Faculty of Computers and Artificial intelligence
Department of Computing and Bioinformatics

Predicting pneumonia disease using CNN

Supervised by

Dr. Sabah Sayed
TA. Menna Youssef

Implemented by

Diaa Ahmed Refat
Mostafa Mohamed Fayez
Abd-elrahman Mohsen

Graduation Project
Academic Year 2021-2022
Documentation

Table of content

List of Figures

List of abbreviation

CNN: convolutional neural network

VGG: Visual Geometry Group

Resnet: Residual Network

Enet: Efficient Neural Network

LIME: Local Interpretable Model-Agnostic Explanations

ADMAX: Adaptive Moment Estimation (Adam) algorithm

SGD: stochastic gradient descent

ReLU: Rectified Linear Unit

API: Application Programming Interface

Val: validation

List of tables:

Chapter 1 Introduction

1.1 Motivation:

The motivation for using CNNs (Convolutional Neural Networks) in pneumonia detection is primarily due to their strong performance in various image classification problems, including the detection and classification of lung diseases from chest X-ray images. Since pneumonia is a lung disease that affects the air sacs in one or both lungs, its detection using chest X-ray images can be challenging for traditional image processing techniques. CNNs, on the other hand, have shown a high level of accuracy in identifying features in medical images, making them a promising tool for pneumonia detection. Additionally, the scarcity of available data for training traditional machine learning models can be addressed by using transfer learning, a type of deep learning that involves transferring knowledge learned from one task to another. This makes the use of CNNs in pneumonia detection even more appealing.

- Diagnosing pneumonia using traditional way:
  - Blood tests.
  - Chest X-ray.
  - Pulse oximetry.
  - Sputum test.
  - Your doctor might order additional tests if you're older than age 65, are in the hospital, or have serious symptoms or health conditions. These may include:
    - CT scan.
    - Pleural fluid culture.

The outbreak of the COVID-19 pandemic has increased the urgency to develop efficient and accurate methods for pneumonia detection, as COVID-19 can cause pneumonia-like symptoms and can have severe respiratory complications.

1.2 problem definition:

The problem definition of pneumonia detection using CNNs is to accurately identify the presence of pneumonia in chest X-ray images using deep learning techniques.
Pneumonia is a common and serious respiratory infection that affects the lungs and can potentially be fatal if left untreated.
Deep learning methods, particularly convolutional neural networks (CNNs), have been shown to be highly accurate in identifying features in medical images, including chest X-ray images.
The goal of pneumonia detection using CNNs is to develop an efficient and accurate method for early detection and diagnosis of pneumonia, which can ultimately improve patient outcomes and reduce the burden of the disease on healthcare systems.
Additionally, the accuracy and efficiency of CNNs in pneumonia detection can be further improved by using transfer learning, a type of deep learning that involves transferring knowledge learned from one task to another.
Overall, the problem definition of pneumonia detection using CNNs is to develop an accurate and efficient method for early diagnosis and management of pneumonia, which can ultimately improve patient outcomes and reduce the burden of the disease on healthcare systems.

1.3 project objective:

the project objective is to make a tool for everyone not only healthcare organizations, this tool will help to have a valid and reliable detection for the chest x-ray if you are infected with pneumonia or not, and also give you a explanation if you have pneumonia how our model detected it (the features that lead to this classification), so we need to have a good model using CNN by comparing different models architecture and choose the best of them and deploy it on website and also to have a model that have high accuracy to be reliable because we need number around 97 and 99 to say that our model good for this problem because this problem can save a lot of human live so we need a high accuracy as we can and this is our project objective.

1.4 Gannt chart:

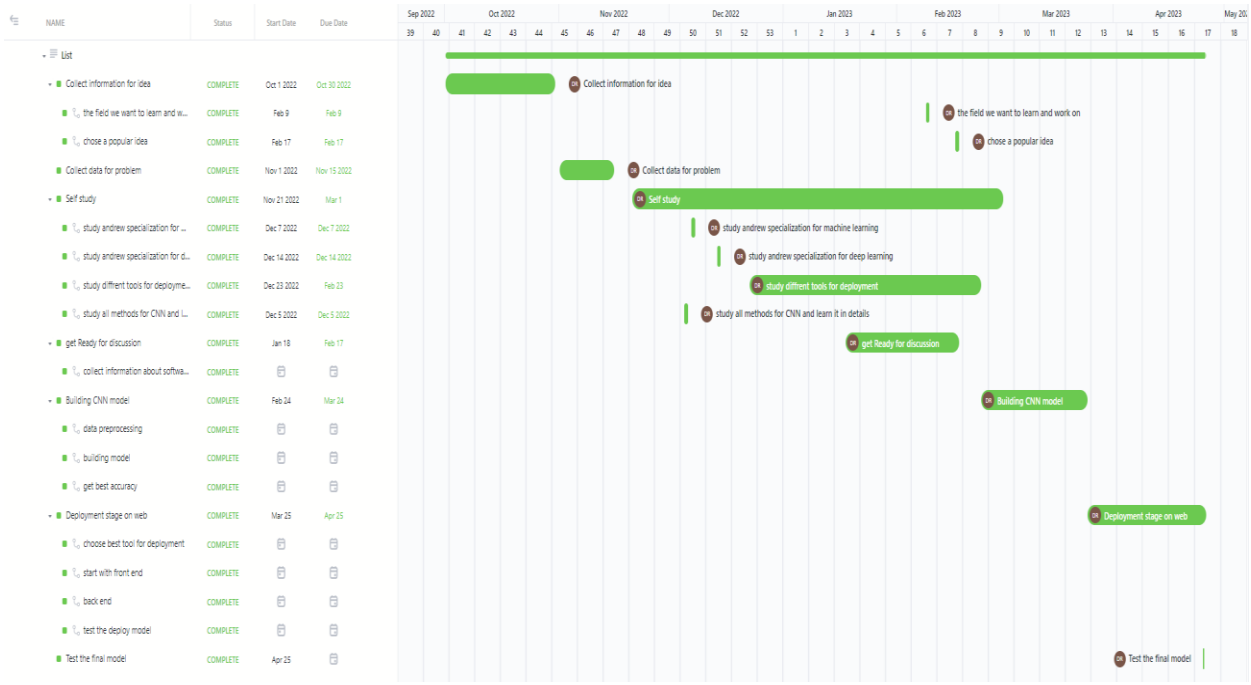


Fig 1.1(Gantt Chart)

1.5 Project development methodology:

First, we need to take a look at data and fix any issues and then we will need to work on this data to build a different model with different architecture to choose the best of them and deploy it on a website.

1. **Prepare the data:**
   1- When we take a first look at the data it shows that it's organized into 3 folders (train, test, Val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal).
   2- So, we will load the data and see what distribution of it is, when we did this, we get this figure below.

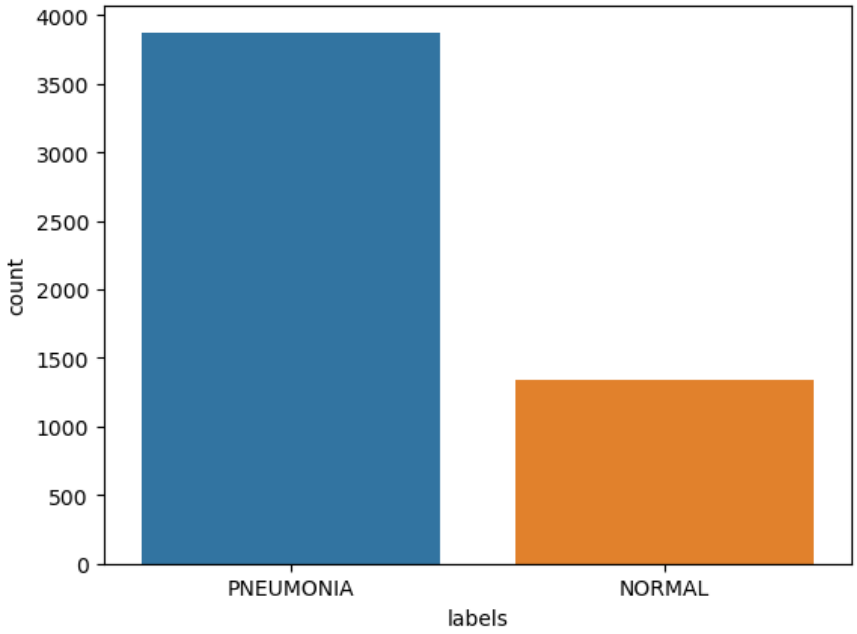


Fig 1.2 (histogram for the data before resampling)

   3- As we can see there is a huge difference between two classes the normal and pneumonia, so we need a method to fix this problem first.
   4- We will make the pneumonia images the same as the normal images by decreasing the number of photos of pneumonia images.

5- As we can see from this figure, we successfully did it and we have a balanced data set to work on.

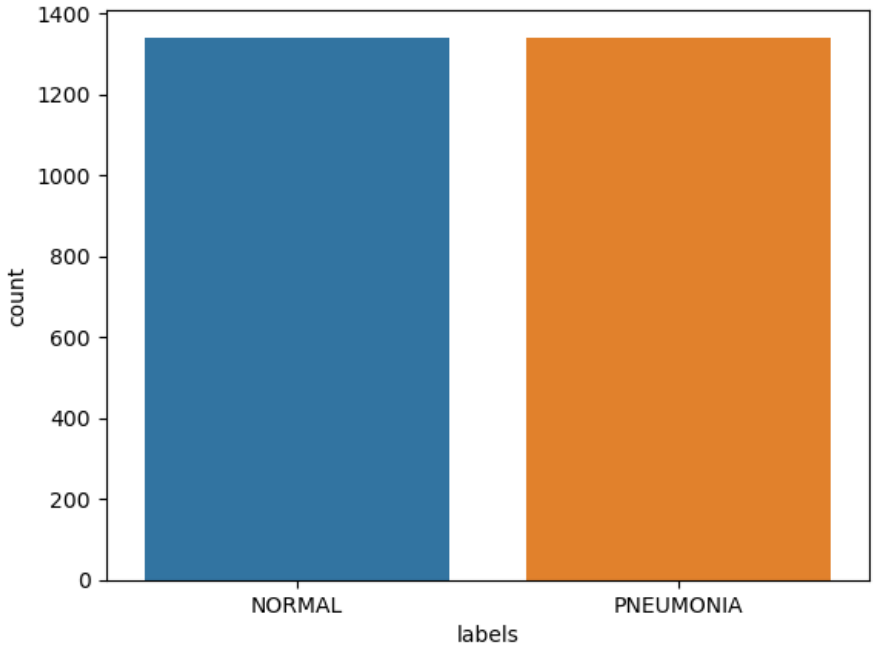


Fig 1.3 (histogram for the data after resampling)

6- Now we need to start to work on the model but before we train the model, we need to make sure that all the labels are true, when we read some comments on this dataset, we found that (test and validation file) include only 600 for test and 16 for validation and this number is too small for this task.

7- And also, in the test file there are a lot of wrong labels we read it from comments and also found when we train including these files, we get lower accuracy than we train with only train file.

8- The train file contains more than 5 thousand images and this very good for our task, so we don't need to include test and valid files in our data, and this gives us very good accuracy on every model we work on.

9- Then we need to make data augmentation which is a technique used in machine learning and computer vision to artificially increase the size of the training dataset by generating new training samples from the existing ones. The goal is to reduce overfitting and improve the robustness and generalization performance of the models.

10- We found that we don't need a lot of augmentation because when we use a lot of augmentation and specially using scaling the validation accuracy stuck and this means we get stuck on local minima.
11- so, we only used rotation_range=5 and width_shift_range=0.1 and height_shift_range=0.1 on this task to just make our model more generalize because if we don't do this, we will get error classify if the image slightly different

2. Models' architecture:

Vgg16

1- a convolutional neural network (CNN) model that supports 16 layers. It was developed by K. Simonyan and A. Zisserman from Oxford University, and it was one of the best performing architectures in the 2014 ImageNet Large Scale Visual Recognition Challenge.
2- The model consists of 13 convolutional layers and three fully connected layers, and it has a large number of weight parameters, close to 550MB in size, leading to the model being relatively large in size

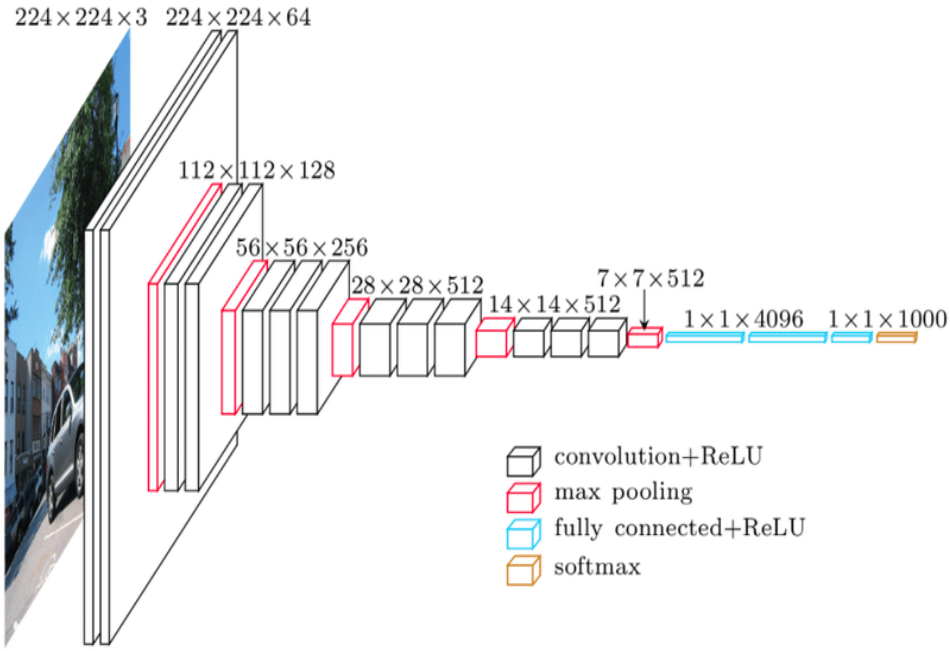


Fig 1.4 (VGG16 architecture)

Resnet50

1- ResNet50 is a convolutional neural network model that has 50 layers, developed by Microsoft for image recognition and computer vision tasks.

