process at different points depending on what you set them up to do. Most commonly, they are used to save the best-performing version of a model during training or to stop the training process early if it is not improving anymore. But there are plenty of other ways you can leverage callbacks in Keras too. Most importantly, they help make sure that your model is being trained properly, so you do not waste hours (or days) running a model that will not produce helpful results. Overall, if you are serious about optimizing your AI and machine learning workflow with Keras, callbacks are worth exploring further!

1.1.6 Evaluation metrics:

1.1.6.1 Confusion matrix

The confusion matrix is a tool used in various fields, especially statistics and data science, to evaluate the accuracy of a model in classifying or predicting outcomes. The matrix is essentially a table that displays the predicted values against the actual values, detailing the number of true positives, false positives, true negatives, and false negatives. This information allows analysts to calculate metrics such as precision, recall, and F1 score to determine how well their model is performing. However, it is important to note that the confusion matrix only works for binary classification problems - those with two outcomes. For multiple-class classifications, an extension called the multiclass confusion matrix is employed to provide insights into each class's performance. Overall, understanding how to interpret and analyze a confusion matrix can help researchers improve their models' accuracy and make more informed decisions based on their findings.

Confusion Matrix

- True Positive (TP) is an outcome where the model correctly predicts the positive class.
- True Negative (TN) is an outcome where the model correctly predicts the negative class.
- False Positive (FP) is an outcome where the model incorrectly predicts the positive class.
- False Negative (FN) is an outcome where the model incorrectly predicts the negative class.

		Recognition/Prediction	
		Positive	Negative
Reference	Positive	 True Positive	 False Negative
	Negative	 False Positive	 True Negative

Figure 1-6: Confusion Matrix

1.1.6.2 Accuracy score

So, have you ever heard of the term "accuracy score"? It is a crucial metric when it comes to evaluating the performance of our predictive models. It measures how well our model's predictions match up with the actual outcomes. The accuracy score is calculated by dividing the number of correct predictions by the total number of predictions made. Now, it is essential to keep in mind that an accuracy score alone cannot indicate the quality or effectiveness of a model entirely. While a high accuracy score means that the model is making accurate predictions, it does not tell us anything about its ability to generalize to new data or handle complex situations. Therefore, we must supplement our analysis with other metrics like precision, recall, and F1-score for better insights and understanding.

To wrap up, if you are developing machine learning models or interpreting their outcome-ability metrics like accuracy score plays a vital role in assessing how good your models are at making decisions from input data and can provide clues for steps to take towards improvements.

- Accuracy:

the proportion of correct classifications.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

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

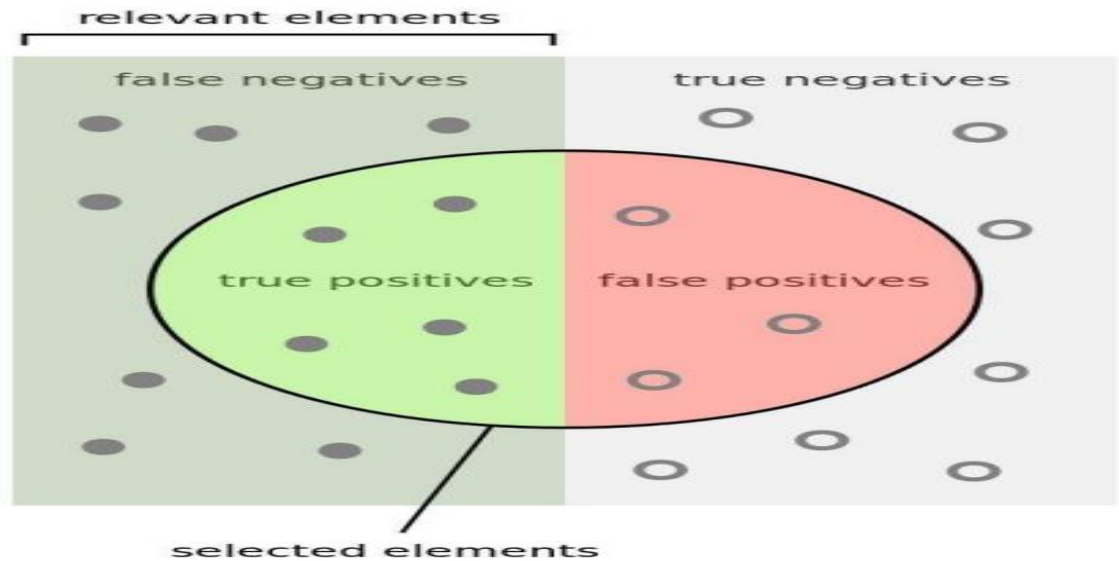


Figure 1-7:Accuracy Score

1.1.6.3 Classification Report

A classification report is a critical tool used in data analytics, machine learning, and artificial intelligence models. It provides a summary of the accuracy and completeness of the model's prediction outcomes against actual observations or target classes. A classification report provides an in-depth analysis of the performance of the classifier algorithm through various metrics such as precision, recall, F1 score, and support.

The precision metric represents how many true positive predictions were made relative to all positive predictions (true positives plus false positives). The recall metric indicates the proportion of true positive predictions relative to actual events (true positives plus false negatives), while the F1 score represents the harmonic mean between precision and recall. Finally, support refers to the total number of instances in each class.

Professionals utilize these metrics to evaluate prototypes designed for training classifiers on new datasets or applications.

In conclusion, it is important for professionals working with data analytics and machine learning models to accurately interpret classification reports when attempting to assess performance criteria. These reports provide insight into areas that may require optimization or fine-tuning for improved model accuracy regarding specific problem sets. Therefore, one must understand its mechanics thoroughly by knowing key metrics such as precision-recall values and their significance behind evaluating model performance efficiently.

- Precision:

Precision is the ability of a classifier not to label an instance positive that is negative.
precision is the Accuracy of positive predictions.

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

- Recall:

Recall is the ability of a classifier to find all positive instances. a Fraction of positives that were correctly identified.

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

- F1-score:

This is a weighted harmonic mean value using both Precision and Recall.

$$F1 = 2 \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- Support:

Support is the number of actual occurrences of the class in the specified dataset.

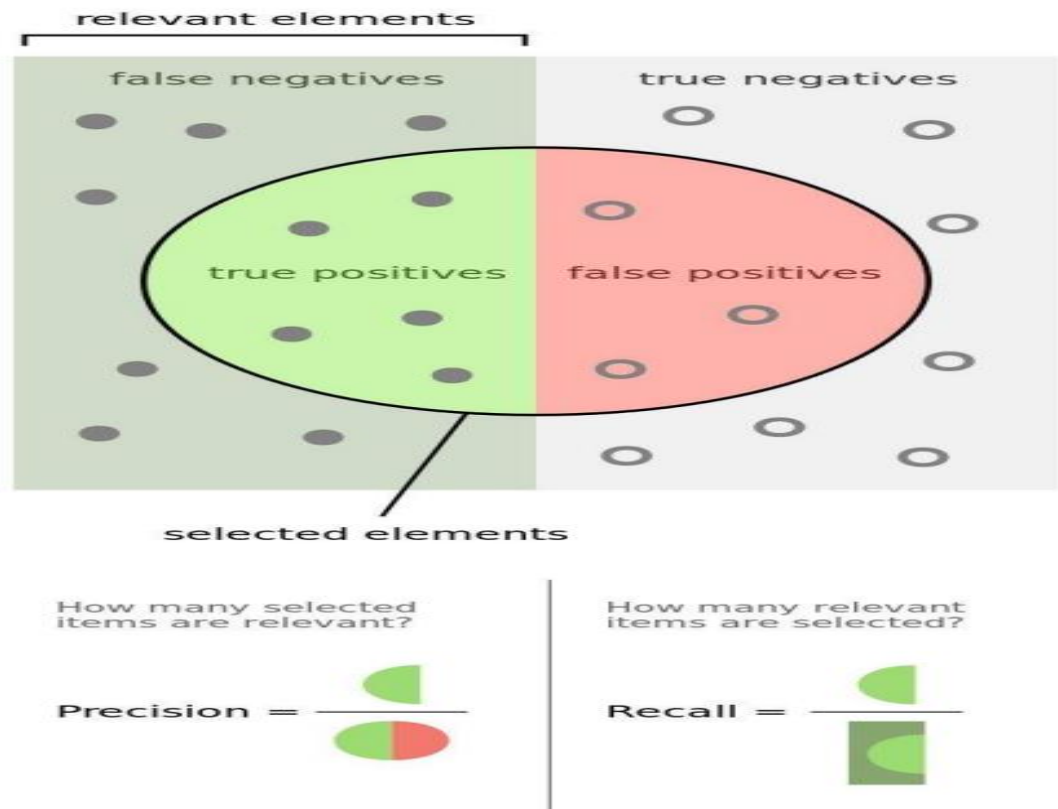


Figure 1-8: Evaluation Metrics

1.1.7 How deep learning work

Deep learning is a powerful artificial intelligence technique that involves several layers of algorithms working together to create complex, meaningful insights from vast amounts of data. The core principle behind deep learning is the ability of neural networks to identify patterns and relationships within datasets that humans might otherwise miss. These neural networks are large, interconnected networks of processing nodes that work together to process information, learn from it, and make predictions based on that learning. During training, data is fed into the network in multiple iterations or batches, with each iteration helping to fine-tune the weights associated with each node in the network. This gradually improves the accuracy of predictions over time and allows deep learning algorithms to excel at tasks like image recognition, speech processing, and natural language understanding. Although deep learning may seem like magic on the surface, its workings are grounded in solid mathematical principles that have been refined over decades by researchers in fields like computer science and statistics. By applying these principles to real-world problems such as disease diagnosis or fraud detection, deep learning has become one of the most powerful tools available for solving some of today's most pressing challenges.

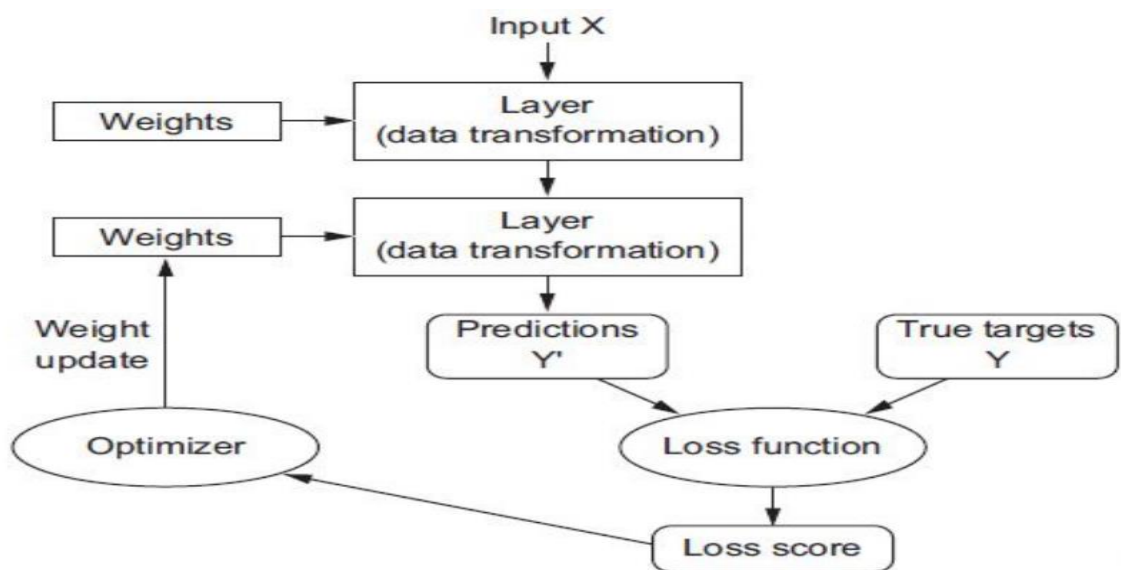


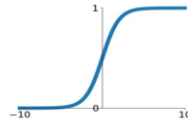
Figure 1-9: How deep learning works

1.1.7.1 Activation function

An activation function is a nonlinear transformation applied to the inputs of a neural network node. It determines whether and how much output should be generated based on the input data. There are several types of activation functions, including sigmoid, ReLU (rectified linear unit), tanh (hyperbolic tangent), SoftMax, and many more. The choice of activation function depends on the nature of the problem being solved and the properties required in the model's output. For example, sigmoid functions are commonly used for binary classification problems, while ReLU functions work well with deep networks for capturing complex features. Activation functions must also be differentiable to allow efficient optimization using backpropagation algorithms. A poorly chosen activation function can severely affect a model's accuracy or training time; therefore, it is essential to choose an appropriate activation function that enhances a neural network's performance.

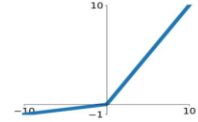
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



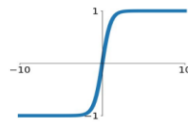
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

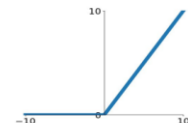


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ReLU

$$\max(0, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

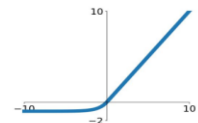


Figure 1-10: Examples of activation functions

1.1.7.2 k-fold Cross-Validation

K-fold cross-validation is a widely used technique in machine learning and data analysis that involves dividing the input data into k subsets or folds, each containing an equal number of samples. The process then runs k iterations, with each fold being used once as the validation set while the others are combined to form the training set. This allows for a comprehensive evaluation of the model's performance by building multiple models and using all available data in both training and validation sets. Additionally, it helps prevent issues such as overfitting and selection bias by reducing variance and increasing generalizability. K-fold cross-validation can improve model accuracy, reliability, and

confidence in findings from small datasets; thus, making it a valuable tool in modern ML workflows where limited sample sizes is a common challenge. However, it should be used with caution when processing extremely large datasets due to potential computational costs associated with repeated running of multiple models.

With K-Fold Cross-Validation, you divide the images into K parts of equal size. You then train your model K several times with a different training and validation set.

	Data				
Iteration 1	Validate	Train	Train	Train	Train
Iteration 2	Train	Validate	Train	Train	Train
Iteration 3	Train	Train	Validate	Train	Train
Iteration 4	Train	Train	Train	Validate	Train
Iteration 5	Train	Train	Train	Train	Validate

Figure 1-11: K-Fold

Stratified K-Fold differs from normal K-Fold in that it makes sure that each fold has the same percentage of samples of each class. This is especially useful if your training data is not uniformly distributed.

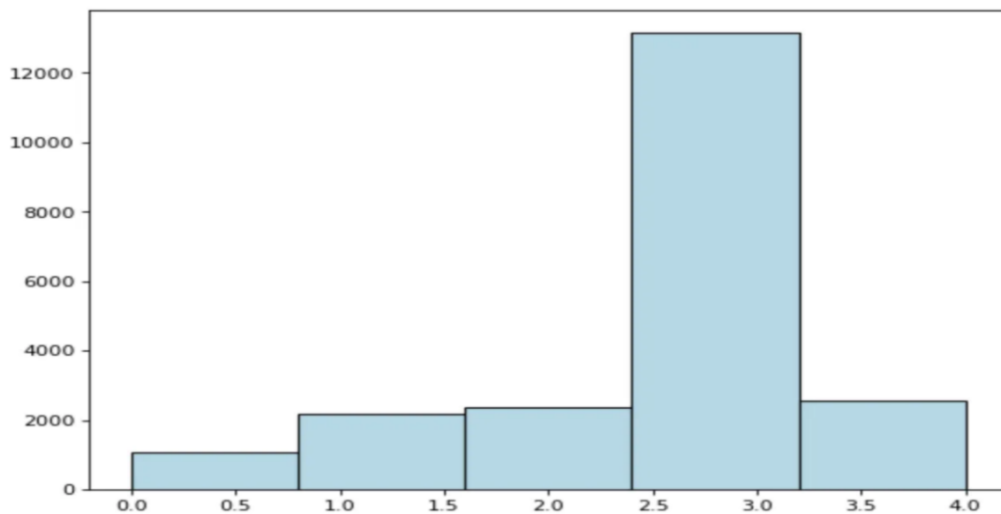


Figure 1-12: Stratified K-Fold

1.1.8 Transfer learning

Transfer learning is a popular technique in machine learning, where a pre-trained model is used as the starting point for training a new task or domain. This approach has become increasingly important due to the increased availability of large-scale pre-

trained models and the need for faster and more accurate training on limited data sets. Transfer learning operates on the idea that knowledge gained from training on one task can often be applied to another similar task with some modifications, thus accelerating the process of training on new tasks.

There are two types of transfer learning: domain adaptation and model adaptation.

Domain adaptation involves adapting an existing trained model to perform well on a different but related distribution of inputs while model adaption involves augmenting pre-trained layers with additional ones trained specifically for an auxiliary task. Both types of transfer learning have their unique benefits depending upon the requirement of the task.

Transfer learning has proven itself as a transformative technology, particularly in various areas such as image recognition, natural language processing, speech recognition, and sentiment analysis. With advancements in Deep Learning techniques such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformer-based models like BERT have opened up new opportunities in transfer learning where models trained on high-level abstraction features can be plugged into downstream.



Figure 1-13: CNN vs Transfer Learning

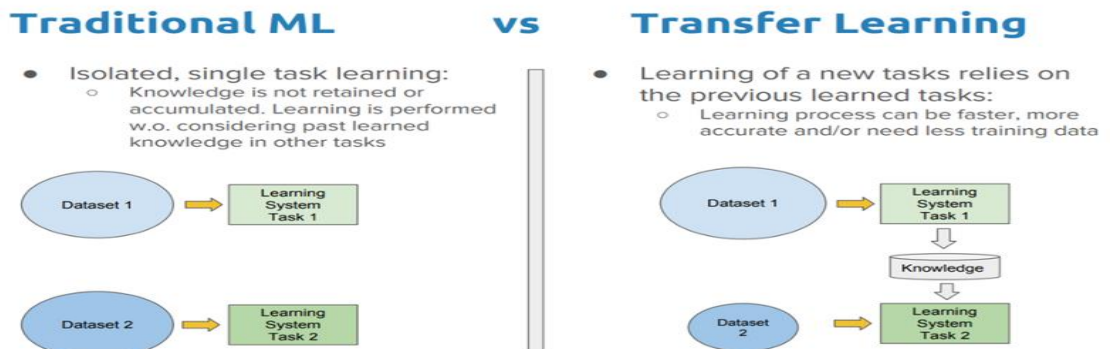


Figure 1-14: Traditional ML vs Transfer Learning

1.1.8.1 ResNet-50

ResNet-50 is a convolutional neural network (CNN) that was first introduced in 2015. It has fifty layers featuring residual connections, which allow the model to differentiate between low-level features such as edges and high-level features like shapes and objects. ResNet-50 has been trained on large datasets such as ImageNet, making it capable of performing various visual recognition tasks with high accuracy levels. Additionally, the network has shown remarkable success in transfer learning where it can be fine-tuned for specific tasks like object detection or image segmentation without requiring extensive training.

However, despite its performance improvements over previous CNN architectures, ResNet-50 requires significant computational resources and time to train due to its depth and complexity. To address this issue, researchers created variations of ResNet-50 that enable efficient inference at faster speeds while still maintaining similar levels of accuracy. For instance, some models deliberately reduce the number of parameters by increasing the stride size or reducing dimensionality in certain layers.

In conclusion, ResNet-50 is a powerful deep-learning architecture that has found widespread use in various computer vision applications. Its ability to extract

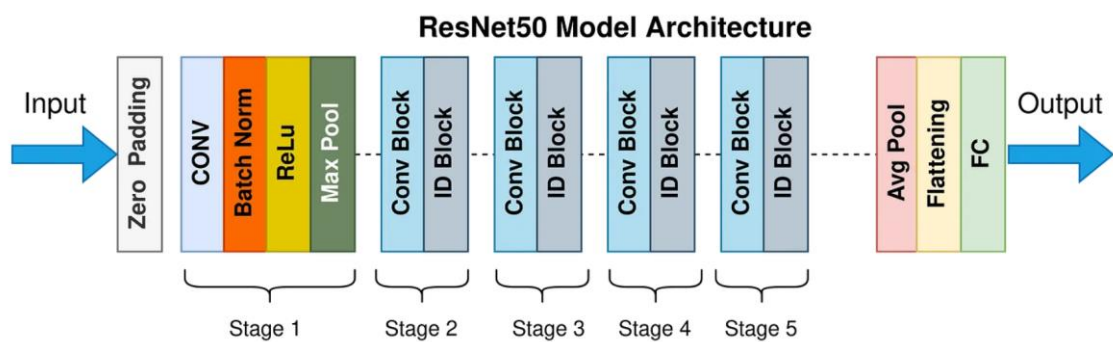


Figure 1-15: ResNet50 Architecture

1.1.8.2 DenseNet

DenseNet is a deep convolutional neural network that has become popular in computer vision tasks due to its improved performance over traditional CNN architectures. DenseNet's key innovation is the dense connectivity between layers, wherein each layer receives input not just from the previous layer but also from all preceding layers. This

facilitates the learning of more complex features and reduces feature redundancy. Additionally, DenseNet employs bottleneck layers which help reduce computational costs and improve generalization. Another strength of DenseNet is its ability to address the vanishing gradient problem by allowing information to easily flow through the network via shortcut connections. In addition to classification tasks, DenseNet has been successfully applied in various computer vision tasks such as object detection, semantic segmentation, and medical image analysis. With its high accuracy and efficiency, DenseNet is expected to continue being an important research direction in the field of computer vision.

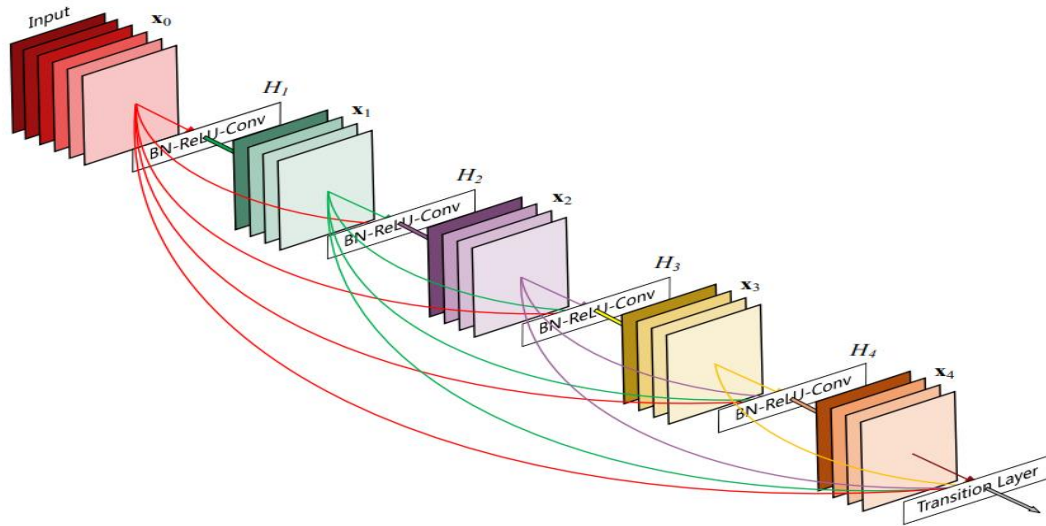


Figure 1-16: DenseNet Architecture

1.1.8.3 MobileNet

MobileNet is a highly efficient neural network architecture designed to enable the deployment of deep neural networks on mobile and embedded devices. It was developed by Google in 2017 and has since become one of the most popular architectures for mobile applications. The key feature of MobileNet is its ability to achieve high accuracy while minimizing the number of trainable parameters, making it ideal for resource-constrained environments. Furthermore, MobileNet employs depth-wise separable convolutions, which decompose regular convolutions into two stages: depth-wise convolutions and pointwise convolutions. This approach reduces the computational cost of each convolution operation by several orders of magnitude while maintaining accuracy levels comparable to more complex architectures. Additionally, MobileNet has been fine-tuned

for specific tasks such as image classification, object detection, and semantic segmentation, among others. Its versatility and efficiency have allowed it to be used in a wide range of applications beyond just computer vision. Overall, MobileNet is an excellent choice for developers who need to implement deep learning models with limited resources without sacrificing accuracy or performance.

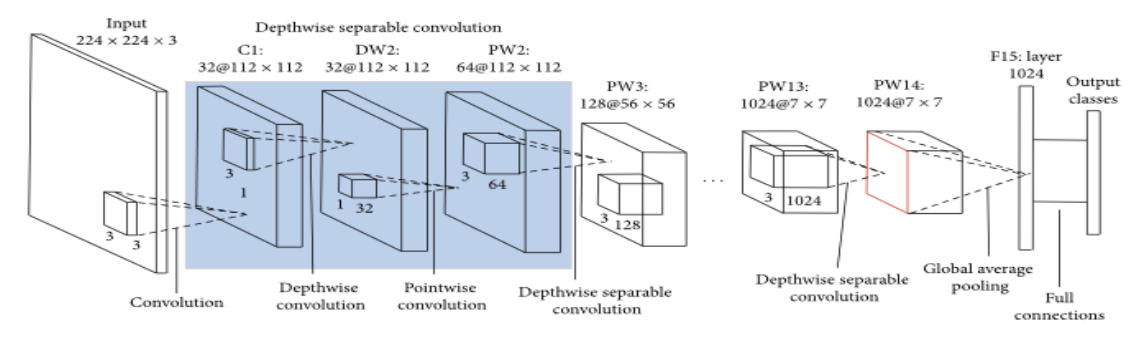


Figure 1-17: MobileNet Architecture

1.1.8.4 VGG-19

VGG-19 is a convolutional neural network (CNN) architecture that has nineteen layers and was developed by researchers at the Visual Geometry Group (VGG) at Oxford University. It has achieved outstanding performances on various image recognition tasks, including the ImageNet dataset, which contains over fourteen million images for training and testing object detection algorithms. VGG-19 is a deep learning model that uses a filter size of 3×3 with a stride of one pixel in its convolutional layers. It also uses max pooling after every two convolutional layers to reduce their spatial dimensions. The fully connected layers are preceded by a global average pooling layer that aggregates the output of each feature map. Therefore, VGG-19 is considered a powerful network because it can identify features from various levels of abstraction. Despite its performance efficiency, VGG-19 requires large amounts of computational resources due to its complexity, but this can be mitigated through parallel computing or optimization techniques such as pruning and quantization. In conclusion, VGG-19 is an efficient and effective deep learning model that has demonstrated high accuracy on several benchmark datasets; however, its applicability depends on the nature of the problem being solved and available computation resources.

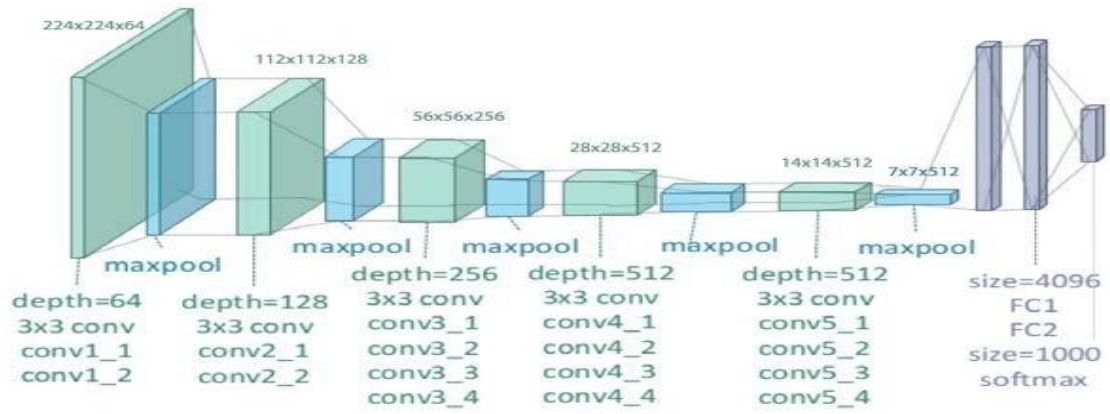


Figure 1-18: VGG-19 Architecture

1.2 RELATED WORKS

In the field of medical picture categorization, deep learning approaches are becoming increasingly popular. This is due in large part to the high success rate of these algorithms. Early machine learning models faced difficulties such as low accuracy, which was mostly dependent on the performance of feature extraction layers. Feature extraction techniques such as Scale Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), and others are used in traditional machine learning techniques. Many of the most well-known and successful medical picture classification projects used deep learning algorithms like Convolutional Neural Networks (CNN). The results of a CT scan were used by Roth et al to identify and classify Lymph Node (LN) detection using CNN. They were able to achieve high classification accuracy when compared to previous approaches that used boosting-based feature selection, which resulted in a high number of false positives. For cell picture segmentation and tracking, CNN with the U-net technique was utilized, as demonstrated in. Olaf Ranneberger, Philipp Fischer, and Thomas Brax introduced U-net, a 23-layer CNN, in 2015. To correctly identify and categorize lung nodules, Lan et al used deep CNN with the U-Net algorithm and the addition of a Residual network. In 2015, the residual network was introduced.

Another significant aspect of medical image analysis is organ segmentation. The pancreas, liver, kidneys, and heart are just a few of the organs in our body that have a lot of structural variation. As a result, medical picture analysis becomes a more challenging task. Deep CNN was employed for pancreatic segmentation using Computerized

Tomography (CT) findings, as previously stated. Organ segmentation is a critical step in detecting aberrant growth or malignancy in organs with a lot of anatomical variation. Bier et al. present a new method for recognizing anatomical landmarks in X-ray pictures that is independent of viewing direction. They were able to identify twenty-three anatomical landmarks of the pelvis from a single X-ray using the CNN model. presented a bottom-up strategy including intensive tagging of picture patches to cover the entire organ. When compared to prior organ segmentation methods that used random forests, this method has a high level of accuracy. This is a difficult assignment since scans of the lung field tend to involve rib cages with variable bone densities, the presence of clavicles, and, in some circumstances, the existence of different lung irregularities that can change the lung field. The thoracic cage boundary was produced using this method from several manually built boundaries. However, this approach has been only evaluated on a small dataset, so I am not sure how the model performs on a wider scale. Furthermore, when various pulmonary abnormalities are present, like those in cases of pneumonectomy, this methodology tends to fail

Chapter Two

2 ANALYSIS AND DESIGN

We will be using Deep Learning Algorithms to detect Pneumonia in this project. There are different methods to diagnose Pneumonia using Chest X-Ray images, but we will examine different algorithms, test overall accuracy, and select the best algorithm for detecting Pneumonia. We will use Flask and HTML to create a Web app to view the results after choosing the best algorithm from the pre-trained ones.