2- It uses residual connections to mitigate the vanishing gradient problem that can occur in deep neural networks.

3- The model has 48 convolutional layers, one max pool layer, and one average pool layer.

4- Each convolutional block in the model has three convolutional layers with a skip connection that allows the model to learn more complicated mappings with residual connections.

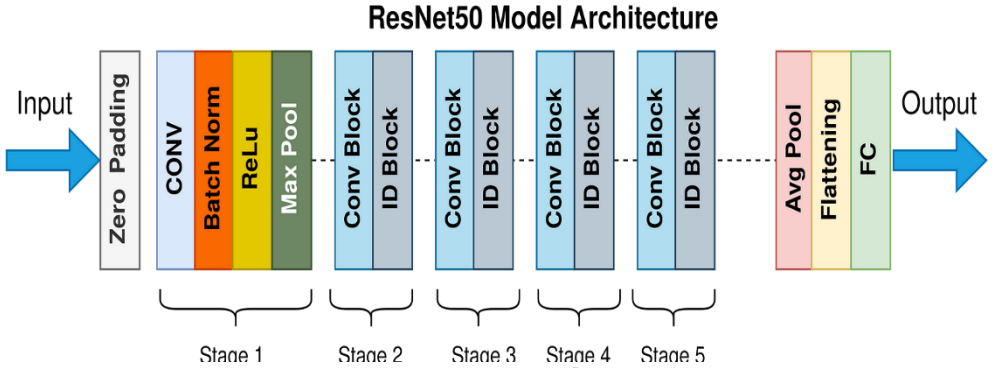


Fig 1.5 (Resnet 50 architecture)

LIME

1- Local Interpretable Model-agnostic Explanations (LIME) is a technique for explaining the predictions of black box machine learning models in a way that is easy to understand and interpret.

2- LIME works by approximating the black box model with a local, interpretable surrogate model.

3- This surrogate model is then used to generate explanations for individual predictions made by the black box model.

4- LIME is considered model-agnostic, since it can be used to explain any model without requiring knowledge of its internal workings. The technique has been widely used in applications such as image classification.

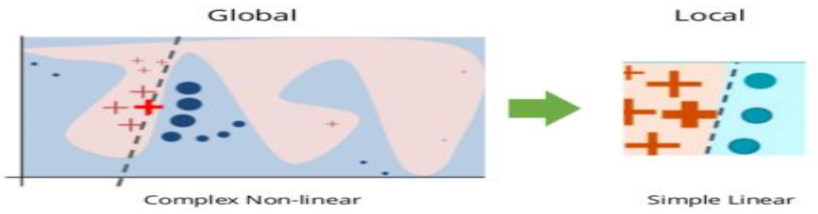


Fig 1.6 (LIME explanation)

Enet (proposed model)

1- Efficient Net is a family of deep neural networks designed for efficient and accurate image classification tasks.
2- It was introduced by Mingxing Tan and Quoc V. Le in their 2019 paper, "Efficient Net: Rethinking Model Scaling for Convolutional Neural Networks".
3- Efficient Net uses a novel compound scaling method that uniformly scales the depth, width, and resolution of the network. This allows it to achieve state-of-the-art accuracy with significantly fewer parameters and computations than other popular models such as ResNet and Inception. In other words, it can achieve higher accuracy with less computational resources.
4- After we tried different versions of this model, we found that enet b4 is the most effective version for our model and gives us more generalized and accuracy on the test data around 98.81.

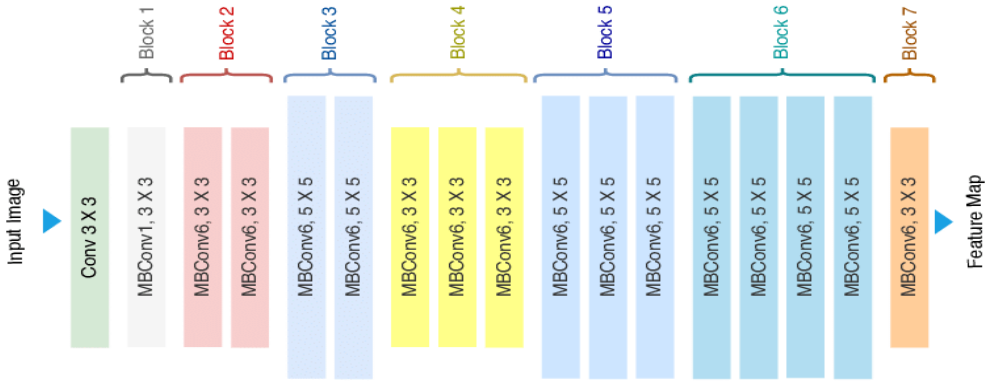


Fig 1.7 (e net architecture)

3. Foundations

Dropout

1- Dropout layers are a common regularization technique used in Convolutional Neural Networks (CNNs). In CNNs, dropout is typically applied to the fully connected layers at the end of the network, by randomly dropping out some of the neurons during training.
2- This helps to prevent overfitting and improve the generalization performance of the model. Dropout can also be applied to the convolutional layers, but care must be taken to avoid too much loss of information.
3- So, in our training phase we found that the best dropout is 0.5 and this helps our model to learn faster and don't get an overfitting problem.

Stop early on plateau.

1- technique used in machine learning to determine the optimal time to stop training a model.
2- It helps to prevent overfitting and reduce training time by monitoring the validation loss during training and stopping the training process when the validation loss stops improving or plateaus.
3- We use patience of 5 If the validation loss does not improve after the 5 epochs training is stopped and the model with the best validation loss is saved.

Checkpoint

1- A checkpoint callback is a technique used in training deep learning models to save the weights of the model during the training process. This allows training to be stopped and resumed at a later time without losing the progress or the state of the model.
2- It can be customized to save the weights of the model at different intervals, based on the frequency or the improvement of the metric being monitored.
3- This helps to save disk space and avoid overwriting previously saved checkpoints unnecessarily.

Early stopping

1- The Early Stopping callback is a technique used in machine learning to stop the training of a model at an early stage if the performance on the validation set does not improve for a specified number of epochs.
2- This technique helps to prevent overfitting and improve the generalization performance of the model.
3- We used in our case the parameter patience to be equal 5 epochs.

Adamax

1- AdaMax is an optimization algorithm, which is a variant of the Adam optimization algorithm that is designed to converge faster than the classical stochastic gradient descent (SGD) method. It was introduced by Diederik P. Kingma and Jimmy Ba in their paper, "Adam: A Method for Stochastic Optimization" in 2015.
2- AdaMax is particularly useful for deep learning applications since it provides faster convergence and improved generalization performance compared to other optimization algorithms.
3- It uses the same exponential moving average of gradient values as used in the Adam algorithm but replaces the second moment of gradients with the L-infinity norm of the gradients.
4- In AdaMax, the learning rate adapts to the gradient accumulation over iterations and takes an average of infinity norms over an infinite set of values. The algorithm is computationally efficient and is well-suited for large scale and high-dimensional datasets.

Loss function

1- Categorical cross entropy is a loss function used in training neural networks for classification tasks where the output variable is categorical.
2- It is commonly used in multi-class classification problems and calculates the cross-entropy loss between the predicted output and the true output.
3- The goal is to minimize this loss function during training by adjusting the weights of the neural network to achieve better predictions on the input data.

4. Results

1- This our evaluation accuracy from three architecture.

Arch name	VGG16	Resnet50	ENETB4
Accuracy	0.9604	0.9688	0.9921
F1 SCORE	0.95	0.97	0.98
Acuuracy on test	0.948	0.967	0.981

table 1.1(compare accuracy between the three models)

from this table it shows that ENET is more efficient than vgg16 and resnet50

2- And this is evaluation graphs for the best model enetb4 show (AUC score and validation loss) vs training.

ENETB4

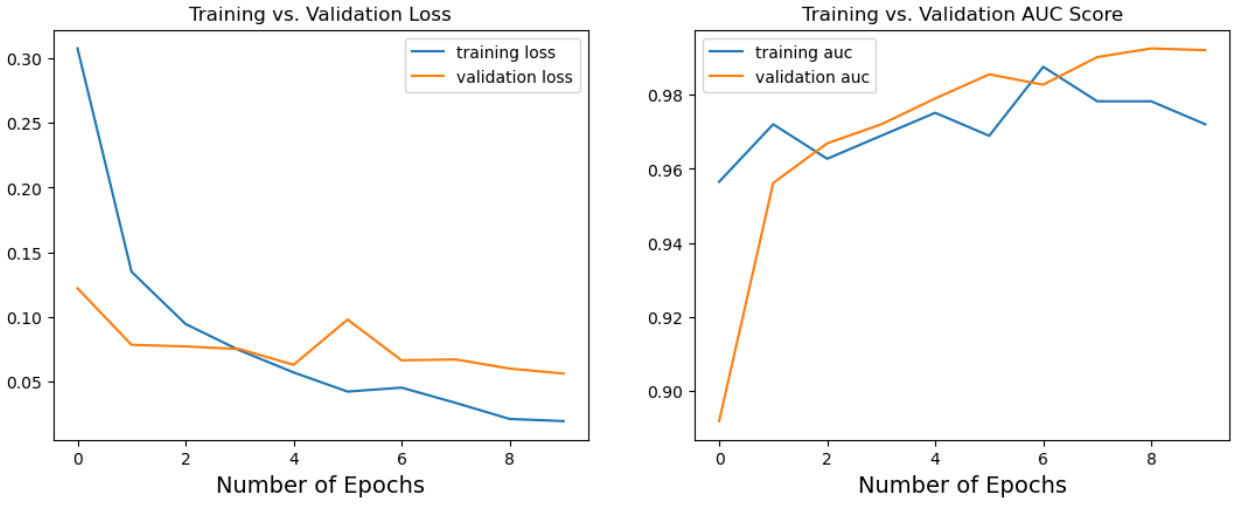


Fig 1.8(Enet evaluation graph)

3- And this is classification report.

ENETB4

	precision	recall	f1-score	support
NORMAL	0.99	0.97	0.98	105
PNEUMONIA	0.97	0.99	0.98	110
accuracy			0.98	215
macro avg	0.98	0.98	0.98	215
weighted avg	0.98	0.98	0.98	215

Fig 1.9(Enet classification report)

4- Finally, confusion matrix for best model enetb4

ENETB4

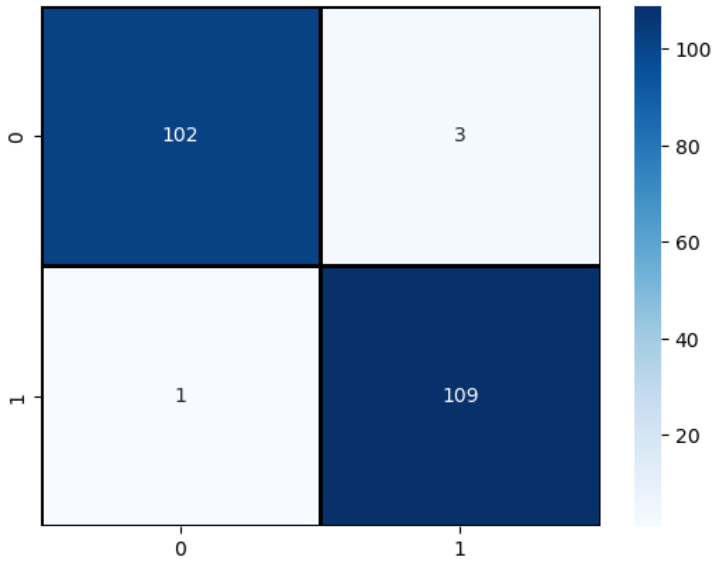


Fig 1.10(Enet confusion matrix)

1.6 The used tools in the project (SW and HW)

1- Hardware

This project does not need any hardware except one's personal computer. –on web site

2- Software

1. Models

I. Languages:

Python	Python is a programming language that lets you workquickly and integrate systems more effectively.

II. Libraries:

SKLearn	Which is probably the most useful library for machine learning in Python with efficient tools formachine learning and statistical modeling including classification, regression, clustering and dimensionality reduction-which needed for splitting to test and train set and evaluate confusion matrix and classification report and also resample tool.
TensorFlow	Is an end-to-end platform that makes it offers multiple levels of abstraction so you can choosethe right one for your needs. Build and train models by using the high-level Keras API which easy for us to build and deploy ML models.
Keras	Is an API designed for human beings which followsbest practices for reducing cognitive load and offers consistent & simple APIs,

skimage	scikit-image or skimage is a Python library designed for image processing and computer vision tasks. It provides a collection of algorithms for image processing, including filtering, segmentation, feature extraction, and transformations. The library is built on top of other popular scientific computing libraries in Python, such as NumPy, SciPy, and Matplotlib.
LIME	LIME (Local Interpretable Model-Agnostic Explanations) is a Python library used for explaining the predictions of any machine learning model. It was introduced by Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin in their 2016 paper, "Why Should I Trust You?": Explaining the Predictions of Any Classifier. The library is built on top of popular machine learning frameworks in Python, such as scikit-learn and TensorFlow.

III. IDES:

Kaggle	Kaggle is a platform for data scientists and machine learning enthusiasts to find and participate in data science competitions, collaborate on projects, and explore open datasets. It was founded in 2010 and has since grown to become one of the largest and most active communities of data scientists and machine learning practitioners in the world.
HTML	Is an open-source UI software development kit, itis used to develop cross platform applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase.
CSS	CSS stands for Cascading Style Sheets. CSS describeshow HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout ofmultiple web pages all at once
Flask	Flask is a micro web framework written in Python. It is classified as a microframework because it does notrequire particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

Visual studio code	Visual Studio Code is a powerful and lightweight source code editor that can be used on Windows, macOS, and Linux. It is developed by Microsoft and is available for free. With its support for hundreds of programming languages, it offers countless features and extensions that help developers boost their productivity

table 1.2(Web App tools and libraries)

1.7 Report Organization (summary of the rest of the report)

1- Chapter Two - Related Work:

in this chapter, we establish studies and approaches related to our research. We will present various methods to achieve our objectives, including the authors of these approaches, their classification methods, and their achieved accuracies.