2.1 DATASET DESCRIPTION

The dataset is organized into 2 folders (train, and test) and both train and test contain 3 subfolders (COVID-19, PNEUMONIA, NORMAL). The dataset contains a total of 6432 x-ray images and test data have 20% of total images.



Figure 2-1: Samples of Chest X-ray Images in dataset

Images are collected from various publicly available resources:

- <https://github.com/ieee8023/covid-chestxray-dataset>,
- <https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia>
- <https://github.com/agchung>

2.2 BLOCK DIAGRAM OF THE SYSTEM

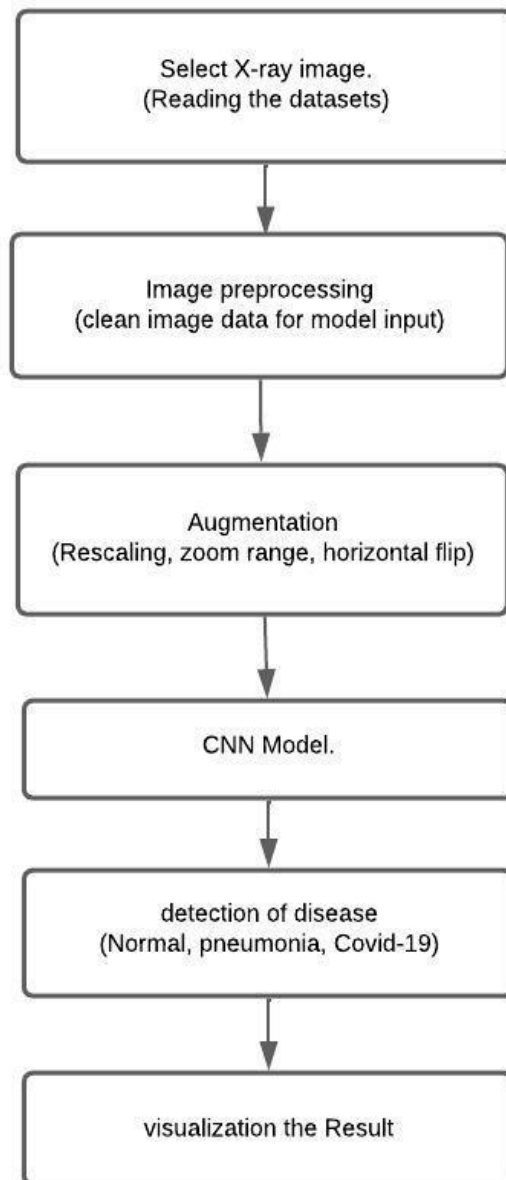


Figure 2-2:Block Diagram of the System

First, we will take the image from the dataset for one of three classes. Then we will apply some image processing algorithms to enhance the image for later use. We will apply augmentation to increase the image by creating a new transformation version of the image. We will pass the images to the CNN model to train for the detection of the disease

and test the model by images not found in the train dataset meaning model not seeing these images.

2.3 USE CASE

In software and systems engineering, a use case is a list of actions or event steps, typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system, to achieve a goal. The actor can be a human, an external system, or time. In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals. Another way to look at it is a use case that describes a way in which a real-world actor interacts with the system. In

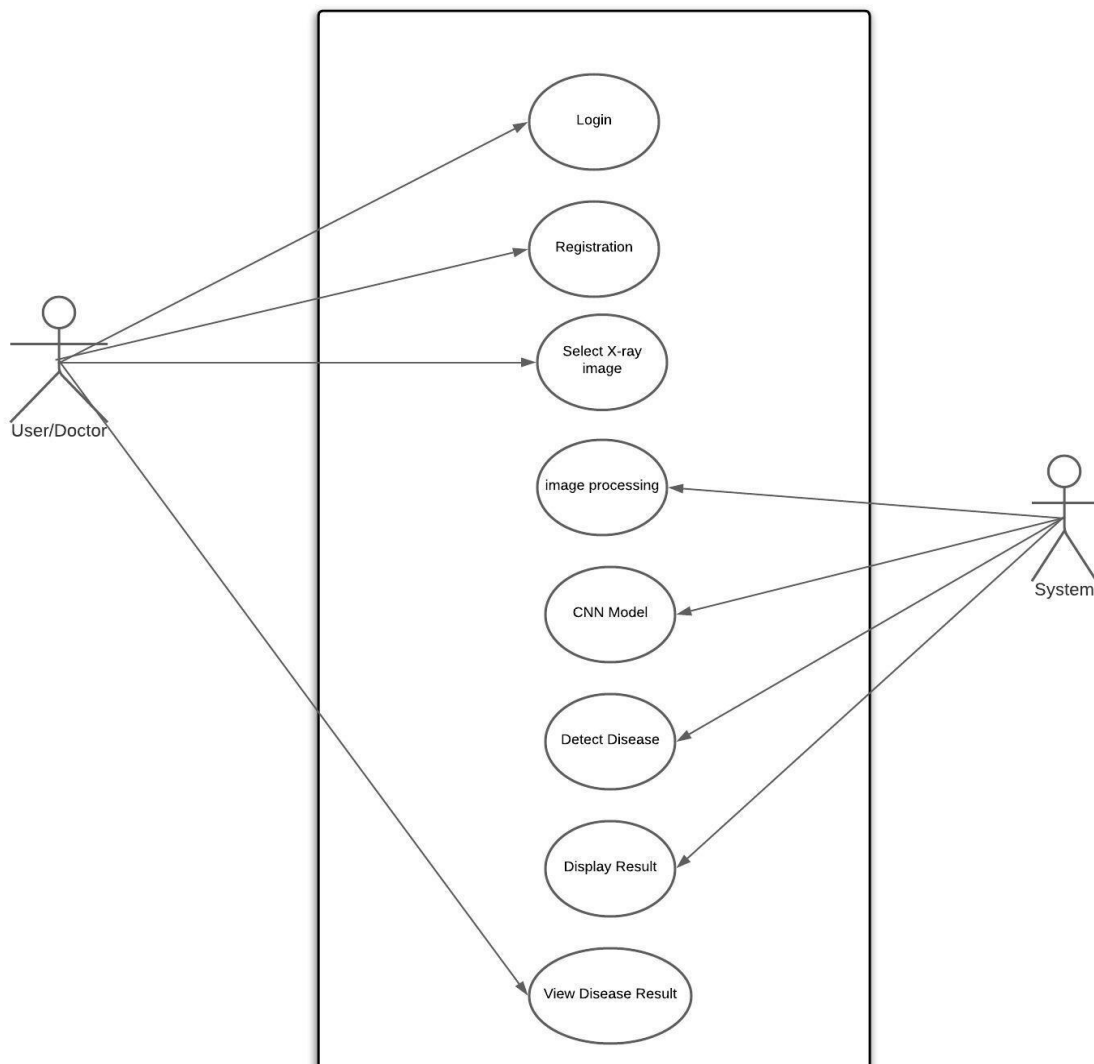


Figure 2-3:Use Case of the system

a system use case, you include high-level implementation decisions. System use cases can be written in both an informal manner and a formal manner.

2.4 DATABASE DIAGRAMS

2.4.1 Entity Relationship Diagram (ERD)

ERD is also known as the Entity-Relationship Model. Peter Chen originally proposed ERD. Entity means any object used to store information and is distinguishable, relationship means connection, and diagram/model means a picture uses to represent something. So, ERD is simply the diagram or model that is used to represent or show the relationship between the entities or data objects that are stored in a database. The main components of the E-R model are an entity, attributes, and relationship. It is a very easy way to represent the database design.

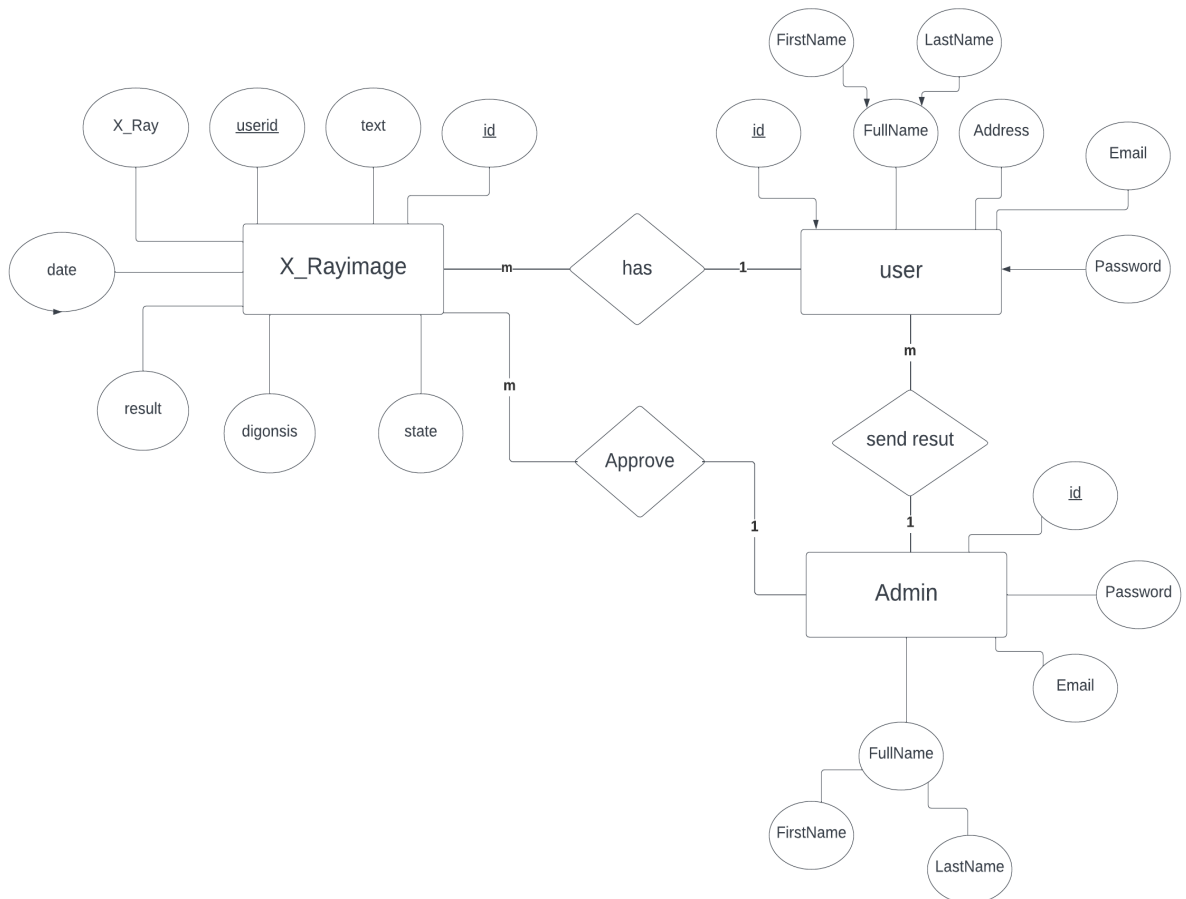


Figure 2-4:Entity Relationship diagram of database

2.4.2 Schema

A schema in a database refers to the blueprint or structure that defines various data tables and the relationships between them. It acts as a logical framework for organizing and storing data, ensuring consistency and accuracy in its retrieval. A well-designed schema helps developers understand how various components of the database are interconnected, facilitating efficient querying and analysis. Key considerations when designing a schema include identifying relevant entities and their attributes, defining relationships between these entities, selecting appropriate data types, and determining primary keys and foreign keys to enforce referential integrity. Schemas can be modified over time to accommodate changing business requirements; however, altering existing schemas may have implications on performance or compatibility with existing applications. Overall, drafting an optimal database schema requires careful planning and consideration of trade-offs between competing objectives such as scalability, flexibility, maintenance ease, or security challenges.

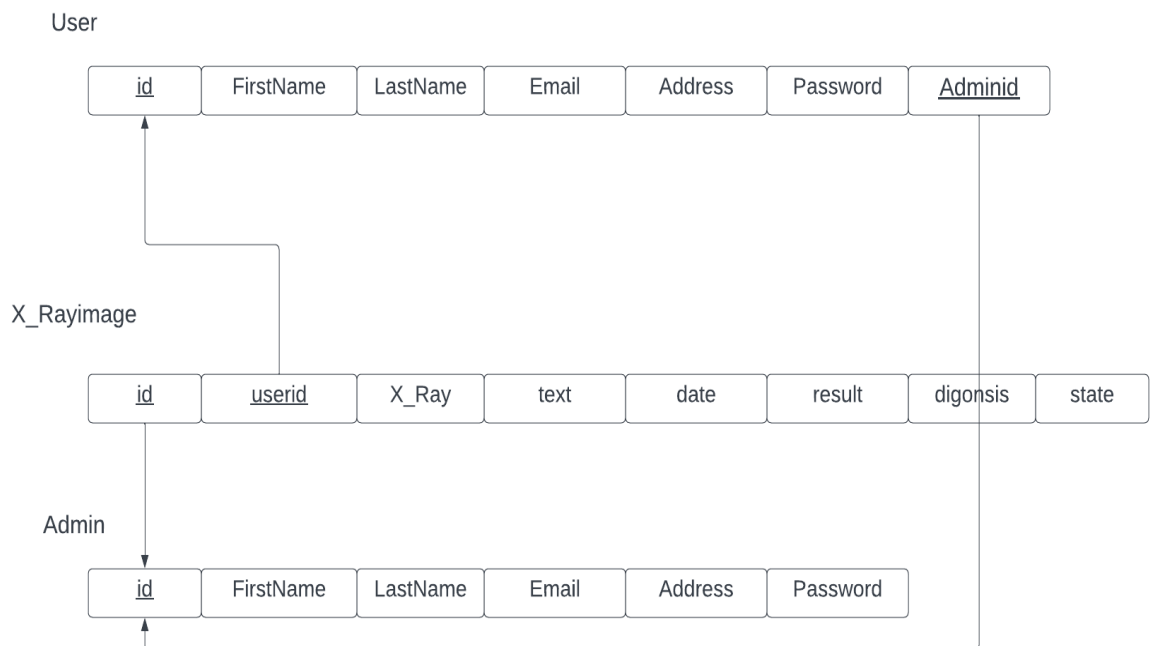


Figure 2-5: Schema

2.5 FLOWCHART OF THE SYSTEM

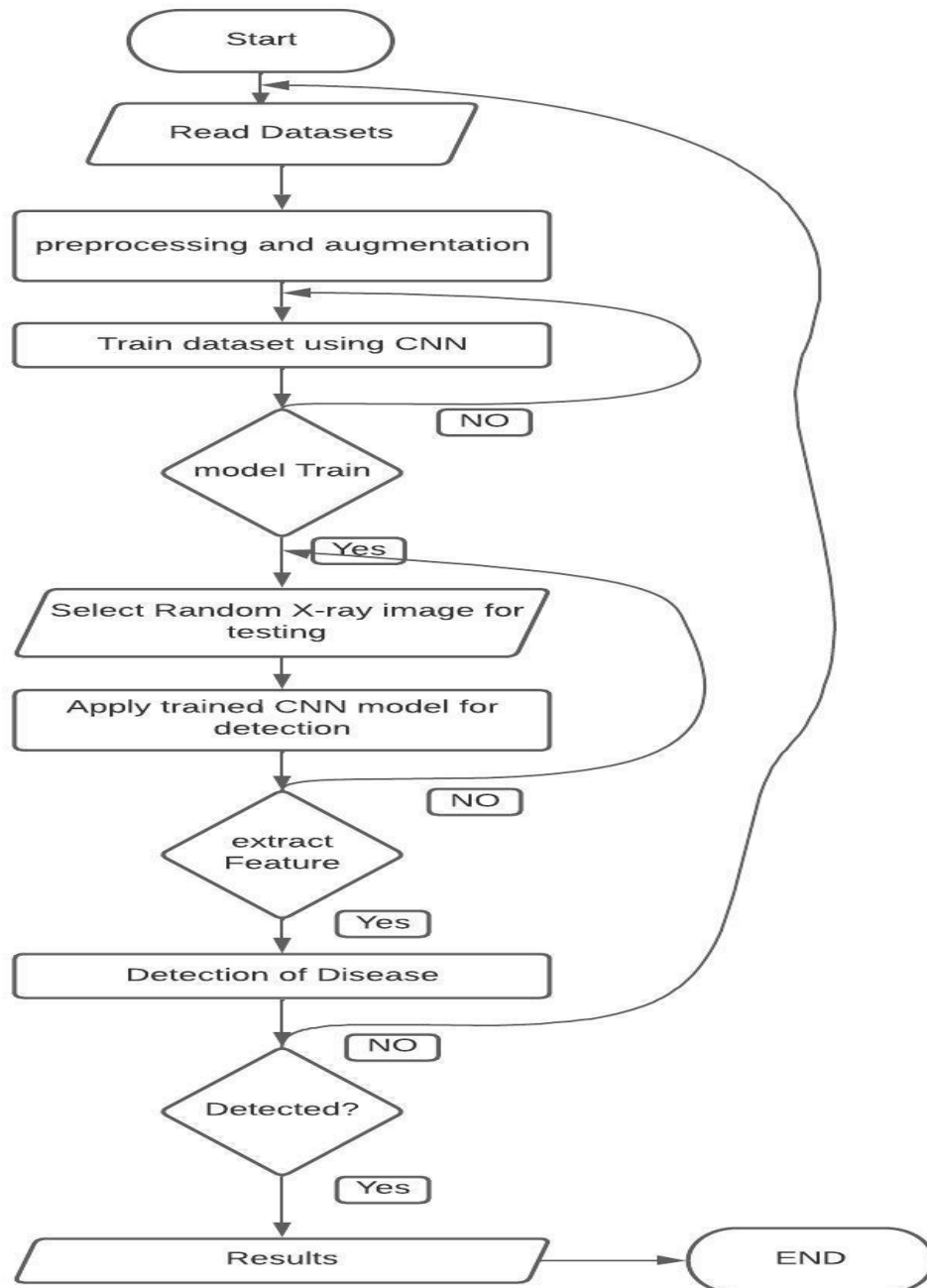


Figure 2-6:Flowchart of the System

2.6 DATA FLOW OF THE SYSTEM

Data flow is an essential component of system design as it enables the efficient transfer and exchange of information between various components within a system. In this

context, data flow defines how data moves through a system, which entities are responsible for controlling the movement, and how data changes when it passes through different stages. The goal of designing a robust data flow is to ensure that the information being exchanged is accurate, timely, and securely handled throughout its journey in the system. This requires careful consideration of factors such as data formats, protocols for transmission/reception, speed of transmission, security measures such as encryption/decryption mechanisms, and audit trails to ensure accountability. By leveraging design principles such as modularity, abstraction, and layering in conjunction with appropriate tools for analysis and testing, developers can design systems with optimal data flows that provide reliable performance while mitigating any risks or potential bottlenecks in transfers.

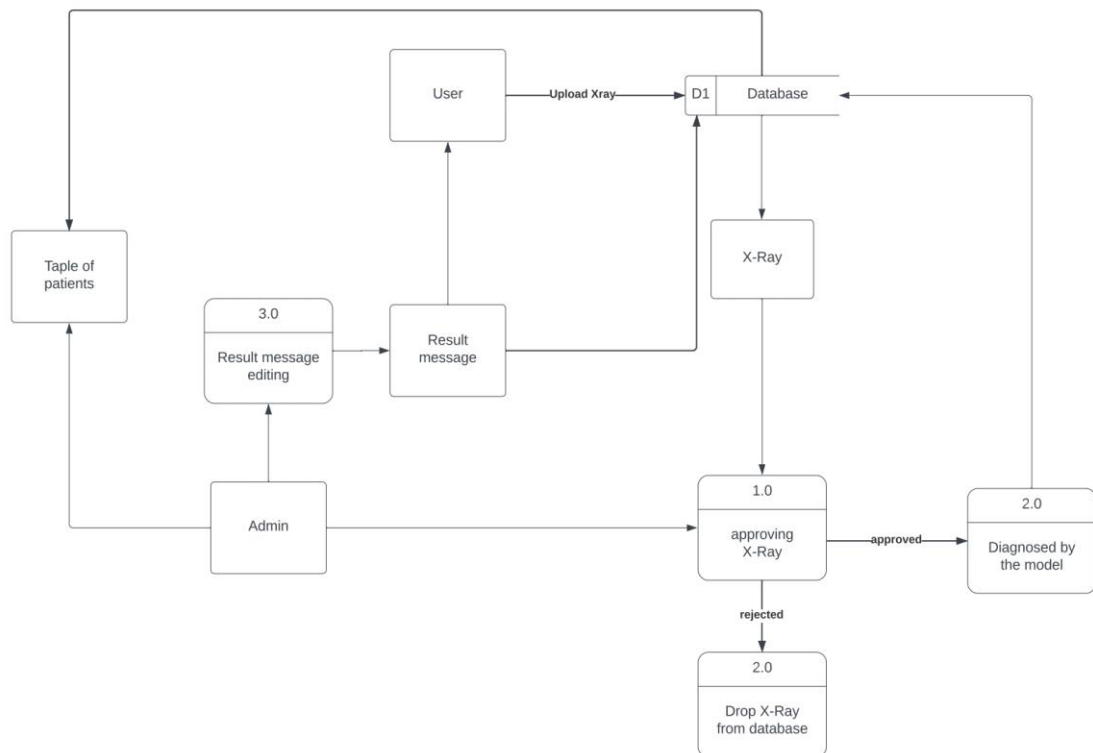


Figure 2-7: Data Flow of the System

2.7 SEQUENCE DIAGRAM OF THE SYSTEM

A sequence diagram is a powerful visual representation that depicts the interactions between objects and the order in which they take place. It is an essential tool for software development teams because it helps them to identify potential issues and inefficiencies by showing how one part of the system interacts with another. Sequence diagrams illustrate complex processes using simple symbols, such as actors, messages, and lifelines, making them easy to understand for all stakeholders involved in software development. Additionally, they can provide insight into how different applications function together in an integrated system. By applying sequence diagramming techniques, developers can create robust systems that are reliable, efficient, and easily maintainable over time. Overall, sequence diagrams are indispensable tools for any software engineering team focused on delivering high-quality products on time and within budget.

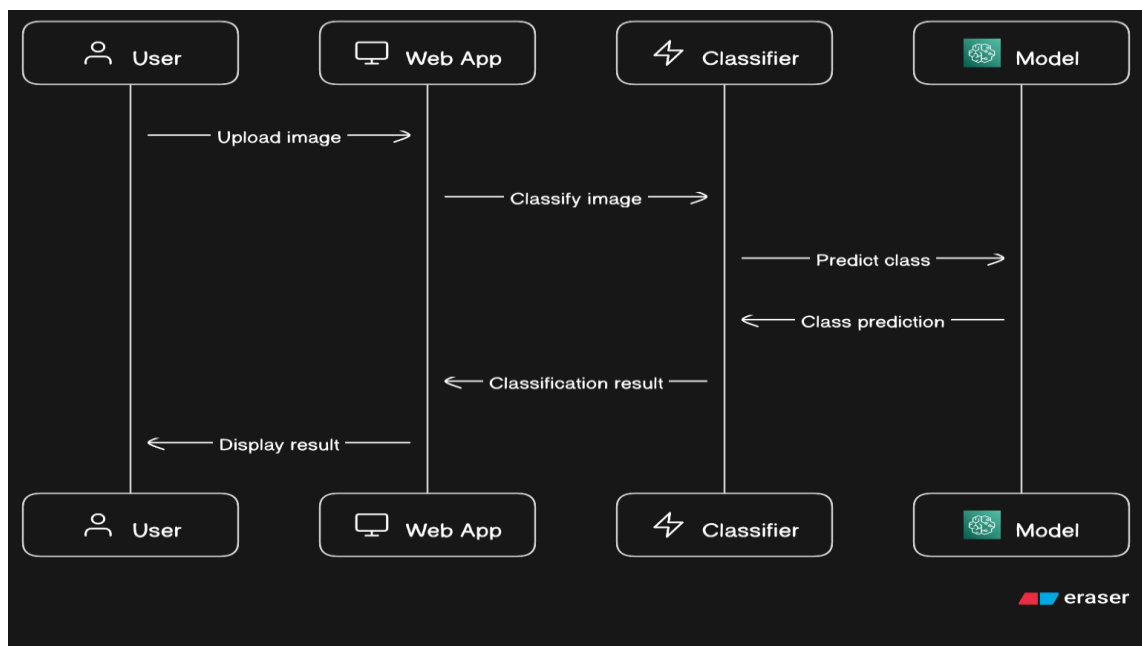


Figure 2-8: Sequence Diagram of the system

Chapter Three

3 DELIVERABLES AND EVALUATION

3.1 INTRODUCTION

The Different Deep Learning Models used in comparing and detecting Pneumonia are CNN_1, CNN_2, DenseNet121, VGG19, ResNet50, and mobileNet.

3.2 IMPORT REQUIREMENTS LIBRARIES

Import requirements libraries as NumPy for testing with matrices. Pandas for reading data. Matplotlib for visualizing model outputs. Sklearn for evaluating models. TensorFlow and Keras for building the model. Os for treating paths of data. Datetime for calculating the time of the training model.



Figure 3-1:requirements libraries

Cv2 for treating with images like resizing it and reading.

3.3 GET AND PROCESS DATA

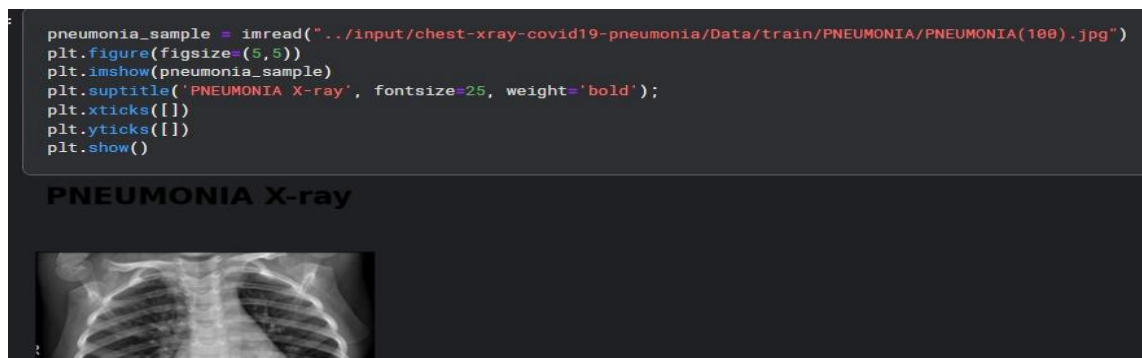


Figure 3-2:Sample image of pneumonia data

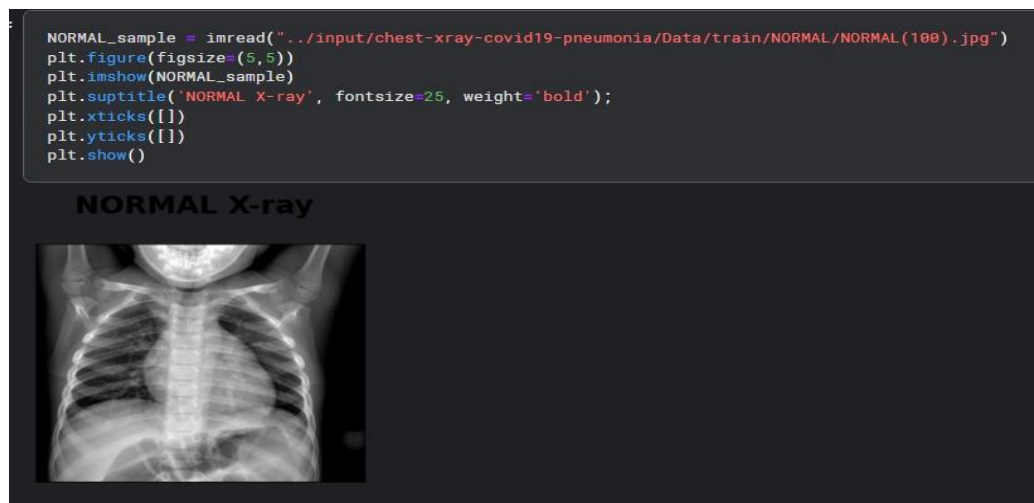


Figure 3-4: Sample image of normal data

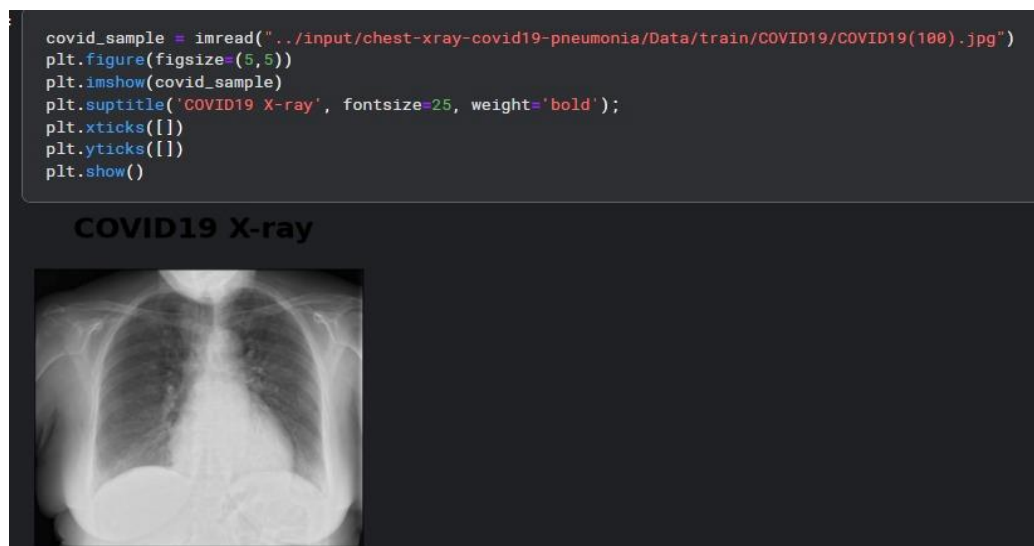


Figure 3-3: Sample image of covid19 data

Pandas DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.

```
[ ] #/content/drive/MyDrive/Colab Notebooks/Pneumonia diagnosis system/dataset/Data/train
train_folder = '/kaggle/input/chest-xray-covid19-pneumonia/Data/train'
x, y = [],[]

for category in os.listdir(train_folder):
    for file in os.listdir(os.path.join(train_folder,category)):
        x.append(os.path.join(train_folder,category,file))
        y.append(category)

train = pd.DataFrame({
    'paths_train':x,
    'class_train':y })
train.head()
```

	paths_train	class_train
0	/kaggle/input/chest-xray-covid19-pneumonia/Dat...	PNEUMONIA
1	/kaggle/input/chest-xray-covid19-pneumonia/Dat...	PNEUMONIA
2	/kaggle/input/chest-xray-covid19-pneumonia/Dat...	PNEUMONIA
3	/kaggle/input/chest-xray-covid19-pneumonia/Dat...	PNEUMONIA
4	/kaggle/input/chest-xray-covid19-pneumonia/Dat...	PNEUMONIA