2- Chapter three - System Analysis:

This chapter focuses on the project specifications and usability, reliability, and stability performance details from a user's perspective. Additionally, we will illustrate a use case diagram that highlights the functionality of our program.

3- Chapter Four - System Design:

The System Design Document outlines the system and subsystem architecture, operating environment and requirements, database and file design, output layouts, and the interaction between human-machine interfaces. It also includes detailed designs, processing logic, and external interfaces using various diagrams such as System Component, System Class, and Sequence Diagrams, Project ERD, and System GUI Design.

4- Chapter Two - Implementation and Testing:

This phase includes screen shots of training and testing of (VGG16, RESNET50, ENETB4)
Screenshots of the system running and samples of the applied test cases.

Chapter 2 Related work

1. **First paper authors V. Sirish Kaushik, Anand Nayyar, Gaurav Kataria and Rachna Jain**

❖ They used four models to compare and take the best of them and this is the structure of the process.

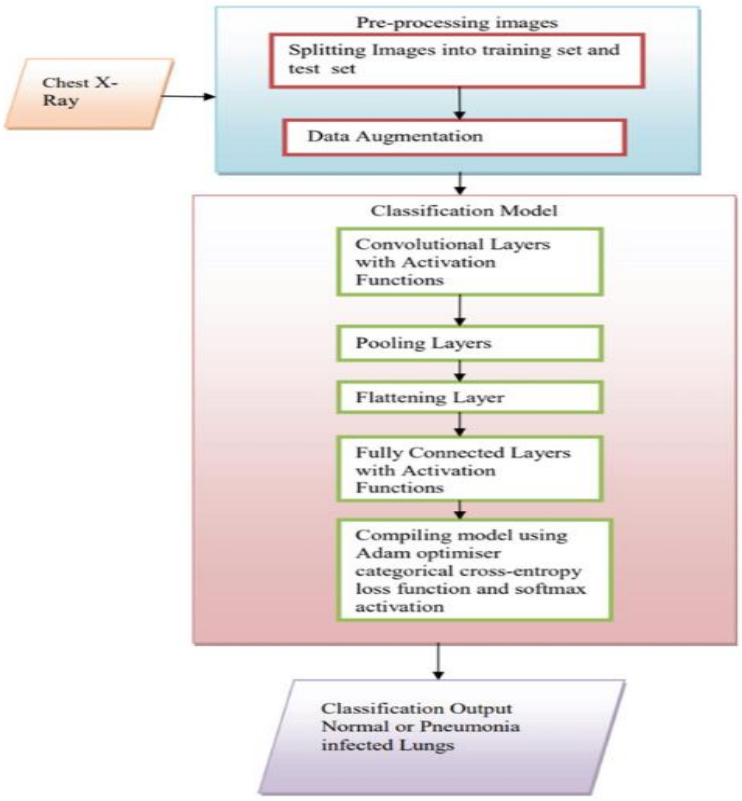


Fig 2.1(structure process paper 1)

❖ Now let's compare their results on the different three models and our model.

Arch name	model 1	model 2	model 3	model 4	ENETB 4
Valid loss	27.31	38.36	25.23	26.61	5.63
F1	92	89	94	94	98

table 2.1(comparing models of paper one and our model)

- It shows that their models didn’t achieve our success for different reasons:

    1- First, they didn’t use transfer models, but they used a model that they built in their task, but we used transfer learning because our target is to get the highest accuracy.
    2- And they used all the data set on their task, but they didn’t take comments that say that the data on the test file have a lot of wrong labels and it makes the training have low accuracy.
    3- Also, they seem used the data with the problem of imbalance and this is big problem they will get high recall but low precision and that what happened on their models what makes f1 score is lower than ours

- This is a small part of their conclusion.

    - The validation accuracy, recall and F1 score of CNN classifier model 3 with three convolutional layers are 92.31%, 98% and 94%, respectively, which are quite high compared to other models that were trained. CNN classifier model 4 with four convolutional layers also comes very close in performance with 91.67% validation accuracy, 98% recall and 94% F1 score. Both models have the same recall and F1 scores.

2. Second paper authors Patrik Szepesi, László Szilágyi (published on ScienceDirect)

❖ First let's talk about their model from their paper they said:

"Our new model slightly resembles the VGG-16 architecture in that some of its early layers follow a similar structure of convolutional layer, batch normalization layer, activation layer, which is finally followed by a pooling layer.
However, our network could not be more different from the VGG's architecture which performed poorly on the given dataset. As previously described, the unique integration of the dropout is located in a convolutional layer.
As depicted the model follows the pattern of convolutional layer followed by pooling layer.
This pattern is repeated several times, because it yet again proves the power of the convolutional layers and its ability to learn specific features, which are often referred to as feature maps when many of them are stacked together.
Filters are essentially the elements that enable the neural network to pick up on specific patterns such as edges.
Filters are often initialized in many ways but ultimately that can be omitted, because the purpose of the convolutional layer is to learn the specific features during the training; they are basically the mechanism needed for feature extraction.
At last, the matrices are flattened and are then connected to the dense fully connected layers, which are responsible for the classification.
It is worth noting that within each convolutional block in the model's architecture there is a batch normalization layer followed by a ReLU activation function."

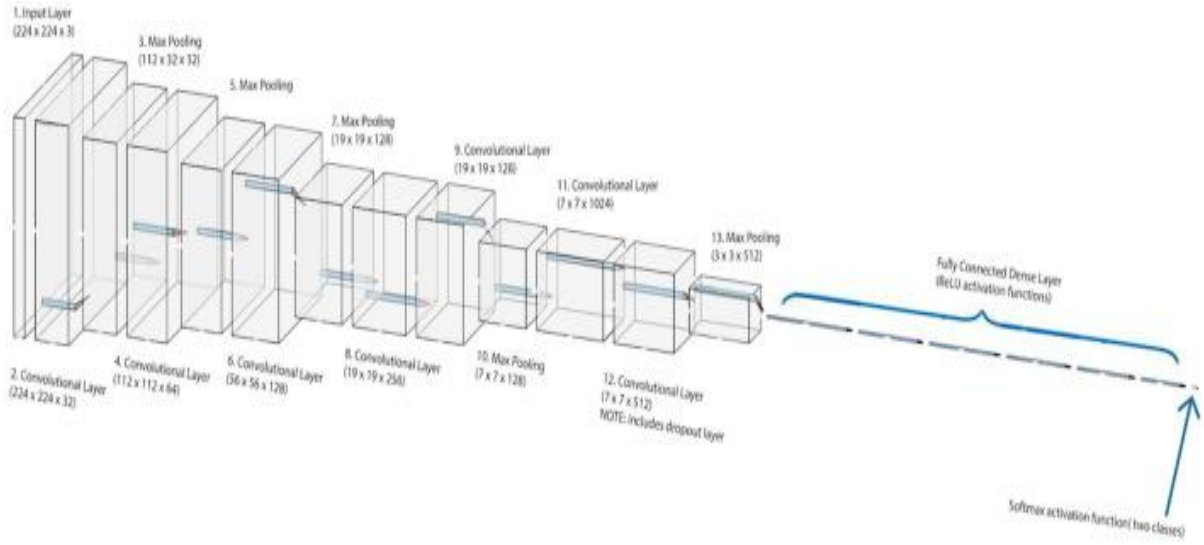


Fig 2.2(model architecture paper 2)

Let' compare their model and our model:

Arch name	paper-model	ENETB4
accuracy(%)	97.02	98.1
Recall (%)	97.34	99
Precision (%)	97.40	97
F1 score (%)	97.37	98

table 2.1(comparing models of paper two and our model)

❖ Let's talk about some problems in this paper:

1- From their source code in GitHub, they give us easy way to know what their problems are and how to fix them.
2- We saw that they used all the data in their process and when we did the same thing we have a accuracy same as them but when we take

a look on comments on the dataset we found that the data in test and Val files have a lot of wrong labels which it makes the prediction accuracy very low

3- And they also used the data without balance the data which is mean they will work on imbalanced data that makes the model prediction accuracy stuck on 97%
4- They also used 50 epochs for this task, and this is making the computational cost very high which it takes like 2 hours to train it on Kaggle GPU
5- The accuracy is good, it's 97 % but we want to have the highest score that we could achieve.
6- They say in their paper that they compared with different pre-trained models like vgg16 and etc. and this is their results from their paper.

"Mainly based on convolutional neural networks (CNN) and deep learning. Jain et al. [2] deployed VGG-16, VGG-19, Resnet50, and Inception-V3 network architectures and transfer learning for the detection of viral pneumonia, obtaining 71–88% accuracy on 624 test images. Dey et al. [3] combined the VGG-19 architecture with various classifiers like SVM or random forest, and based on a set of 1650 X-ray images they achieved accuracy, precision, and score values ranging up to 98%, 95%, and 96%, respectively. Brunese et al. [4] established a three-class pneumonia detection framework based on VGG-16 architecture and the so-called GradCAM algorithm for visual debugging and achieved a 96.2% accuracy on a set of 6523 CXR images. Panwar et al. [5] combined the VGG-19 architecture with the GradCAM algorithm and achieved 95.6% accuracy in a three-class pneumonia classification framework. Similarly, Ibrahim et al. [6] proposed a solution for three-class pneumonia detection, using VGG and ResNet152V2 architectures, and obtained recall values of 93–97% based on an own collection of X-ray images. Jin et al. [7] proposed a three-stage hybrid model consisting of feature extractor, feature selector, and SVM classifier stage, which achieved a remarkable 98.62% accuracy on a private set of 1743 X-ray images. Karthik et al. [8] employed the so-called Channel-Shuffled Dual-Branched (CSDB) CNN architecture to distinguish various types of pneumonia from several publicly available datasets, achieving scores of 94–98%. Quan et al. [9] combined the DenseNet and the CapsNet architecture and achieved 96% recall on a small sized COVID-19 dataset, but only 90.7% accuracy on larger set of pneumonia X-

ray images. Alhudhaif et al. [10] deployed a DenseNet-201 based architecture and achieved accuracy and recall values up to 95% and almost 90% precision on Kaggle's dataset of over 6000 X-ray images. Sirazitdinov et al. [11] used the 26,684 CXR images of the Kaggle Pneumonia Detection Challenge to train an ensemble of two CNNs, namely a RetinaNet and a Mask R-CNN, but they achieved only a recall value of 80%. The Mask R-CNN architecture was also deployed by Jaiswal et al. [12] to localize regions of the lung affected by pneumonia. Mahmud et al. [13] proposed a so-called CovXNet architecture, a CNN that uses depthwise convolution with varying dilation rates to extract a wide range of features from X-ray images, which was trained and validated on 6161 CXR scans, achieving accuracy of 90.2% in a four-class and 97.4% in a two-class pneumonia classification problem. Ouchicha et al. [14] proposed a residual CNN architecture and trained it on 2905 X-ray images, reaching a 96.7% overall accuracy in three-class pneumonia classification. Wang et al. [15] designed a so-called prior-attention residual learning block and combined it with two 3D-ResNets and obtained 93.3% accuracy in the three-class pneumonia detection problem, based on 4697 X-ray images. Probably the largest available CXR image dataset, the one published by RSNA Pneumonia Detection Challenge consisting of over 120,000 items, was used by Wang et al. [16] to train a VGG architecture. They obtained an accuracy of 94.62% in the detection of pneumonia."

- Let's talk about how we make the solve all these problems:

1- From the above we found that they didn't used efficient net for this kind of problem, and we know that is very effective for this kind of problems and we tested it we found a very good accuracy which is better than their model with 98 % accuracy on the test set.
2- On the dataset we did some of the fixes, first we didn't take data from test and Val folders so we can avoid problem of wrong labels second, we make the data to be balanced by using resample method from sckilit learn library.
3- And we also used some augmentation on these images by rotation and other methods that makes our model more generalized, but they didn't do it.

Chapter 3 System Analysis

3.1 Project specification:

3.1.1 Functional requirement:

1. **Image Classification:** The system should be able to classify chest X-ray images as either infected (pneumonia) or normal using a CNN model, specifically an efficient net.

2. **User information:** The web application should allow users to enter their personal information, including name, phone number, email address, age, gender, and marital status.

3. **Symptom Selection:** Users should be able to select the presence and severity of various symptoms associated with pneumonia, such as cough, fever, shortness of breath, chest pain, chills, headache, muscle aches, nausea or vomiting, and confusion.

4. **Medical History and Comments:** The system should provide an input field for users to enter their medical history and any additional comments.

5. **Data Storage:** The system should store user information, symptom details, and uploaded chest X-ray images in a database.

6. **Image Upload:** Users should be able to upload a chest X-ray image for examination.

7. **Examination Result:** After image analysis, the system should provide the user with the examination result, indicating whether the X-ray suggests an infection or is normal.

3.1.2 Non-Functional requirement:

1. **Accuracy:** The CNN model should have a high accuracy rate, preferably around 99%, in classifying chest X-ray images.

2. **Performance:** The system should process user input and image classification quickly to provide a timely examination result.

3. **User Interface:** The web application's front-end should have an intuitive and user-friendly interface using HTML and CSS.

4. **Security:** User data, including personal information and medical history, should be securely stored and protected.

5. **Scalability:** The system should be able to handle an increasing number of user registrations, image uploads, and database entries without significant performance degradation.

6. **Reliability:** The system should be reliable, ensuring that user data is accurately stored and retrieved from the database.

7. **Compatibility:** The web application should be compatible with various web browsers and devices to ensure accessibility for users.