Figure 3-5: We convert train and test data to the data frame

```
test_folder = '../input/chest-xray-covid19-pneumonia/Data/test'
x2, y2 = [],[]

for category in os.listdir(test_folder):
    for file in os.listdir(os.path.join(test_folder,category)):
        x2.append(os.path.join(test_folder,category,file))
        y2.append(category)

test = pd.DataFrame({
    'paths_test':x2,
    'class_test':y2 })
test.head()
```

	paths_test	class_test
0	../input/chest-xray-covid19-pneumonia/Data/tes...	PNEUMONIA
1	../input/chest-xray-covid19-pneumonia/Data/tes...	PNEUMONIA
2	../input/chest-xray-covid19-pneumonia/Data/tes...	PNEUMONIA
3	../input/chest-xray-covid19-pneumonia/Data/tes...	PNEUMONIA
4	../input/chest-xray-covid19-pneumonia/Data/tes...	PNEUMONIA

Figure 3-6: train data frame of image

. shuffle () function is used to arrange a list of arrays randomly. This `_. shuffle ()` underscore function uses Fisher-Yates Shuffle which is discussed in the below-mentioned article. So, every time we use this the output of this function will be different according to the Fisher-Yates Shuffle.

Parameters: This function accepts a single parameter List. This parameter is used to hold the list of items that will be shuffled.

Return values: The returned value is the new randomized array containing all the elements which are in the original array which is passed.

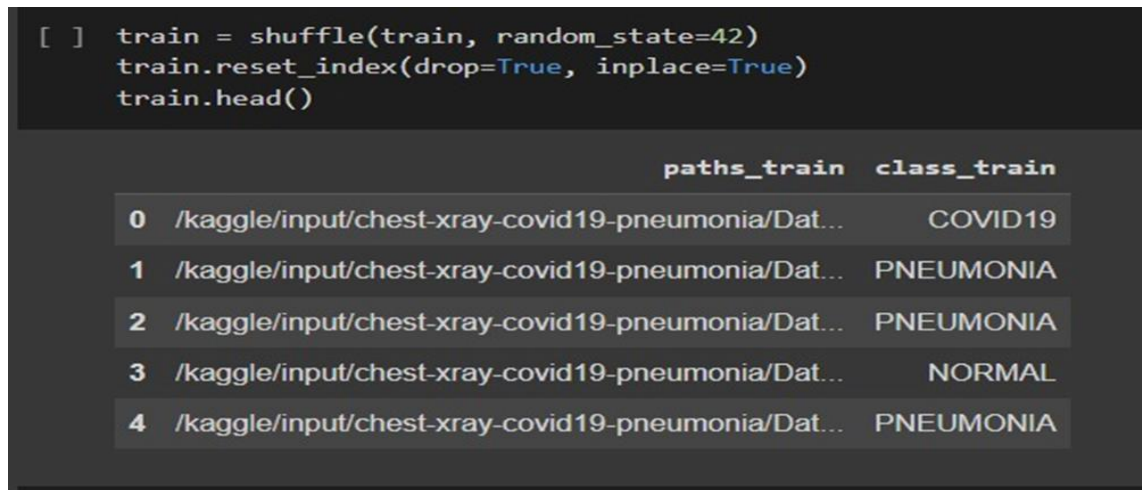


Figure 3-7:the shuffle() function of train data.

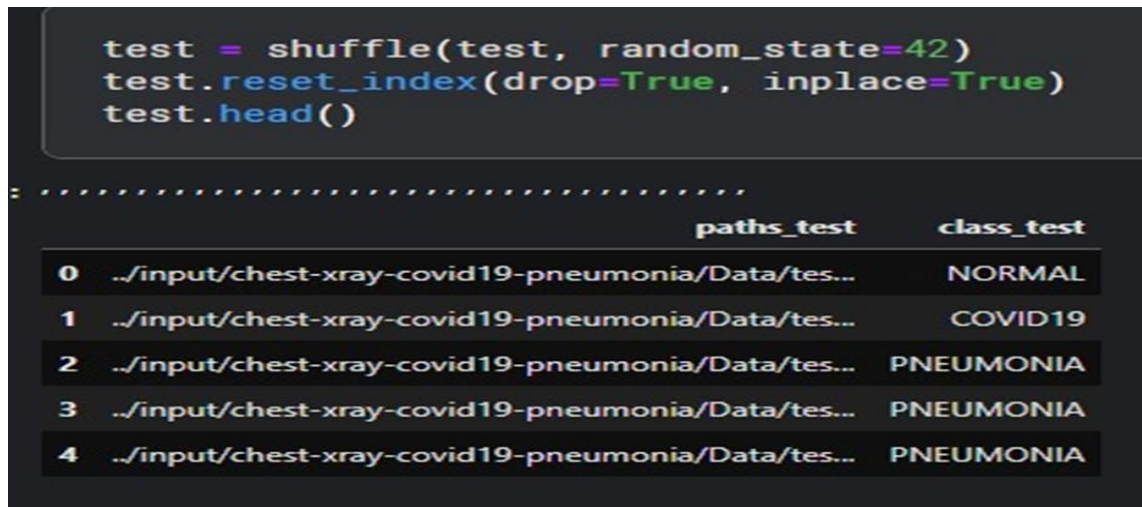


Figure 3-8:the shuffle () function of test data.

3.3.1 Splitting data into train and validation data

3.3.1.1 Train test split

Train Test Split Using Sklearn

The `train_test_split ()` method is used to split our data into train and test sets.

First, we need to divide our data into features (X) and labels (y). The data frame gets divided into `X_train`, `X_test`, `y_train`, and `y_test`. `X_train` and `y_train` sets are used for training and fitting the model. The `X_test` and `y_test` sets are used for evaluating the model

if it is predicting the right outputs/labels. we can explicitly test the size of the train and test sets. It is suggested to keep our train sets larger than the test sets.

```
train, val = train_test_split(train, test_size=0.15, random_state=42, stratify=train['class_train'])
print(train['class_train'].value_counts(dropna=False))
print(val['class_train'].value_counts(dropna=False))
```

1	2905
0	1076
2	391

Name: class_train, dtype: int64

1	513
0	190
2	69

Name: class_train, dtype: int64

Figure 3-9: The train data and test data are used

3.3.1.2 Stratified k-fold

Stratified K-Fold differs from normal K-Fold in that it makes sure that each fold has the same percentage of samples of each class. This is especially useful if your training data is not uniformly distributed.

StratifiedShuffleSplit is a combination of both ShuffleSplit and StratifiedKFold. Using StratifiedShuffleSplit the proportion of distribution of class labels is even between the train and test datasets. The major difference between StratifiedShuffleSplit and StratifiedKFold (shuffle=True) is that in StratifiedKFold, the dataset is shuffled only once in the beginning and then split into the specified number of folds. This discards any chances of overlapping the train-test sets.

However, in StratifiedShuffleSplit the data is shuffled each time before the split is done and this is why there's a greater chance that overlapping might be possible between train-

```
init_time = datetime.datetime.now()
from sklearn.model_selection import StratifiedKFold
kfold=StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
for train_idx, val_idx in list(kfold.split(train['paths_train'], train['class_train'])):
    train_df=train.iloc[train_idx]
    val_df=train.iloc[val_idx]
```

Figure 3-10: Stratified K-Fold

test sets Parameters: n_splits: int, default=10 Number of re-shuffling & splitting iterations. test_size: float or int, default=None

3.3.2 Image Data Generator

Overfitting is caused by having too few samples to learn from, rendering you unable to train a model that can generalize to new data.

Data augmentation takes the approach of generating more training data from existing training samples, by "augmenting" the samples via several random transformations that yield believable-looking images.

The goal is that at training time, our model would never see the same picture twice. This helps the model get exposed to more aspects of the data and generalize better.

In Keras, this can be done by configuring several random transformations to be performed on the images by ImageDataGenerator.

The output images generated by the generator will have the same output dimensions as the input images.

Image Data Generator uses less memory and consumes fewer resources, we do not produce all the data at once using batch by batch.

These are just a few of the options available:

rotation_range is a value in degrees (0–180), a range within which to randomly rotate pictures.

width_shift and height_shift are ranges (as a fraction of total width or height) within which to randomly translate pictures vertically or horizontally.

shear_range is for randomly applying shearing transformations.

zoom_range is for randomly zooming inside pictures.

horizontal_flip is for randomly flipping half the images horizontally relevant when there are no assumptions of horizontal asymmetry (for example, real-world pictures).

fill_mode is the strategy used for filling in newly created pixels, which can appear after a rotation or a width/height shift.

```
datagen = ImageDataGenerator(rescale = 1./255,
                             rotation_range=10,
                             width_shift_range=0.1,
                             height_shift_range=0.1,
                             shear_range=0.1,
                             zoom_range=0.2,
                             fill_mode='nearest',
                             horizontal_flip=True,
                             vertical_flip=False,
                             )

test_datagen = ImageDataGenerator(rescale = 1./255)
```

Figure 3-11:Data augmentation Parameter

we apply damage on the train data to create a new transformation version of train data but for validation or test data, we only apply rescale parameter to Keep the test images as they came but in different scales between 0 and 1 to be faster in computation.

```
train_data = datagen.flow_from_dataframe(train,
                                        x_col="paths_train",
                                        y_col="class_train",
                                        target_size=(image_w,image_h),
                                        color_mode='rgb',
                                        batch_size=batch_size,
                                        class_mode='categorical',
                                        shuffle=True,
                                        num_parallel_calls=AUTOTUNE)

val_data = test_datagen.flow_from_dataframe(val,
                                        x_col="paths_train",
                                        y_col="class_train",
                                        target_size=(image_w,image_h),
                                        color_mode='rgb',
                                        batch_size=batch_size,
                                        class_mode='categorical',
                                        shuffle=False,
                                        num_parallel_calls=AUTOTUNE)

test_data = test_datagen.flow_from_dataframe(test,
                                        x_col="paths_test",
                                        y_col="class_test",
                                        target_size=(image_w,image_h),
                                        color_mode='rgb',
                                        batch_size=batch_size,
                                        class_mode='categorical',
                                        shuffle=False,
                                        num_parallel_calls=AUTOTUNE)

Found 4372 validated image filenames belonging to 3 classes.
Found 772 validated image filenames belonging to 3 classes.
Found 1288 validated image filenames belonging to 3 classes.
```

Figure 3-12: Apply Data Augmentation.

3.4 CREATE MODEL

3.4.1 Cnn1 hidden layer

The construction of a convolutional neural network is a multi-layered feed-forward neural network, made by assembling many unseen layers on top of each other in a particular order.

It is the sequential design that permits CNN to learn hierarchical attributes.

In CNN, some of them are followed by grouping layers and hidden layers are typically convolutional layers followed by activation layers.

The pre-processing needed in a ConvNet is kindred to that of the related pattern of neurons in the human brain and was motivated by the organization of the Visual Cortex.

Convolutional Neural Networks are made up of three types of layers:

Convolutional Layer: It is the main building block of a CNN. It inputs a feature map or input image consisting of a certain height, width, and channels and transforms it into a new feature map by applying a convolution operation. The transformed feature map consists of different heights, widths, and channels based on the filter size, padding, and stride.

Pooling Layer: Pooling layers do the operation of downsampling, where it reduces the size of the input feature map. Two main types of pooling are Max Pooling and Average Pooling.

Fully Connected Layer: Each node in the output layer connects directly to a node in the previous layer.

The `compile()` function configures and makes the model for the training and evaluation process. By calling `compile()` function we prepare the model with an optimizer, loss, and metrics. The `compile()` function takes an argument object as a parameter.

Note: If you call `fit()` or `evaluate()` function on an uncompiled model, then the program will throw an error.

Parameters:

optimizer: It is a compulsory parameter. It accepts an object of `tf.train.Optimizer` or a string name for an `Optimizer`. Following are string names for the optimizers — “sgd,” “adam,” “adamax,” “adadelta,” “adagrad,” “rmsprop,” and “momentum”.

loss: It is a compulsory parameter. It accepts a string value or string array for the type of loss. If our model has multiple outputs, we can use a different loss on each output by passing an array of losses. The loss value that will be minimized by the model will then be the sum of all individual losses. Following are the string name for loss — “meanSquaredError,” “meanAbsoluteError,” etc.

metrics: It is an optional parameter. It accepts a list of metrics to be evaluated by the model during the training and testing phase. Usually, we use `metrics=[‘accuracy’]`. To

specify different metrics for different outputs of a multi-output model, we can also pass a dictionary.