3.2 Use case diagrams:

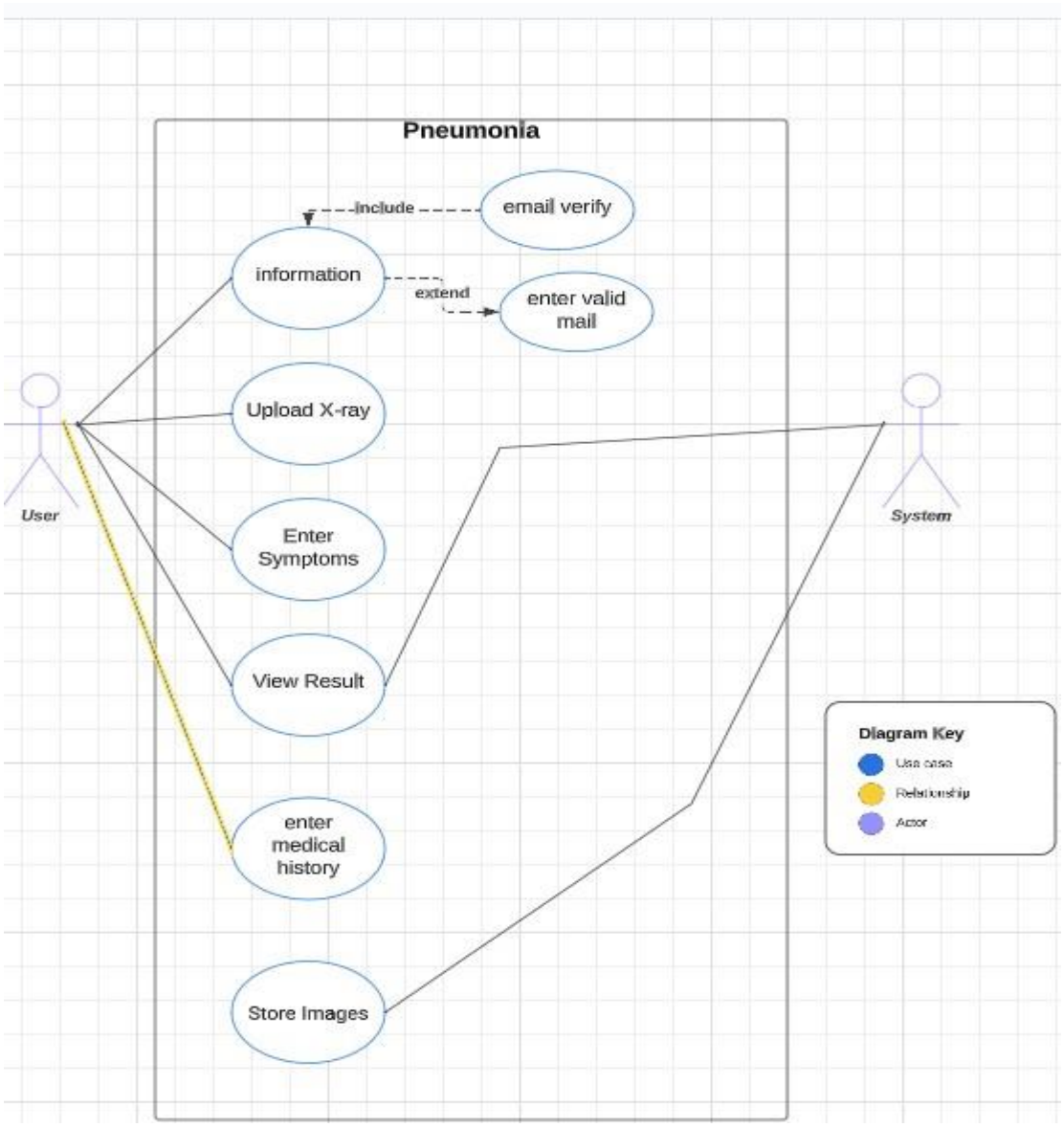


Fig 3.1(use case diagram)

Chapter 4 System Design

1. System Component Diagram

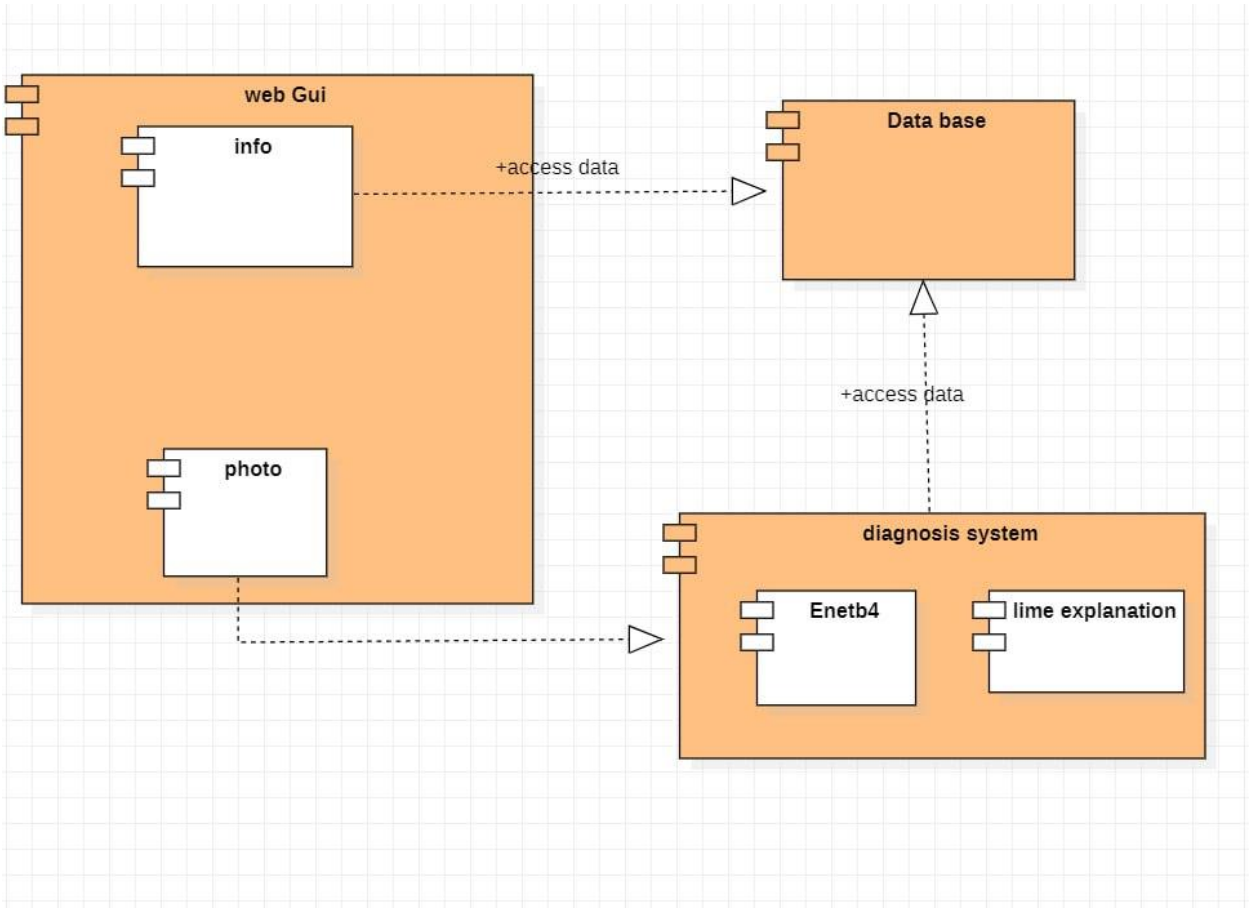


Fig 4.1(system component diagram)

2. System Class Diagrams

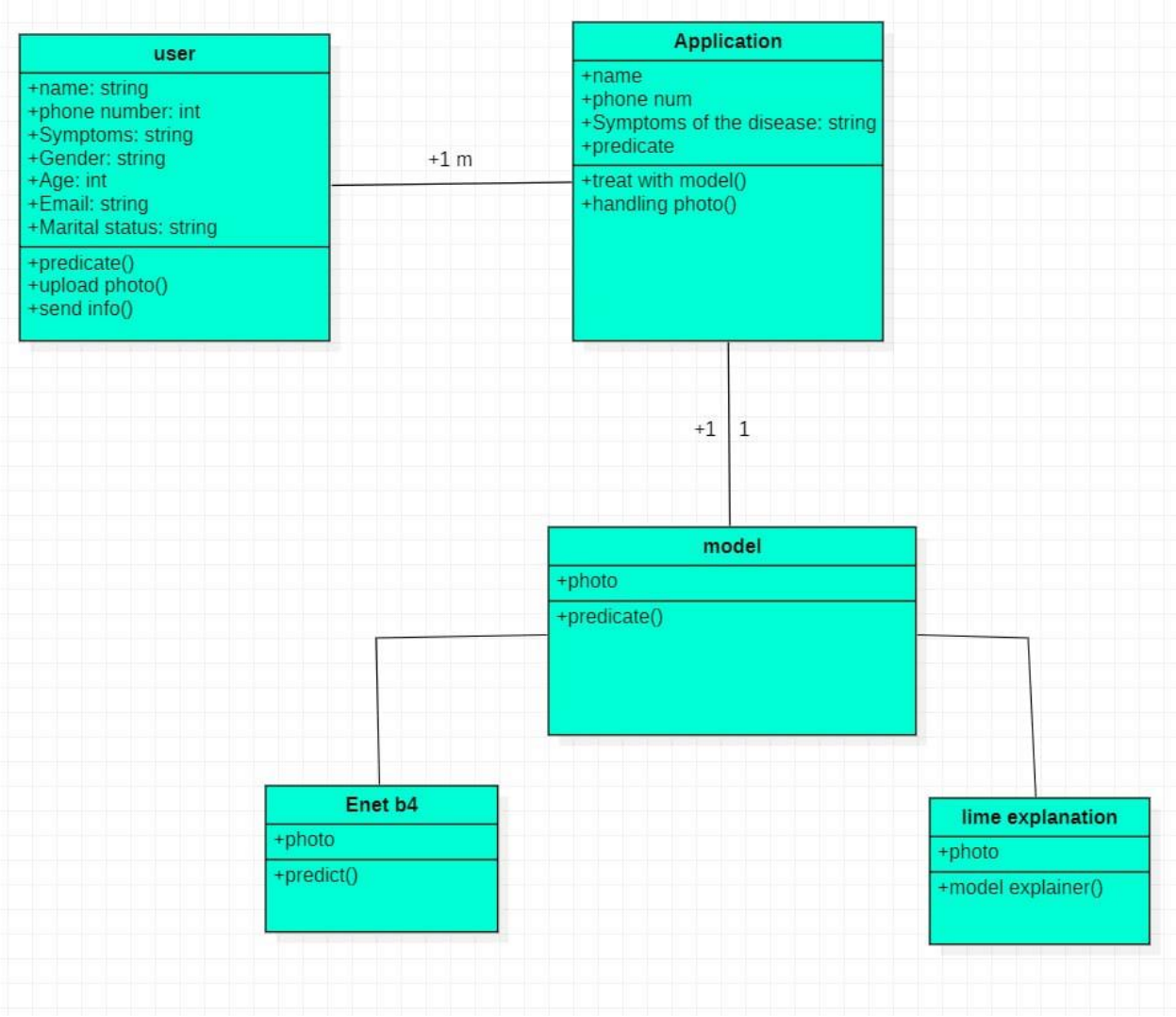


Fig 4.2(system class diagram)