Return Value: Since it prepares the model for training, it does not return anything. (i.e., the return type is void)

3.4.2 Cnn2 hidden layer

Add more layers Dense with 256 neurons and a dropout layer of 0.2 to a fully connected layer.

```
model=Sequential()

model.add(Conv2D(32,(3,3),activation='relu',padding = 'same',input_shape=(image_w,image_h,3),name='conv2d_0'))
model.add(MaxPool2D(2,name='pool_0'))

model.add(Conv2D(64,(3,3),activation='relu',padding = 'same',name='conv2d_1'))
model.add(MaxPool2D(2,name='pool_1'))

model.add(Conv2D(128,(3,3),activation='relu',padding = 'same',name='conv2d_2'))
model.add(MaxPool2D(2,name='pool_2'))

model.add(Conv2D(256,(3,3),activation='relu',padding = 'same',name='conv2d_3'))
model.add(MaxPool2D(2,name='pool_3'))

model.add(Conv2D(512,(3,3),activation='relu',padding = 'same',name='conv2d_4'))
model.add(MaxPool2D(2,name='pool_4'))

model.add(Flatten(name='flatten'))

model.add(Dense(512,activation='relu',name='Dense_0'))
model.add(Dropout(0.4,name="dropout_1"))

model.add(Dense(256,activation='relu',name='Dense_1'))
model.add(Dropout(0.2,name="dropout_2"))

model.add(Dense(3,activation='softmax',name='output'))

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

Figure 3-13:CNN-2 hidden layer

3.4.3 Mobilenetv2

As the name applied, the Mobile Net model is designed to be used in mobile applications, and it is TensorFlow's first mobile computer vision model.

Mobile Net uses depth-wise separable convolutions. It significantly reduces the number of parameters when compared to the network with regular convolutions with the same depth in the nets. This results in lightweight deep neural networks.

This code defines a multi-layer neural network using TensorFlow's Keras library. The network is based on MobileNetV2, a pre-trained model, and is used for image classification.

Firstly, the MobileNetV2 model is loaded with the 'imagenet' weights, which are pre-trained on the ImageNet dataset. The input shape is specified as (image_w, image_h, 3), which means that the model takes 3-channel images of size (image_w, image_h) as input.

Then, the layers of the pre-trained model are looped over, and the last six layers are set to be trainable, meaning their weights will be updated during the training process.

After that, a new model is created by adding a few more layers to the pre-trained model. The output of the pre-trained model is fed into an AveragePooling2D layer, which computes the average of the values in a 2D spatial window. The output is then flattened into a 1D tensor and fed into a dense layer with 512 neurons and a ReLU activation function. The output of this layer is passed through a dropout layer with a rate of 0.15, which randomly sets 15% of the output to zero during each training iteration to prevent overfitting.

Finally, a dense output layer with 3 neurons and a softmax activation function is added, and the model is compiled with the Adam optimizer and categorical cross-entropy loss.

The summary of the model is printed to give a summary of its architecture, including the number of parameters and the shapes of the inputs and outputs of each layer.

```
from tensorflow.keras.applications import MobileNetV2
pretrained_mobilenet = MobileNetV2(input_shape=(image_w, image_h, 3), weights='imagenet', include_top=False)
for layer in pretrained_mobilenet.layers[-6:]:
    layer.trainable = True

x = pretrained_mobilenet.output
x = AveragePooling2D(name="averagepooling2d")(x)
x = Flatten(name="flatten")(x)
x = Dense(512, activation="relu", name="dense_0")(x)
x = Dropout(0.15, name="dropout_0")(x)
model_out = tf.keras.layers.Dense(3, activation='softmax', name="output")(x)

model = Model(inputs=pretrained_mobilenet.input, outputs=model_out)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

model.summary()
```

Figure 3-14: MobileNet model.

3.5 TRAIN MODEL

Early stopping: stop training when a monitored metric has stopped improving.

monitor: Quantity to be monitored and make it the val_loss.

patience: Number of epochs with no improvement after which training will be stopped and make its value 10.

mode: One of {"auto", "min", "max"}. In min mode, training will stop when the

quantity monitored has stopped decreasing.

ModelCheckpoint callback is used in conjunction with training using the model. fit () to save a model or weights (in a checkpoint file) at some interval, so the model or weights can be loaded later to continue the training from the state saved.

Reduce the learning rate when a metric has stopped improving. Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.

```
early_stopping = EarlyStopping(monitor="val_loss", patience=10, mode="min")

checkpoint = ModelCheckpoint("loss-{val_loss:0.4f}.h5", monitor="val_loss", verbose=1,
                             save_best_only=True, save_weights_only=True, mode="min")

learning_rate_reduction = ReduceLROnPlateau(monitor="val_loss", factor=0.1, patience=5,
                                              min_lr=1e-7, verbose=1, mode="min")
```

Figure 3-15:callback functions

fit () method is used to train the model for the fixed number of epochs (iterations on a dataset).

Parameters: This method accepts the following parameters:

x: It is tf. A tensor that contains all the input data.

y: It is tf. A tensor that contains all the output data.

args: It is an object type; its variables are as follows:

batchSize: It defines the number of samples that will propagate through training.

epochs: It defines iteration over the training data arrays.

verbose: It helps in showing the progress for each epoch.

If the value is 0 – It means no printed message during the fit () call. If the value is 1 – It means in Node-js, it prints the progress bar. In the browser, it shows no action. Value 1 is the default value. 2 – Value 2 is not implemented yet.

callbacks: It defines a list of callbacks to be called during training. Variable can have one or more of these callbacks onTrainBegin (), onTrainEnd (), onEpochBegin(), onEpochEnd(), onBatchBegin(), onBatchEnd(), onYield().

validationSplit: It makes it easy for the user to split the training dataset into train and validation.

For example: if its value is validation-Split = 0.5, it means to use the last 50% of data before shuffling for validation.

validation data: It is used to give an estimate of the final model when selecting between final models.

shuffle: This value defines the shuffle of the data before each epoch. it has not affected when stepsPerEpoch is not null.

initialEpoch: It is a value-defined epoch at which to start training. It is useful for resuming a previous training run.

stepsPerEpoch: It defines the number of batches of samples before declaring one epoch finished and starting the next epoch. It is equal to 1 if not determined.

validation steps: It is relevant if stepsPerEpoch are specified. The total number of steps to validate before stopping.

Returns: Promise< History >

This code trains a machine-learning model using the fit method of the model object. The fit method takes the training data (train_data), validation data (val_data), batch size (batch_size), number of epochs (epochs), number of steps per epoch (train_steps), number of validation steps (valid_steps), a list of callbacks (checkpoint, early_stopping, learning_rate_reduction), and verbosity level (verbose=1).

The training process is timed using the datetime module in Python. The time before training starts is recorded as init_time, and the time after training is completed is recorded as required_time. The difference between these two times gives the total time required for the training process, which is printed at the end of the code.

000

```
init_time = datetime.datetime.now()

history = model.fit(
    train_data,
    validation_data=val_data,
    batch_size=batch_size,
    epochs=epochs,
    steps_per_epoch=train_steps,
    validation_steps=valid_steps,
    callbacks=[
        checkpoint, |
        early_stopping,
        learning_rate_reduction],
    verbose=1,
)

required_time = datetime.datetime.now() - init_time
print(f'\nRequired time: {str(required_time)}\n')
```

Figure 3-16:Fit the model.

3.6 EVALUATE MODEL

"Evaluating a model's performance" is a crucial step in the machine learning process, and there are several techniques used to do so.

Accuracy: The accuracy of a model is the fraction of correctly classified samples over the total number of samples. It gives an overall idea of the model's performance, but it can be misleading when the classes are imbalanced, meaning one class has many more samples than the other.

Classification Report: The classification report is a detailed report of the model's performance on the test data, providing information about precision, recall, f1-score, and support for each class.

Precision: The precision of a class is the fraction of samples predicted as positive that are positive.

Recall: The recall of a class is the fraction of samples of the class that are correctly predicted as positive.

F1-Score: The F1-Score is the harmonic means of precision and recall, and it balances the two metrics.

Support: The support of a class is the number of samples in the test set that belong to that class.

Confusion Matrix: The confusion matrix is a table that shows the number of correct and incorrect predictions made by the model. The rows represent the actual class, and the columns represent the predicted class. It is a useful tool for analyzing the performance of a binary classifier.

These techniques are commonly used to evaluate the performance of a model and to understand its strengths and weaknesses. They provide insights into the model's ability to generalize to new data and to make accurate predictions, which can inform further model development.

3.6.1 MobileNetV2

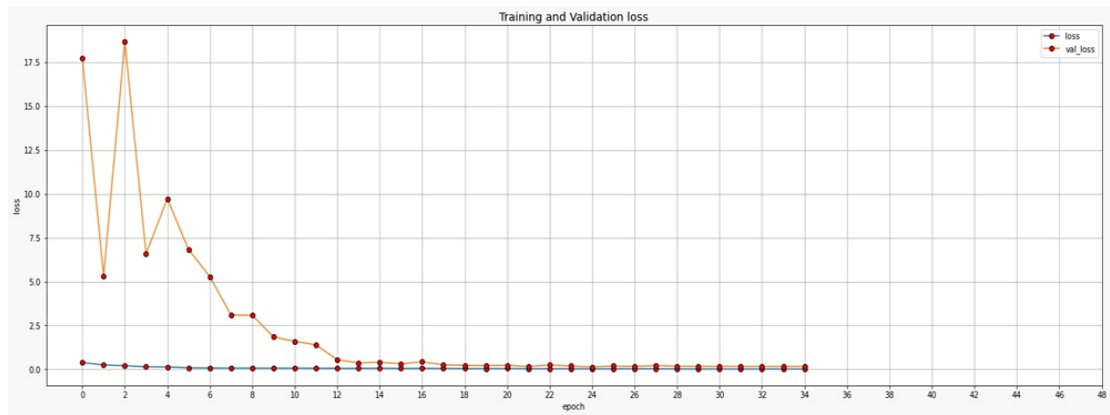


Figure 3-17:MobileNetV2 loss

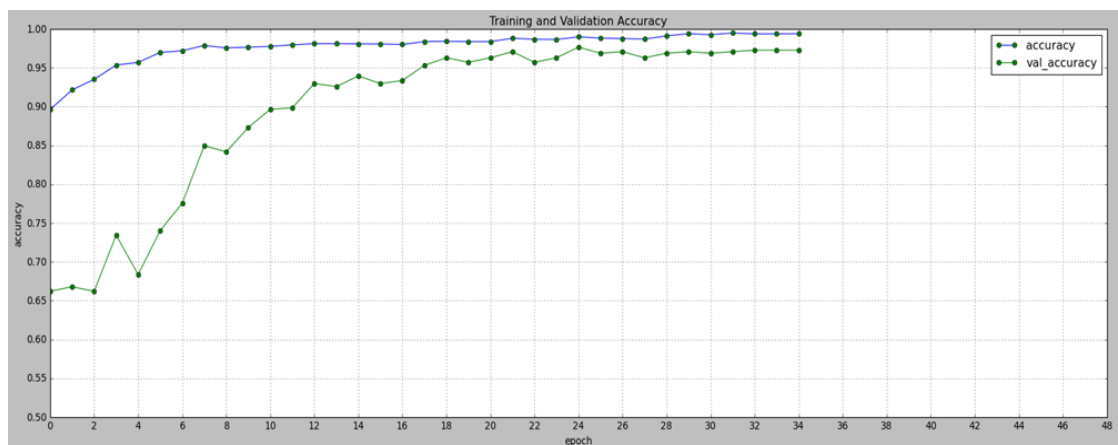


Figure 3-18:MobileNetV2 accuracy

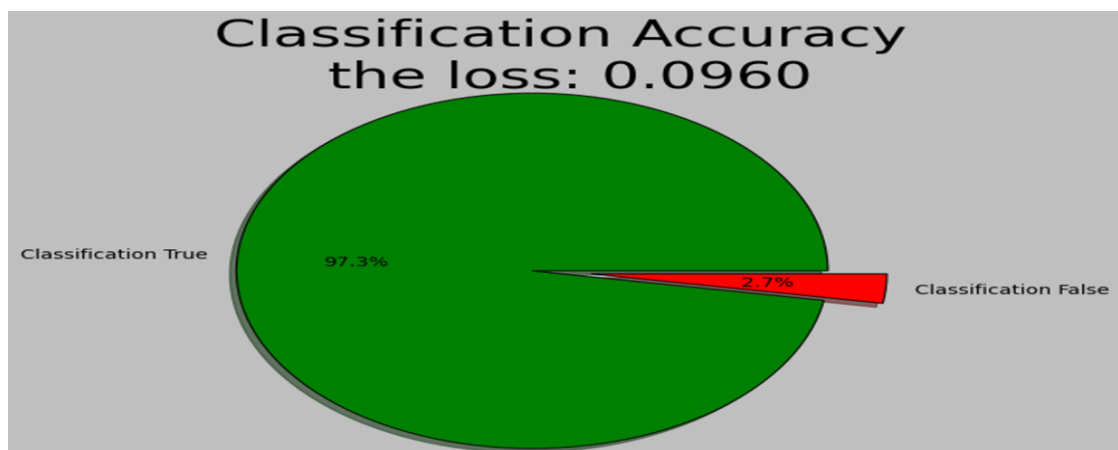


Figure 3-19:MobileNetV2 classification accuracy

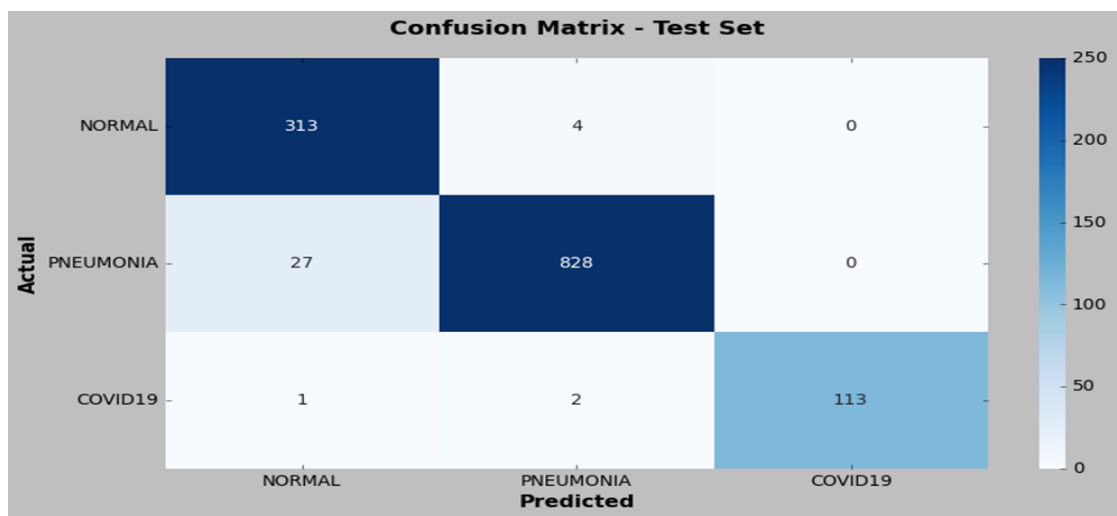


Figure 3-20: MobileNetV2 CM

Classification report for MobileNetV2

classes	precision	recall	F1-score
normal	0.89	0.91	0.90
pneumonia	0.97	0.96	0.96
covid19	0.99	0.97	0.98

Figure 3-21: Classification report for MobileNetV2

Table 1: Summary of Accuracy

Summary	
Model	Accuracy
CNN1 hidden layer	96%
CNN2 hidden layer	95%
CNN1 with k fold	96%
CNN2 with k fold	95.7%
VGG19	94.9%
VGG19 with k fold	92.97%
MobileNetV2	97.3%
MobileNetV2 with k fold	92.73%
ReseNet152V2	95.1%
ReseNet152V2 with k fold	95.8%
DenseNet201	95.4%

3.7 MAKE PREDICTIONS

Grad-CAM (Gradient-weighted Class Activation Mapping) is a deep learning visualization technique used to understand the decision-making process of a Convolutional Neural Network (CNN). It provides a heatmap highlighting the regions in the input image that contributed the most towards a certain class prediction.

Grad-CAM works by combining the activations of the final convolutional layer of the CNN with the gradients of the target class concerning these activations. This allows it to highlight the regions in the input image that contribute the most to the

target class prediction.

To obtain the heatmap, the following steps are taken:

Obtain the activations of the final convolutional layer for a given input image.

Compute the gradient of the target class prediction concerning the activations of the final convolutional layer.

Average the gradients across all channels to obtain a weight for each activation map in the final layer.

Weigh the activations of each channel by the corresponding gradient weights to obtain a weighted sum.

Up-sample the weighted sum to the size of the input image to obtain a final heatmap.

Grad-CAM can be used to understand the model's decisions, identify weaknesses in the model, and improve its accuracy by focusing on the regions that the model has difficulty with. It can also be used to generate explanations for predictions made by the model, making it useful for applications that require trust and transparency.

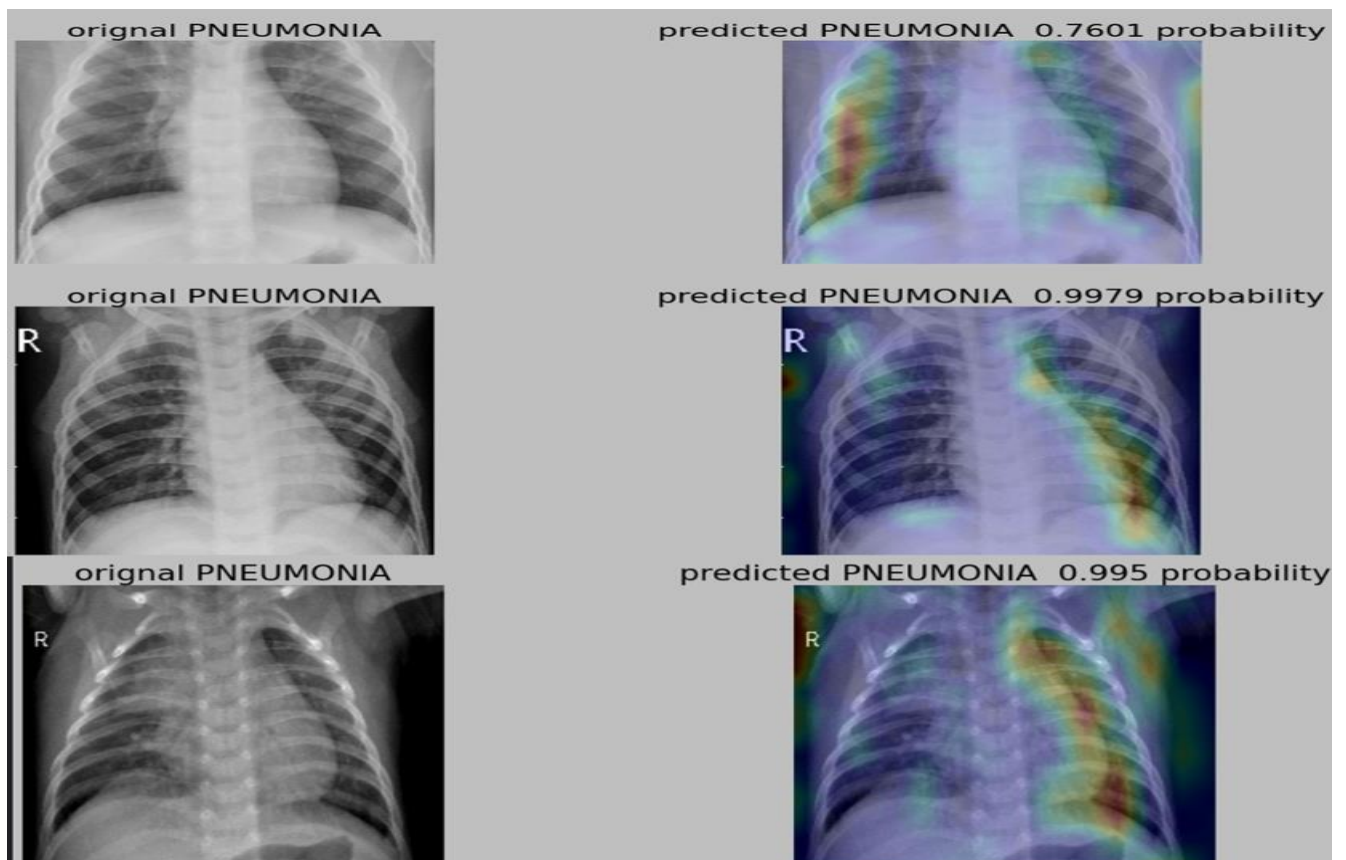


Figure 3-22: Make Prediction with Grad-CAM

3.8 SYSTEM IMPLEMENTATION

Web systems are complex software applications designed to run on the Internet. They consist of various components, including web servers, databases, and client-side applications. A web system can be used to provide a range of services, such as hosting websites, managing online transactions, and providing real-time data access.

One of the key benefits of web systems is their ability to be accessed from anywhere, at any time. This makes them ideal for businesses and organizations that operate across multiple locations or have remote employees. Web systems are also able to scale up or down quickly, making them flexible and cost-effective.

To build a successful web system, it's important to have a clear understanding of the requirements and goals of the system. This includes understanding the target audience, the types of devices and web browsers they will be using, and the desired performance metrics. With these factors in mind, a well-designed web system can provide a seamless and engaging user experience, while also meeting business objectives.

- We used HTML, CSS, and JavaScript to design our system.

HTML, CSS, and JavaScript are the three fundamental technologies used in creating web pages and web applications. HTML (Hypertext Markup Language) is the foundation of every web page, providing the structure and content. CSS (Cascading Style Sheets) is used to style and present the HTML content, making it visually appealing. JavaScript is a programming language that adds interactivity and dynamic functionality to web pages. Together, these three technologies work in harmony to create engaging and user-friendly web experiences. Whether you're creating a simple web page or a complex web application, HTML, CSS, and JavaScript are essential tools for every web developer.

- We used Database and Flask in the back end of our system.

A database is an essential component of web back-end development, allowing for the storage, management, and retrieval of data necessary for web applications. When it comes to choosing a database for web back-end development, various factors such as scalability, security, and performance need to be considered. Common database management systems used in web back-end development include MySQL, PostgreSQL, and MongoDB. MySQL is a popular choice for web developers due to its open-source

nature, reliability, and easy integration with other technologies, the choice of a database system for web back-end development depends on the specific requirements of the application, its expected level of traffic, and the developer's expertise.

Flask is a popular Python-based web framework used for developing web applications and APIs. Flask is known for its simplicity, flexibility, and ease of use, making it an excellent choice for developers who want to quickly build and deploy web applications. Flask provides a wide range of features such as URL routing, request handling, templating, and session management, making it an ideal framework for developing web back-end systems. Flask is also lightweight and has a modular architecture, allowing developers to use only the required components and extensions rather than being forced to adhere to a strict framework structure. Flask is highly extensible and can be easily integrated with other Python packages and libraries, making it a powerful tool for web development. Flask's popularity and community support mean that there are numerous resources available for developers to learn and get help, including extensive documentation, tutorials, and online forums. Overall, Flask is a versatile and user-friendly framework that offers a fast and efficient way to build robust web applications and API

3.8.1 Home Page

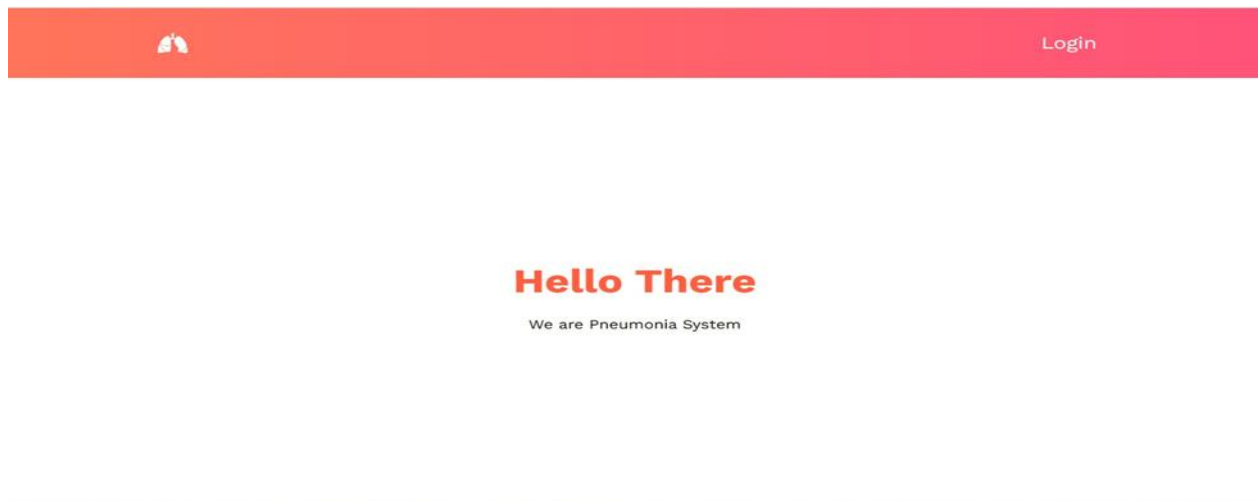


Figure 3-23: Home Page