3. Sequence Diagram

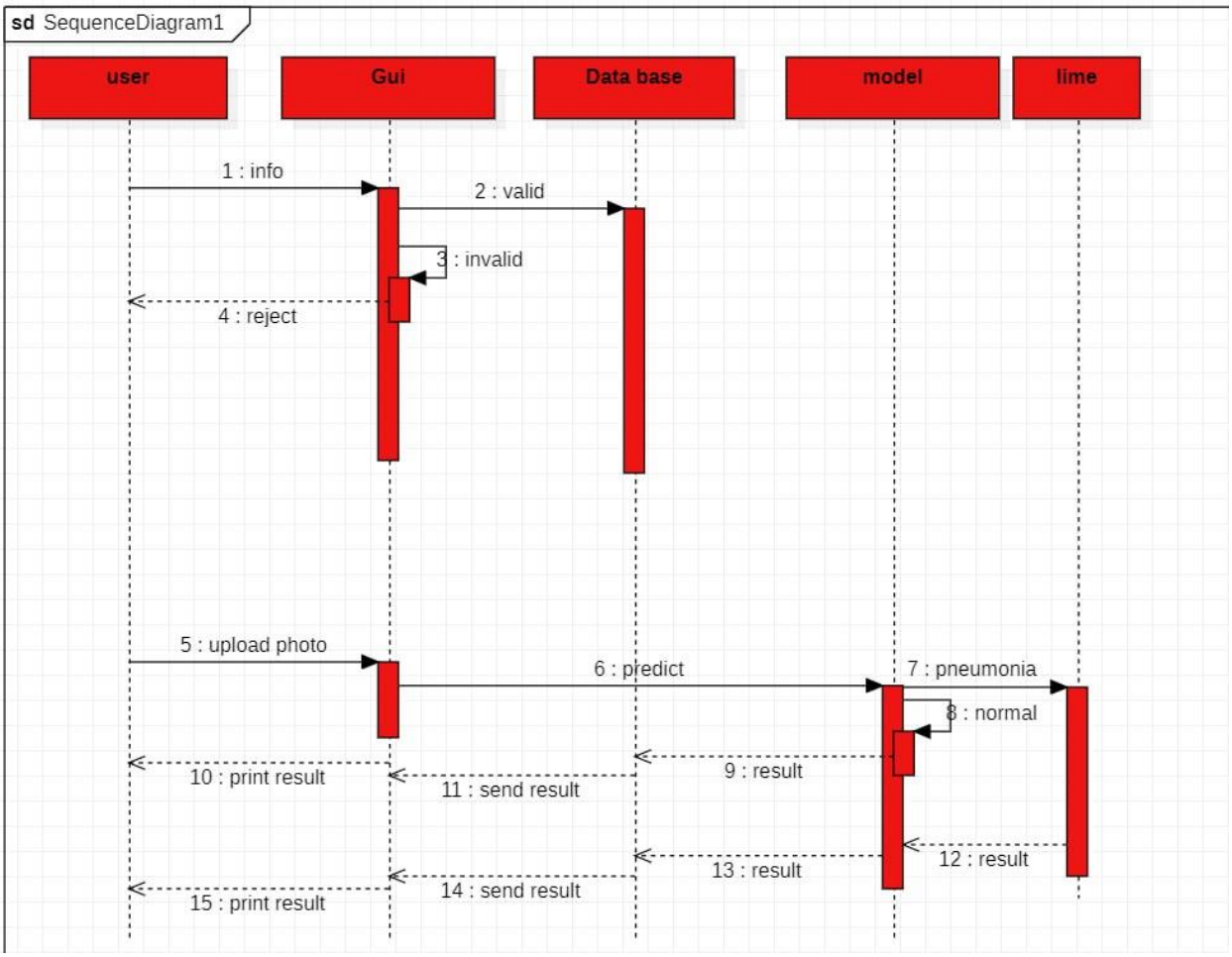


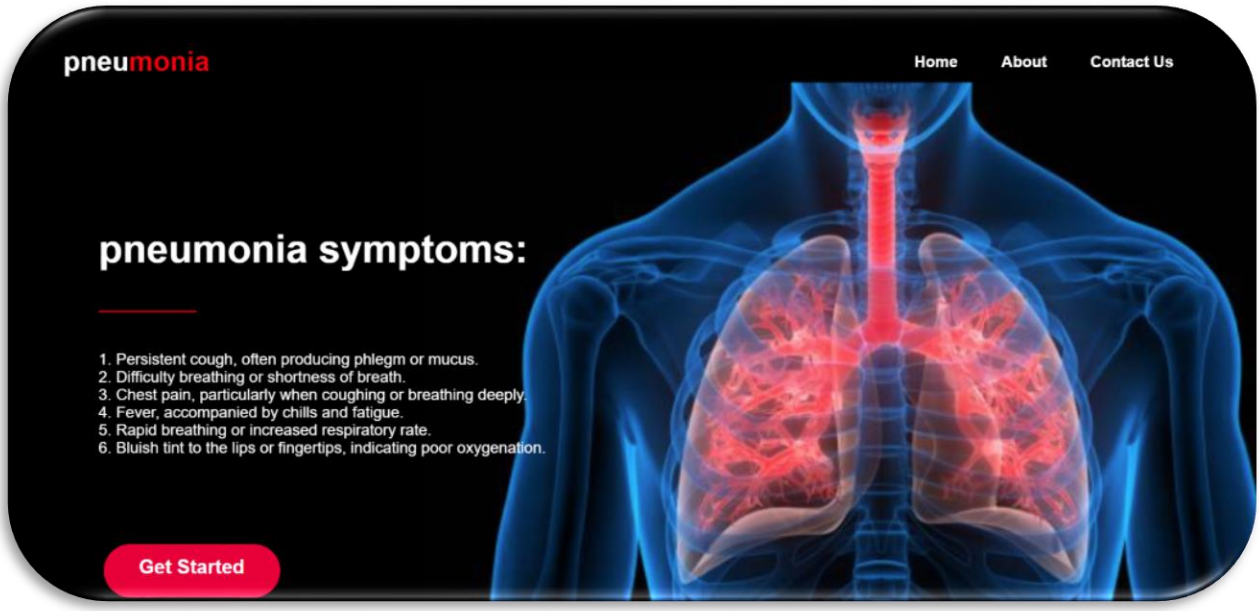
Fig 4.3(sequence diagram)

4. ERD

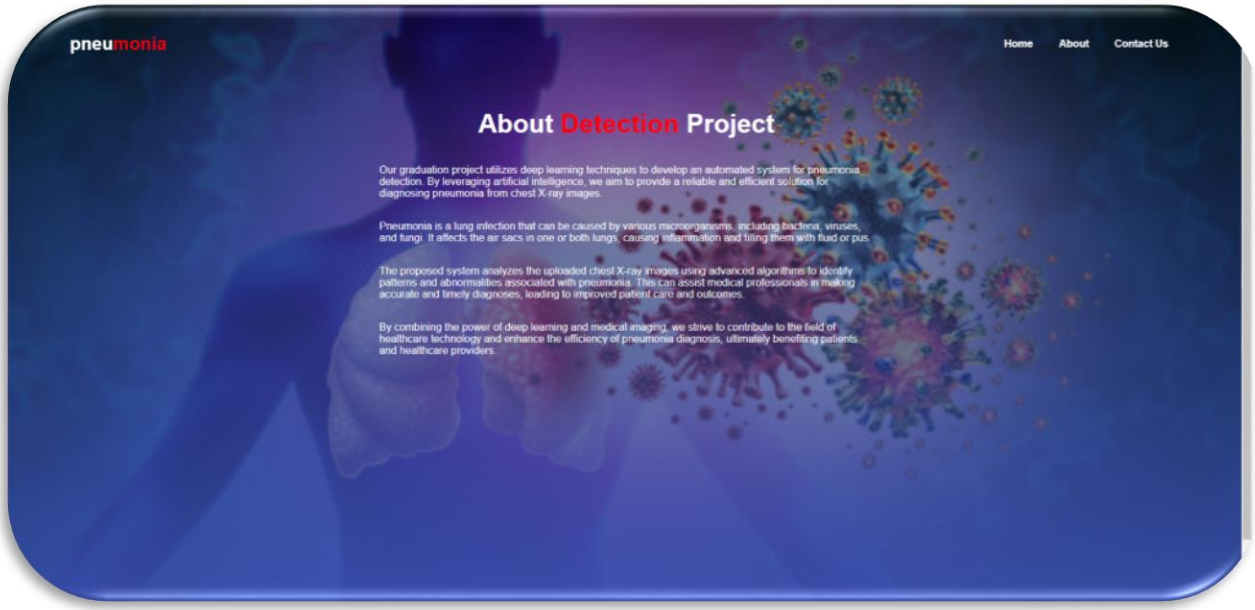
Fig 4.4(ERD)

5. System GUI Design

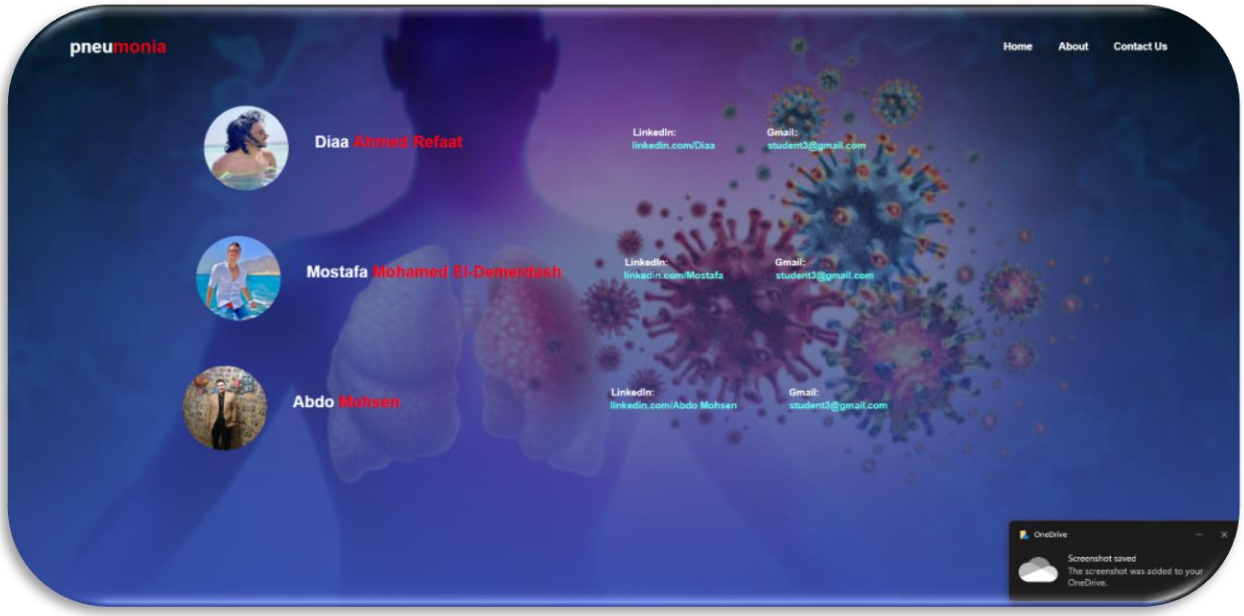
Home page:



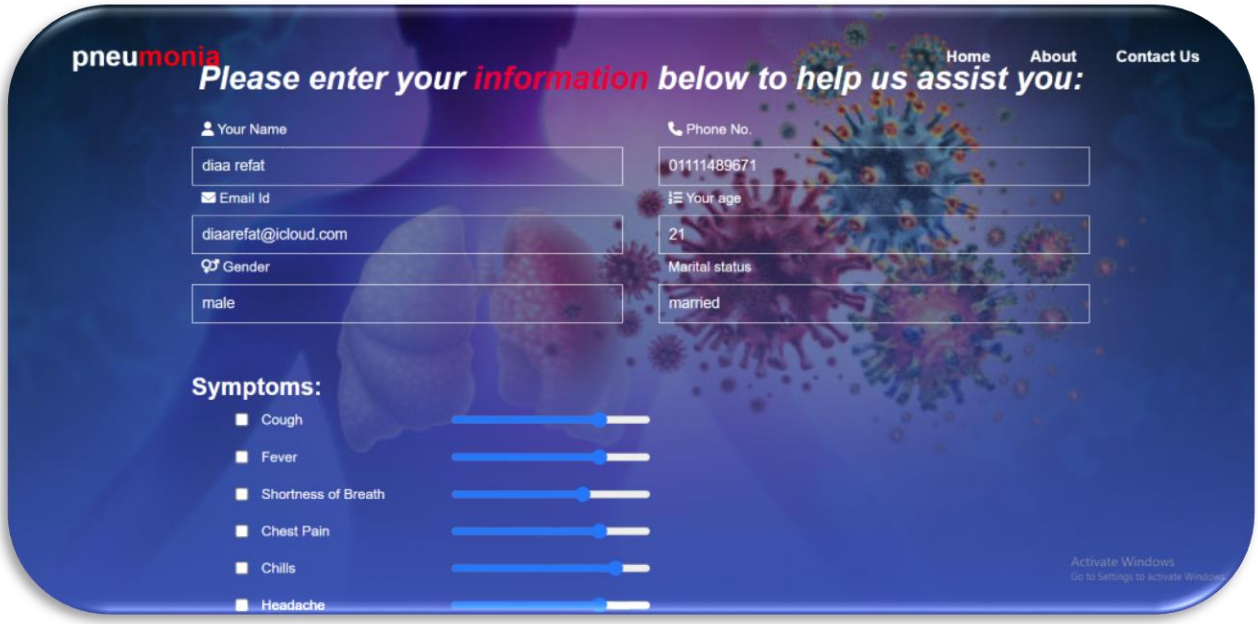
About page

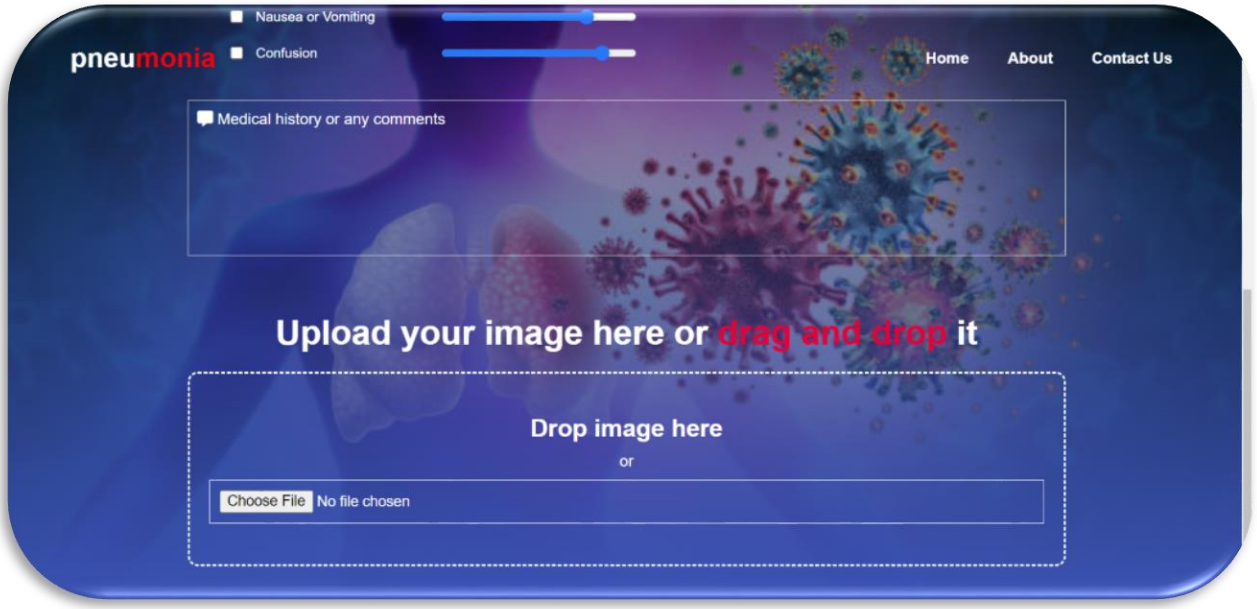

pneumonia
Home
About
Contact Us
About Detection Project
Our graduation project utilizes deep learning techniques to develop an automated system for pneumonia detection. By leveraging artificial intelligence, we aim to provide a reliable and efficient solution for diagnosing pneumonia from chest X-ray images.
Pneumonia is a lung infection that can be caused by various microorganisms, including bacteria, viruses, and fungi. It affects the air sacs in one or both lungs, causing inflammation and filling them with fluid or pus.
The proposed system analyzes the uploaded chest X-ray images using advanced algorithms to identify patterns and abnormalities associated with pneumonia. This can assist medical professionals in making accurate and timely diagnoses, leading to improved patient care and outcomes.
By combining the power of deep learning and medical imaging, we strive to contribute to the field of healthcare technology and enhance the efficiency of pneumonia diagnosis, ultimately benefiting patients and healthcare providers.