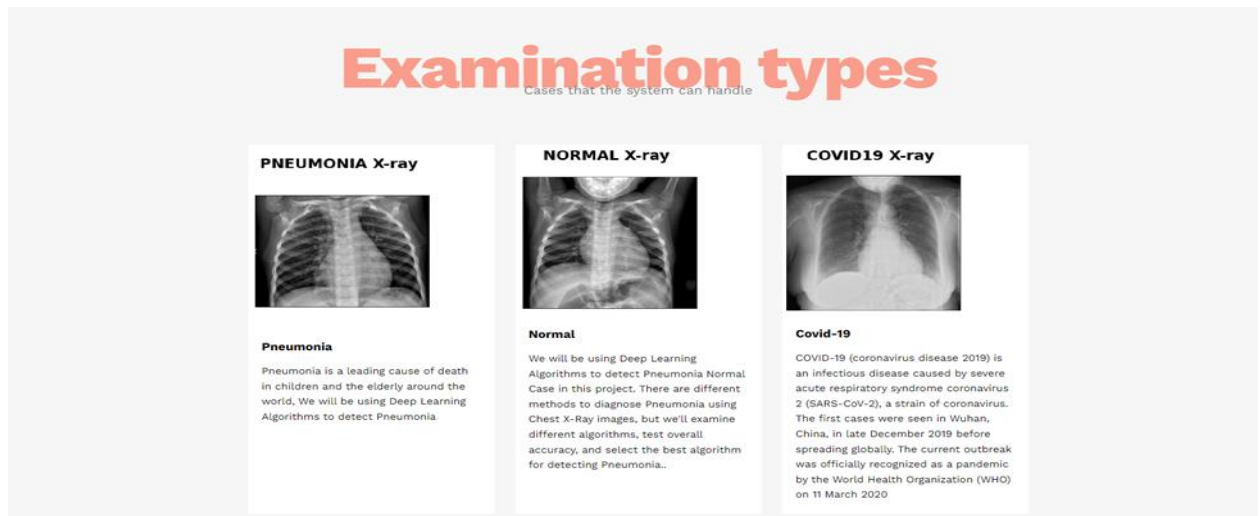


Figure 3-24: Home Page

The Home page in our system consists of a header that contains a button to access the registration and login page in the system and also many small articles that talk in detail about the entire project and the general idea that the project deals with and also contains a part containing contact information with the administrators of the system.

3.8.2 Sign In Page

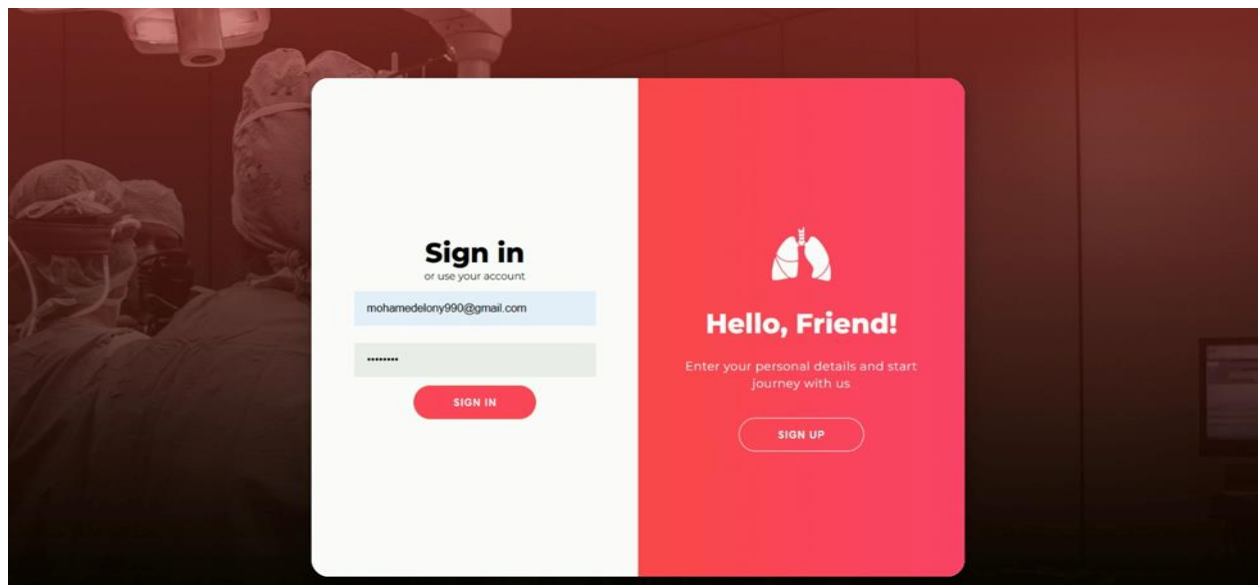


Figure 3-25: Sign-in Page

After the user is directed to this page, he will have two choices: Either he is an old user of the system, and therefore he will log in to complete the service he wants. As for a

new user, then he will press the register button to go to the Sign Up page and create a new email to become a user of the system.

The page consists of two parts, the part on the left contains two fields, one for writing the email and the second for writing the password, and then you press the Sign In button. As for the part on the right, it contains the Sign-Up button, to be directed to the Sign-Up page.

3.8.3 Sign Up Page

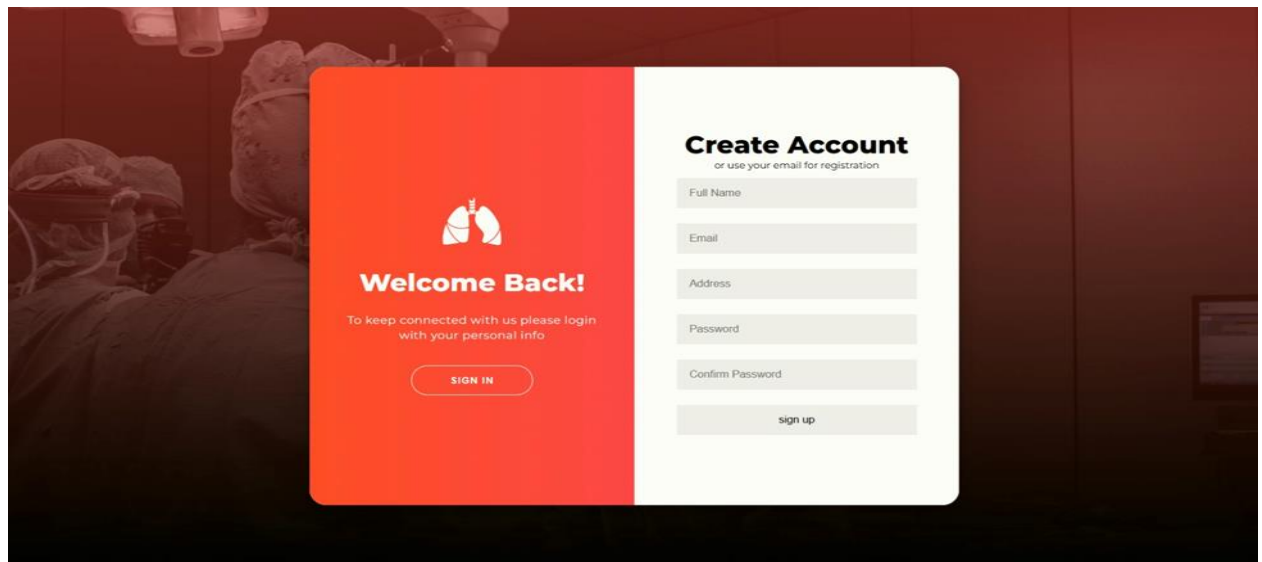


Figure 3-26: Sign-Up Page

On this page, you can create a new email to start accessing the special services.

The method to register a new email must be to fill in all the fields on the right part of the page, which is that you write your name correctly and write your email for communication, the address, and your password, and confirm it, and after completing and making sure that you wrote your data correctly, you press the Sign Up button and so on. You have an email on the system.

3.8.4 Update Profile Page



The screenshot shows a web application interface for updating a user profile. The background is a dark, semi-transparent image of a surgical team in an operating room. Overlaid on this is a bright pink rectangular form titled "Update Profile" in white text. The form contains five white input fields with red borders, labeled "Full Name", "Email", "Address", "Password", and "Confirm Password". Below these fields is a white button labeled "Update". In the top right corner of the pink form, there is a small white button labeled "BACK".

Figure 3-27: Update Profile

On this page, you can edit your email information.

The method of modification with the new data must be filled in the field to be modified with the new data, which is to write your name correctly and write your e-mail to communicate, the address and your password and confirm it, and then when you complete your data and make sure that it is written correctly, you press the update button and this is how the old data is replaced with the data modified

3.8.5 Upload Photo Page



Figure 3-28: Upload Photo Page

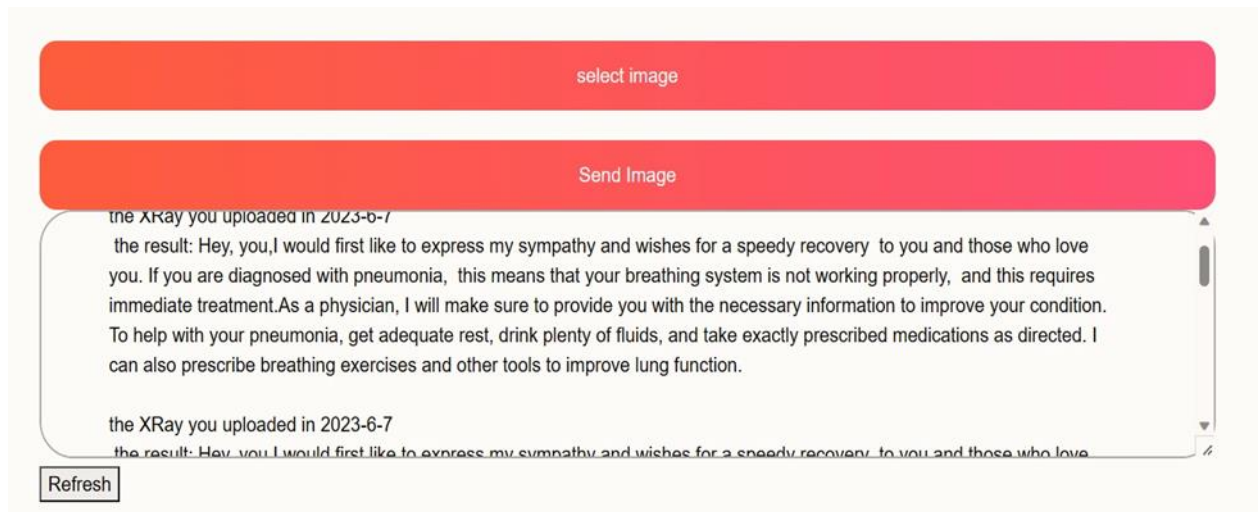


Figure 3-29: Upload Photo Page

On this page, you can upload your X-ray image to the system to be examined by pressing the select image button, and then a window will appear for you to go to your X-ray path on your device and select it, and then press the send image button to send the image to be examined and about it, a message will appear with details The date of sending your photo and the date of the examination, and that you will be contacted via e-mail to receive the results of the examination.

3.8.6 Admin Page

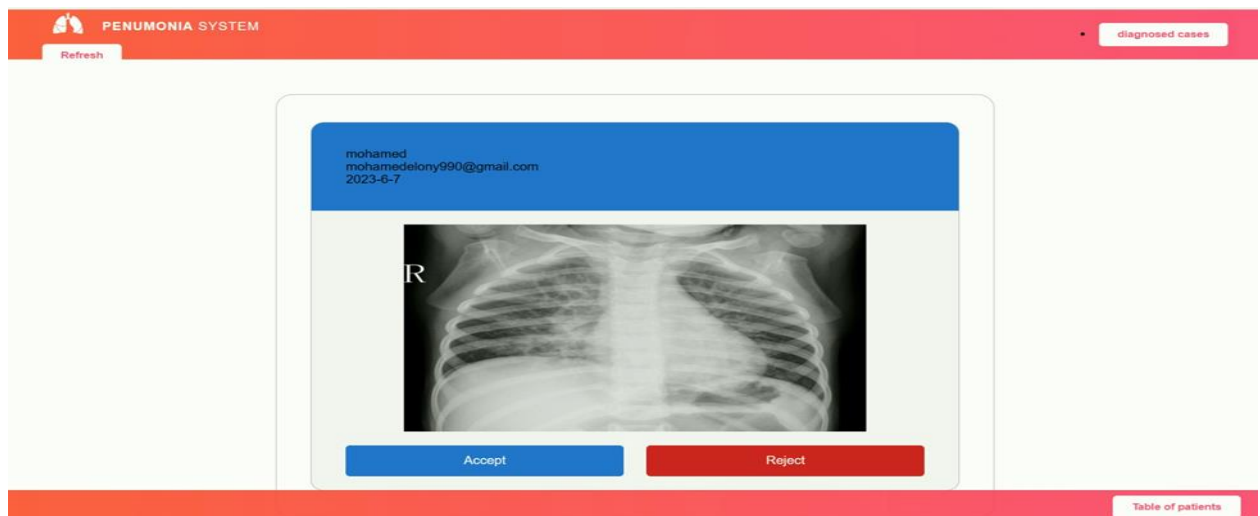


Figure 3-30: Admin Page

On this page, the controller is the admin, and this page can be called the undiagnosed cases page, because when the x-rays are uploaded, they go to this page for the admin to

determine whether the x-rays are scannable, and therefore the x-rays are accepted, but if they are not scanned, the x-rays are rejected.

In each x-ray on the system, the patient's name, his e-mail, and the date the x-rays were sent to the system will be indicated on the x-ray frame.

3.8.7 Diagnosed Cases Page

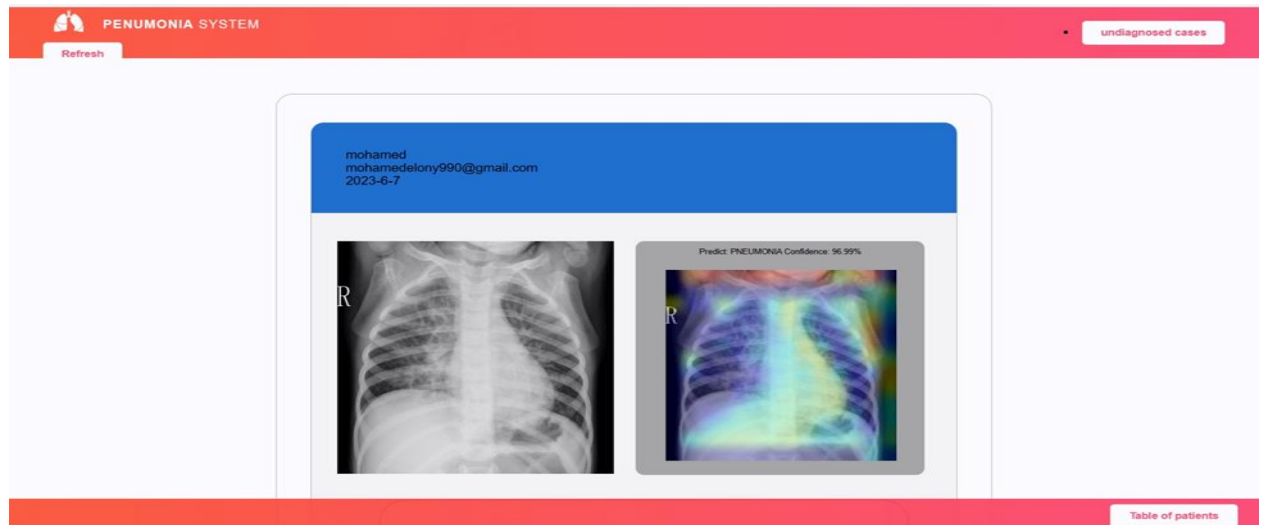


Figure 3-31: Diagnosed Cases Page

On this page, the x-rays that have been examined are saved to be sent to the patient's e-mail, and the e-mail, the patient's name, and the date of the examination are indicated on the x-ray frame. This page is also called Diagnosed Cases and they are arranged in order of screening.

3.8.8 Table Of Patients


 PNEUMONIA SYSTEM Refresh BACK					
FullName	Email	XRay	Date	Diagnosis	Result
mohamed	mohamedelony990@gmail.com	10/images/WhatsApp Image 2023-04-29 at 5.52.40 PM.jpeg	2023-6-7	Predict: PNEUMONIA Confidence: 99.54%	10/results/WhatsApp Image 2023-04-29 at 5.52.40 PM.jpeg
mohamed	mohamedelony990@gmail.com	10/images/WhatsApp Image 2023-04-29 at 5.52.40 PM.jpeg	2023-6-7	Predict: PNEUMONIA Confidence: 99.54%	10/results/WhatsApp Image 2023-04-29 at 5.52.40 PM.jpeg
mohamed	mohamedelony990@gmail.com	10/images/WhatsApp Image 2023-04-29 at 5.52.36 PM.jpeg	2023-6-7	Predict: PNEUMONIA Confidence: 96.99%	10/results/WhatsApp Image 2023-04-29 at 5.52.36 PM.jpeg
mohamed	mohamedelony990@gmail.com	10/images/WhatsApp Image 2023-04-29 at 1.33.32 PM.jpeg	2023-6-7	Predict: PNEUMONIA Confidence: 96.99%	10/results/WhatsApp Image 2023-04-29 at 1.33.32 PM.jpeg
mohamed	mohamedelony990@gmail.com	10/images/WhatsApp Image 2023-04-29 at 5.52.36 PM.jpeg	2023-6-7	Predict: PNEUMONIA Confidence: 96.99%	10/results/WhatsApp Image 2023-04-29 at 5.52.36 PM.jpeg

Figure 3-32: Table of Patients

On this page, the data of all the cases on the system are displayed, the patient's name, the path of the x-rays recorded on his e-mail, his e-mail, the date the x-rays were uploaded to the system, the diagnosis, and the result are displayed.

Chapter Four

4 CONCLUSION

Pneumonia is a contagious illness that has plagued mankind for millennia. Despite substantial technological advancements, pneumonia remains one of the top ten causes of mortality worldwide. To treat pneumonia, it is vital to act quickly and accurately. One of the cheapest and most regularly used diagnostic tools for detecting pneumonia and other lung problems is a chest X-ray. We have proposed a novel tool that can assist enhance the diagnosis accuracy of pulmonary problems from chest radiographs thanks to startling breakthroughs in modern deep-learning algorithms. We used the most prevalent deep learning method known as a convolutional neural network to implement computer-aided diagnostics of pneumonia throughout this study (CNN).

Chapter Five

5 REFERENCES

- 1 G. d. Melo, S. O. Macedo, S. L. Vieira and L. G. Leandro Oliveira, "Classification of images and enhancement of performance using parallel algorithm to detection of pneumonia," 2018 IEEE International Conference on Automation/XXIII Congress of the Chilean Association of Automatic Control (ICA-ACCA), 2018, pp. 1-5
- 2 S. Mahajan, U. Shah, R. Tambe, M. Agrawal and B. Garware, "Towards Evaluating Performance of Domain Specific Transfer Learning for Pneumonia Detection from X-Ray Images," 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), 2019, pp. 1-6
- 3 M. Aledhari, S. Joji, M. Hefeida and F. Saeed, "Optimized CNN-based Diagnosis System to Detect the Pneumonia from Chest Radiographs," 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2019, pp. 2405-2412
- 4 D. Varshni, K. Thakral, L. Agarwal, R. Nijhawan, and A. Mittal, "Pneumonia Detection Using CNN-based Feature Extraction," 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2019, pp. 1-7
- 5 A. Tilve, S. Nayak, S. Vernekar, D. Turi, P. R. Shetgaonkar and S. Aswale, "Pneumonia Detection Using Deep Learning Approaches," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020, pp. 1-8
- 6 A. Pant, A. Jain, K. C. Nayak, D. Gandhi and B. G. Prasad, "Pneumonia Detection: An Efficient Approach Using Deep Learning," 2020 11th

- International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1-6
- 7 S. Singh, "Pneumonia Detection using Deep Learning," 2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), 2021, pp. 1-6
- 8 Datasets: <https://www.kaggle.com/datasets/prashant268/chest-xray-covid19-pneumonia>