Contact us page

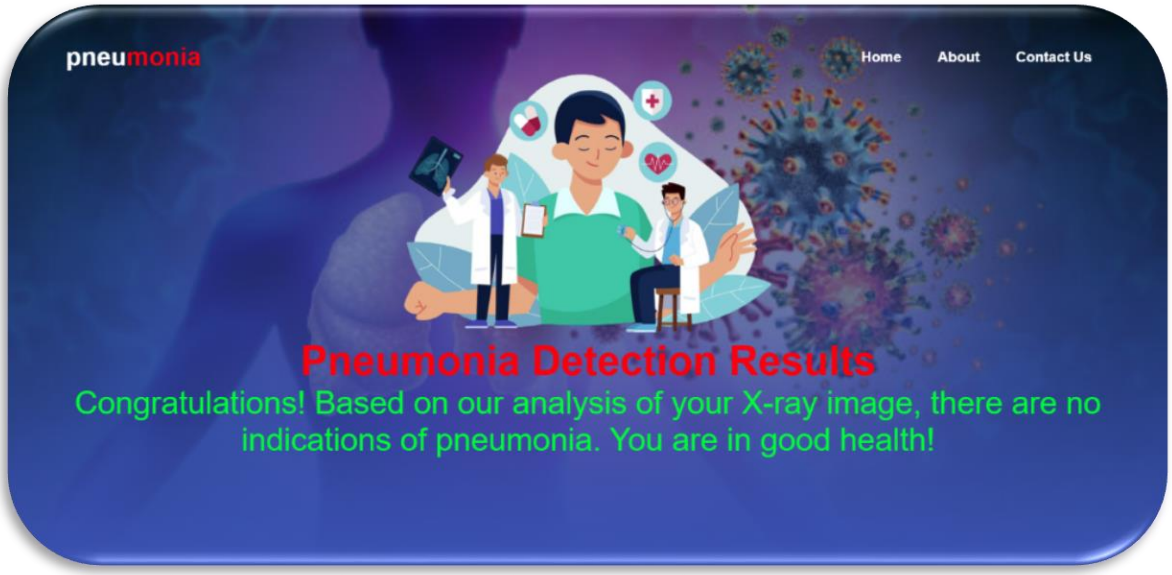


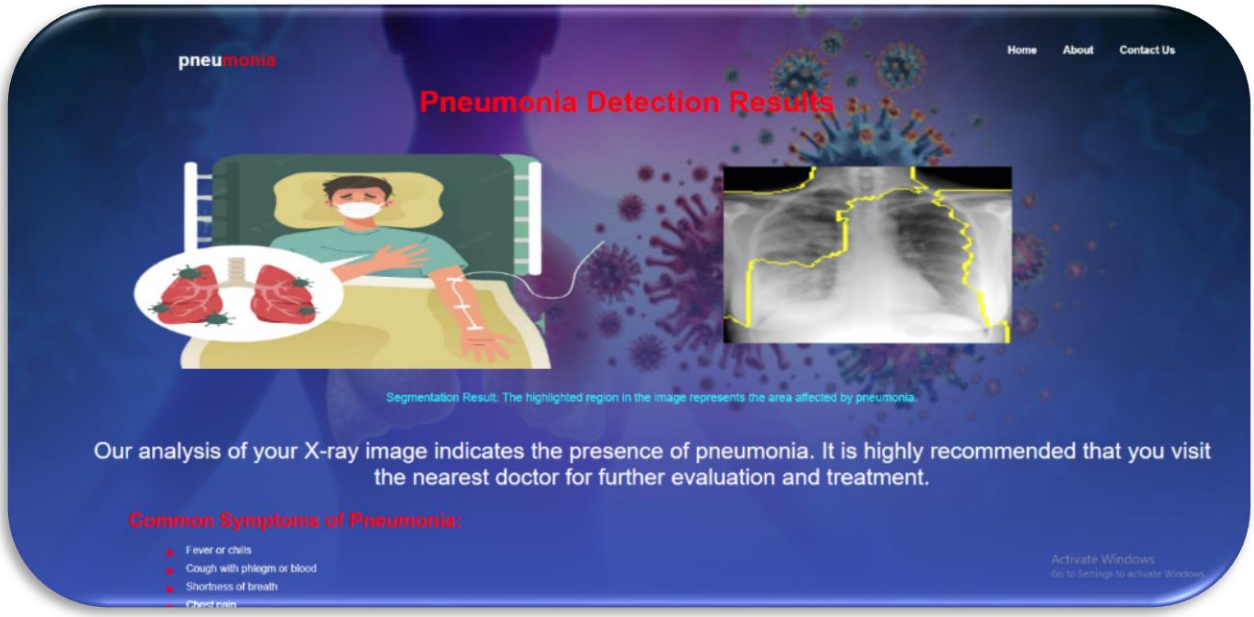
Form for information and medical history and upload photo





Result page


pneumonia
Home
About
Contact Us
Pneumonia Detection Results
Congratulations! Based on our analysis of your X-ray image, there are no indications of pneumonia. You are in good health!

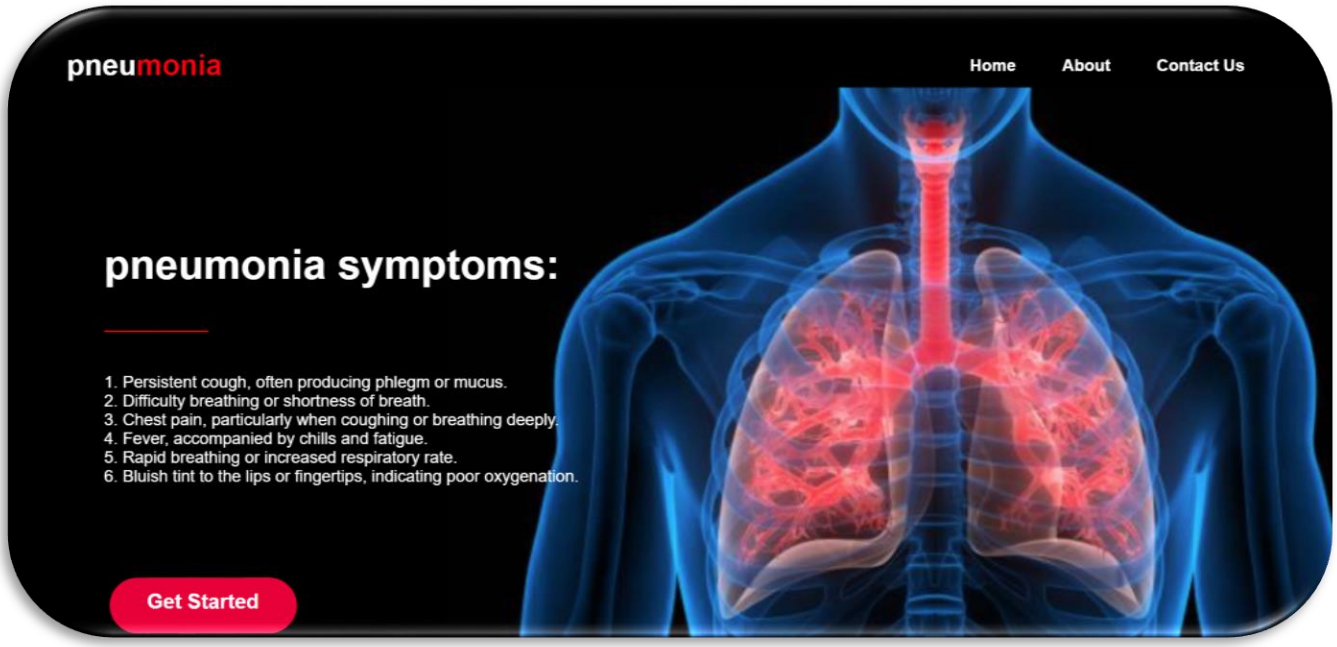

pneumonia
Home
About
Contact Us
Pneumonia Detection Results
Segmentation Result. The highlighted region in the image represents the area affected by pneumonia.
Our analysis of your X-ray image indicates the presence of pneumonia. It is highly recommended that you visit the nearest doctor for further evaluation and treatment.
Common Symptoms of Pneumonia:
Fever or chills
Cough with phlegm or blood
Shortness of breath
Chest pain
Activate Windows
Go to Settings to activate Windows.

Chapter 5 Implementation and testing

1- Web site testing

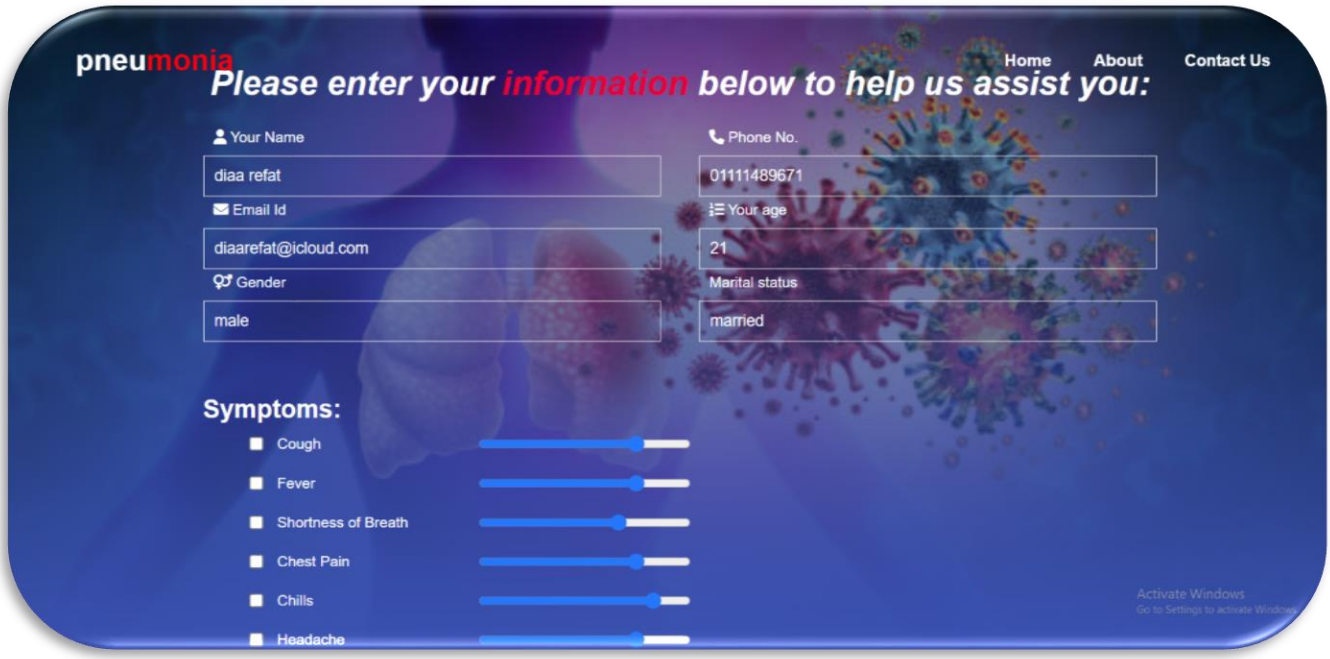
Home page

- **Here you will get some basic information about pneumonia and what the symptoms and also how to avoid it.**



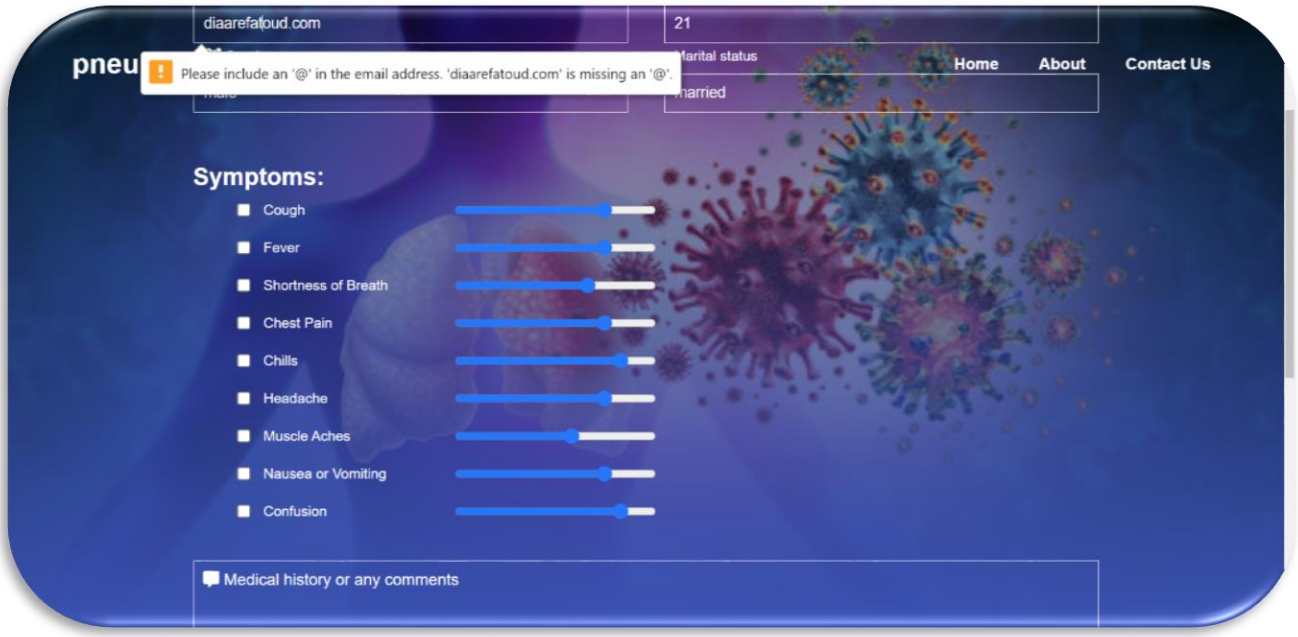
Personal information

- **After user click on get start, he will get this page he should enter all these field (Name, Email, Phone No, Gender, Age, Martial status)**

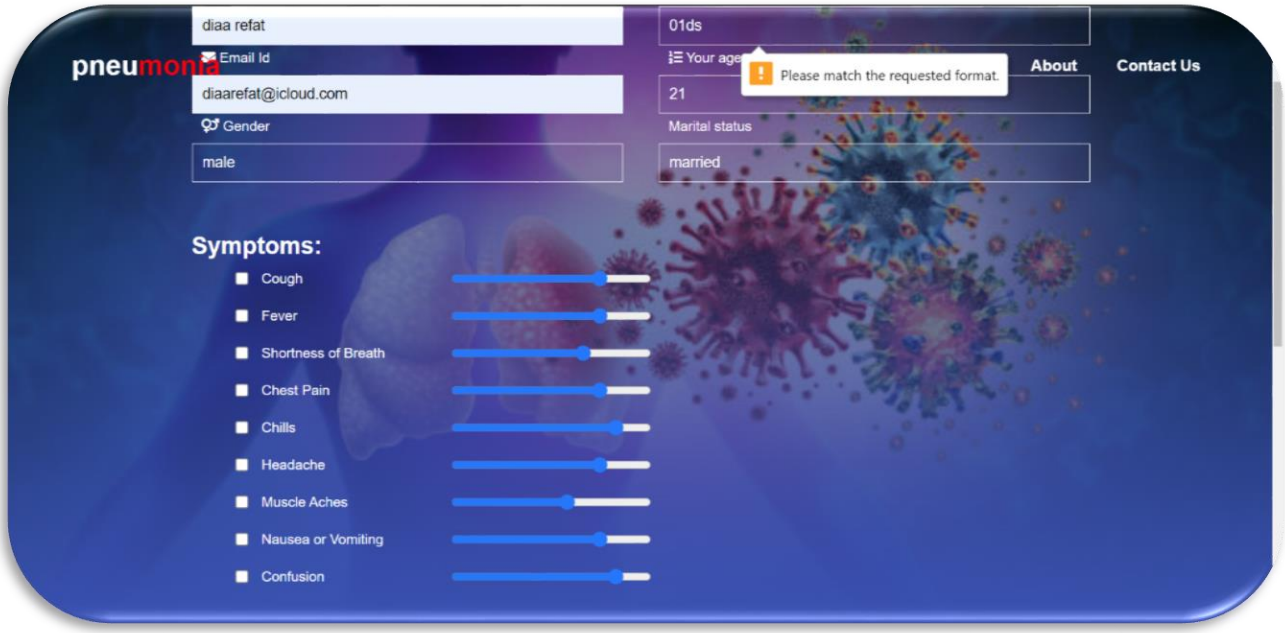


Validation on some information

- **We made some validation on some fields like email and phone no that should follow the format for example mail(@example)**



- **And phone number (format should be 11 number)**

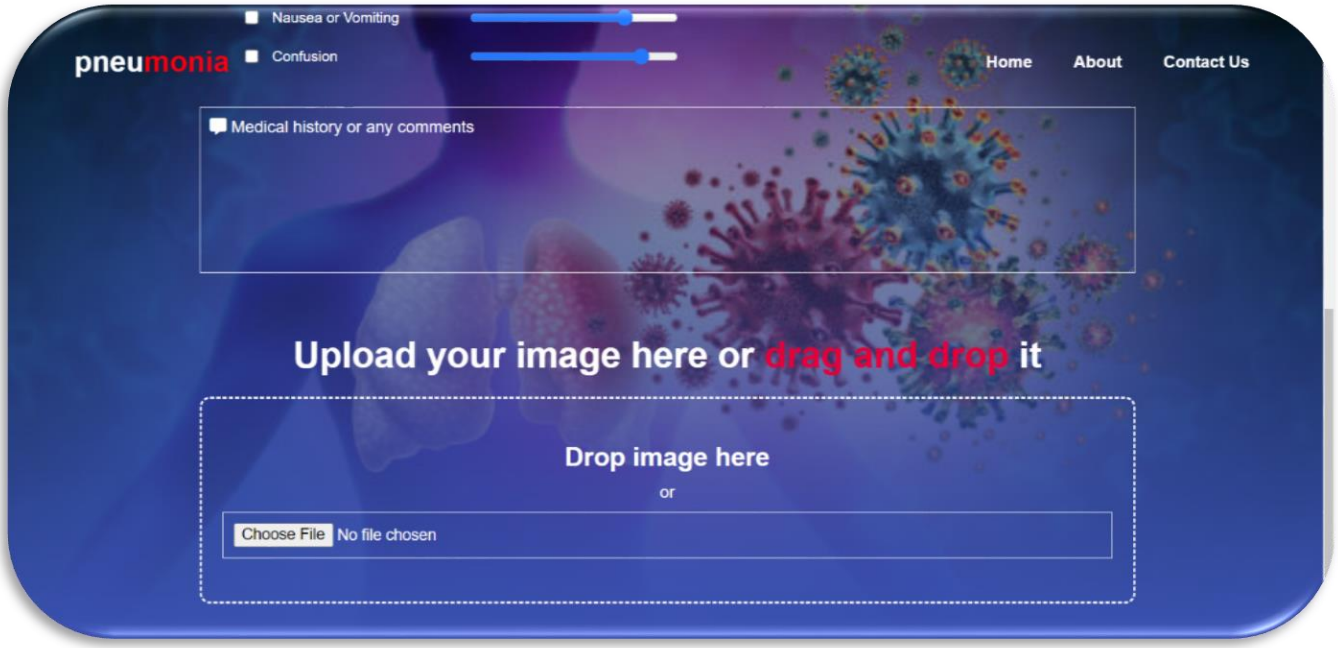


Select symptoms.

- **And he should choose what symptoms he has and also, he will choose the strength of these symptoms by the bar show in this photo.**

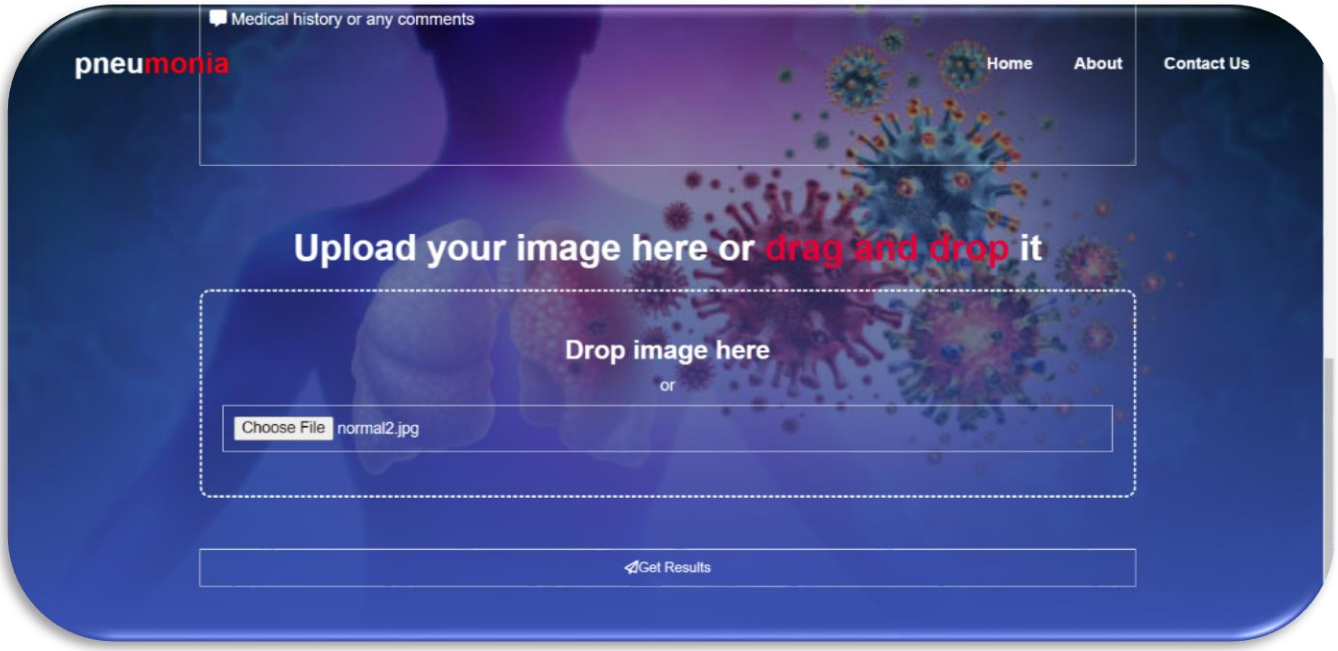
Medical history

- **After entering correct information and choose your strength of the symptoms**
- **You will write down if you have any comment and also your medical health else just enter none.**
- **In this case I entered that I have old heart attack**

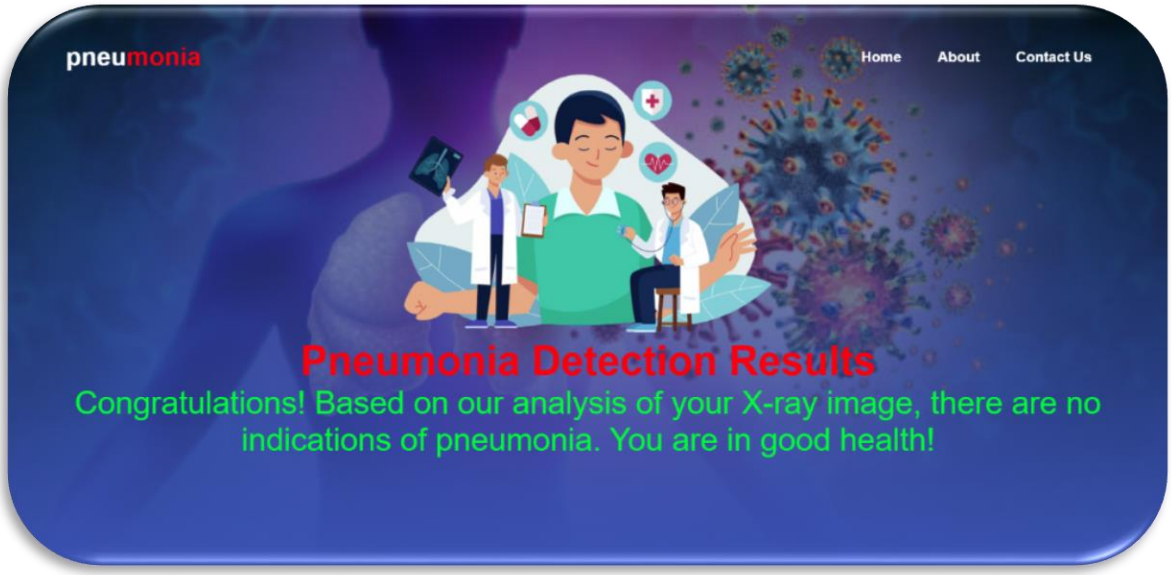


Upload photo

- **In this step you will collect your photo from your computer, or you will just drag the photo and put it on the box automatically photo will be loaded.**
- **And click the get results button.**
- **then you will wait for model to work on your photo so you will get your result**

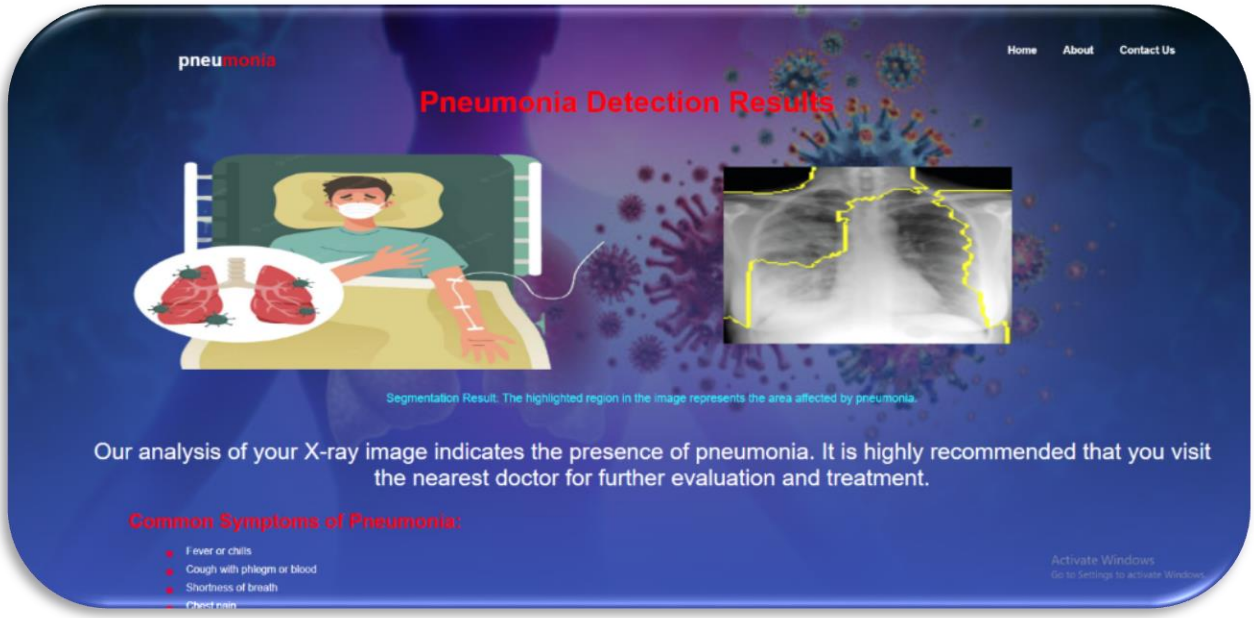


Case if you are normal.



Case if you have pneumonia.

In this case your photo will go to the model if it was affected, we will use an explanation to show why our model says that this x ray chest has pneumonia by making boundaries around the features that our model detects that it's affected.



2- Model Implementation and results:

- The code blow show how we get the data from train folder only because from comments test and valid folders have wrong labels and train folder have 5000 or more and this enough and we also but the images path and labels on array using for loop after that we concatenate them to a data frame.

```
images = glob('/kaggle/input/chest-xray-pneumonia/chest_xray/train/**/*.jpeg', recursive=False)
print(len(images))
import fnmatch
normal = fnmatch.filter(images, '*NORMAL*')
bacteria = fnmatch.filter(images, '*_bacteria_*')
virus = fnmatch.filter(images, '*_virus_*')
x = []
y = []
for img in images:
    x.append(img)
    if img in normal:
        y.append('NORMAL')
    elif img in bacteria:
        y.append('PNEUMONIA')
    elif img in virus:
        y.append('PNEUMONIA')
        print('no class')
x = np.array(x)
y = np.array(y)
Fseries = pd.Series(x, name= 'filepaths')
Lseries = pd.Series(y, name='labels')
df = pd.concat([Fseries, Lseries], axis= 1)
```

- Now let’s take a look at the distribution of data and use this code to see if data to resample to avoid misbalancing.

```
sns.countplot(x='labels', data=df)
df.labels.value_counts()
```

- From the graph it seems that the data is imbalanced, so we need to resample.

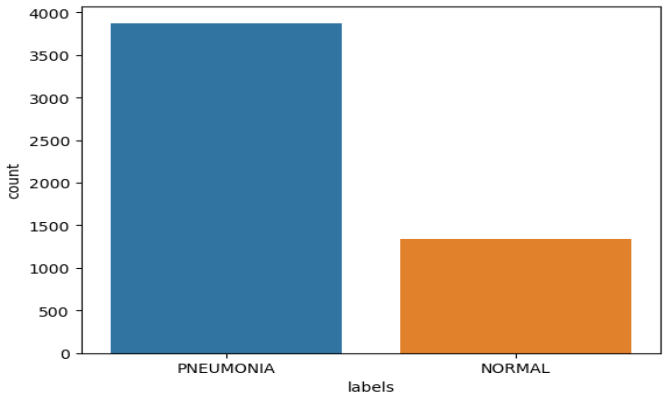


Fig 5.1 (histogram for the data before resampling)

- This code will resample data to make pneumonia samples as the same normal samples.

```
df_majority = df[df['labels']=='NORMAL']
df_minority = df[df.labels=='PNEUMONIA']
df_minority_upsampled = resample(df_minority,
                                 replace=True,     # sample with replacement
                                 n_samples=len(df_majority),    # to match majority class
                                 random_state=42)
df_upsampled = pd.concat([df_majority, df_minority_upsampled])
sns.countplot(x='labels', data=df_upsampled)
df_upsampled.labels.value_counts()
```

- Now the data is balanced let's go to the next stage.

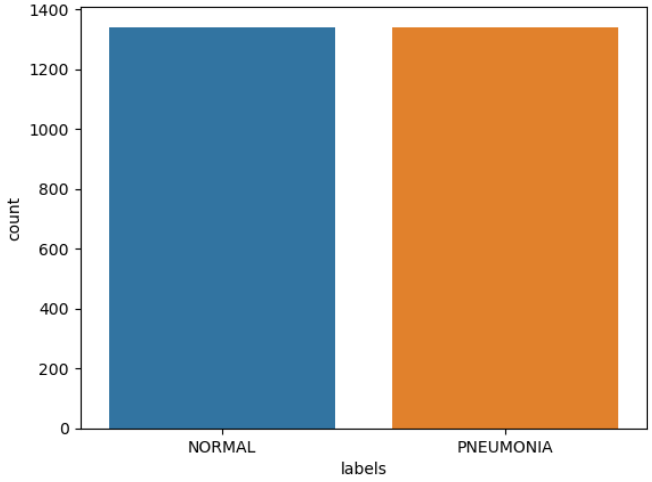
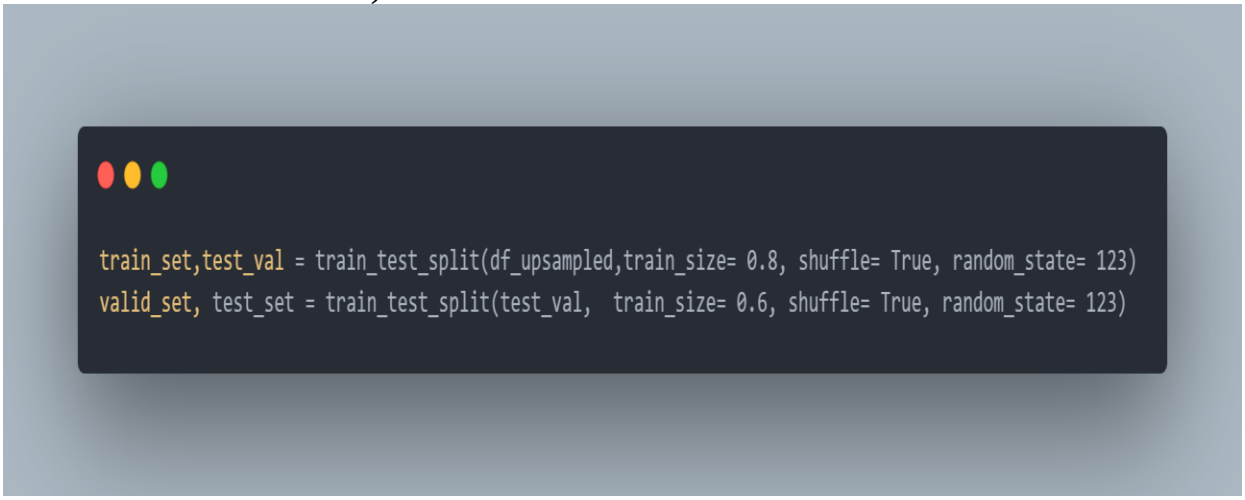


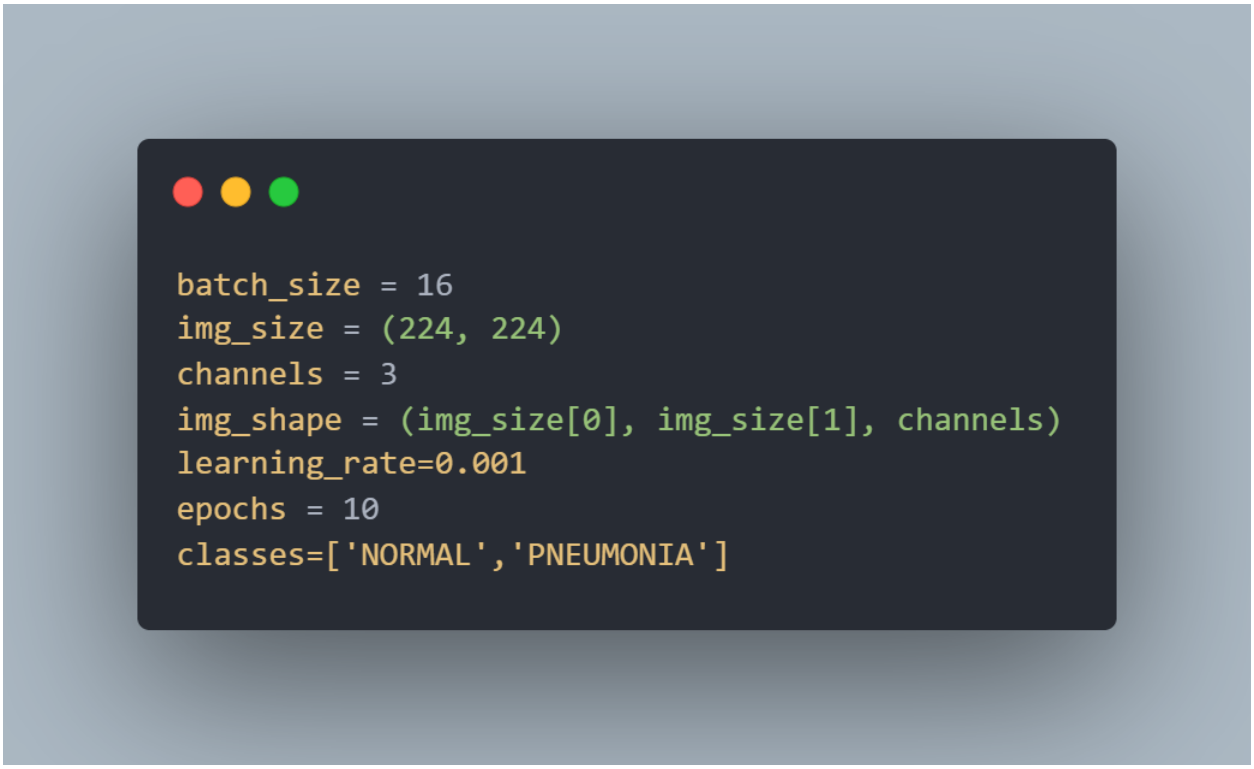
Fig 5.2 (histogram for the data after resampling)

- Now let’s split data to train and test and validation in this code we split the data into 80% train and remain 20% of data is test and validation (12% test and 8% validation)



```
train_set,test_val = train_test_split(df_upsampled,train_size= 0.8, shuffle= True, random_state= 123)
valid_set, test_set = train_test_split(test_val,  train_size= 0.6, shuffle= True, random_state= 123)
```

- This is some hyperparameters it differs from model to another, so we make a section for it for example for enetb4 we use image size 128*128 and batch size 64 and for vgg16 and resnet50 we used 224*224 image size and 16 batch.



```
batch_size = 16
img_size = (224, 224)
channels = 3
img_shape = (img_size[0], img_size[1], channels)
learning_rate=0.001
epochs = 10
classes=['NORMAL','PNEUMONIA']
```

- Now we need to do some image preprocessing we will be using augmentation with using image generator and parameters will be 0.5 rotation and width and height shift =0.1 and this enough for this kind of problem
- using scale make Val accuracy to stuck and that means that it stuck on local minima.
- so, we used only this parameter makes our model generalized for any kind of photo even it was shifted or have little quality.

```
train_gen  = ImageDataGenerator(rotation_range=5,
                                width_shift_range=0.1,
                                height_shift_range=0.1)
test_gen  = ImageDataGenerator()
train_gen = train_gen.flow_from_dataframe( train_set, x_col= 'filepaths', y_col= 'labels', target_size= img_size,
class_mode= 'categorical',
                                color_mode= 'rgb', shuffle= True, batch_size= batch_size)

valid_gen = test_gen.flow_from_dataframe( valid_set, x_col= 'filepaths', y_col= 'labels', target_size= img_size,
class_mode= 'categorical',
                                color_mode= 'rgb', shuffle= True, batch_size= batch_size)

test_gen = test_gen.flow_from_dataframe( test_set, x_col= 'filepaths', y_col= 'labels', target_size= img_size,
class_mode= 'categorical',
                                color_mode= 'rgb', shuffle= False, batch_size= batch_size)
```

This is the output:

```
Found 2145 validated image filenames belonging to 2 classes.
Found 322 validated image filenames belonging to 2 classes.
Found 215 validated image filenames belonging to 2 classes.
```

Now let’s implement useful callbacks.

- We used early stop with patience 5 that means if Val loss didn’t improve for 5 epochs so stop fitting.
- We also used model check point that saves the best weights only according to Val loss.
- And the last callback is reducing the learning rate on plateau with patience 5 which means if the Val loss didn’t improve for 5 epochs it will reduce the learning rate by 0.8.

```
from keras.callbacks import ModelCheckpoint, LearningRateScheduler, EarlyStopping, ReduceLROnPlateau
weight_path="{}.best_only.hdf5".format('save')

checkpoint = ModelCheckpoint(weight_path, monitor='val_loss', verbose=1,
                             save_best_only=True, mode='min', save_weights_only = True)

reduceLROnPlat = ReduceLROnPlateau(monitor='val_loss', factor=0.8,
                                   patience=5, verbose=1, mode='auto',
                                   epsilon=0.0001, cooldown=5, min_lr=0.0001)
early = EarlyStopping(monitor="val_loss",
                      mode="min",
                      patience=5)
callbacks_list = [checkpoint, early,reduceLROnPlat]
```

Now let’s implement our model here is some details.

- We used pretrained models for this task from keras applications we used vgg16 and resnet50 and enetb4.
- And after that we add batch normalization layer and Dense layer with 256 neurons and RELU function
- And we used drop out layer with rate 0.5 this randomly making the activation of 50% of the neurons in the input to zero so we are avoiding overfitting problem.
- Then the output layer is consisting of 2 neurons SoftMax
- Our loss function because of our problem classification categorical labels we used categorical cross entropy loss function.

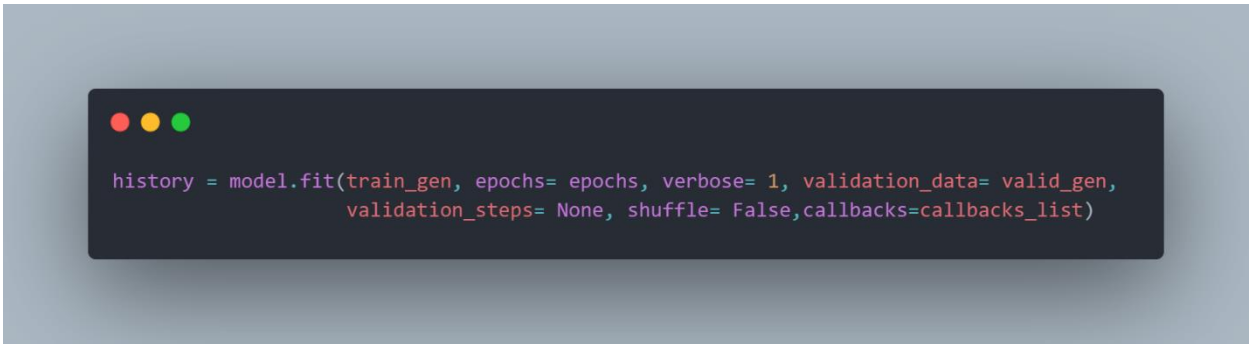
And here is the code.

```
base_model = tf.keras.applications.efficientnet.EfficientNetB4(include_top= False, weights= "imagenet",
input_shape= img_shape, pooling= 'max')

model = Sequential([
    base_model,
    BatchNormalization(axis= -1),
    Dense(256, activation= 'relu'),
    Dropout(rate= 0.5, seed= 123),
    Dense(2, activation= 'softmax')
])
model.compile(Adamax(learning_rate= learning_rate), loss= 'categorical_crossentropy', metrics= ['accuracy'])

model.summary()
```

- Now let's fit our model using the training set we made, and validation set and also number of epochs we decide is 10 because it's enough and make the model not fall into overfitting problem.



```
history = model.fit(train_gen, epochs= epochs, verbose= 1, validation_data= valid_gen,
                    validation_steps= None, shuffle= False,callbacks=callbacks_list)
```

- The last thing before going to results let's talk about our model explanation method called Lime.
- Lime is an explanation method that helps us to see which features lead to this prediction. We used this to explain to the user how our model detected if it's pneumonia by drawing boundaries around the features.
- We choose size of the neighborhood to learn the linear model = 1000 examples this take time but make the prediction more understandable by users

- This is the code below of the segmentation function which takes the image and returns the explainer image to be shown to the user.

```
def segmentation (image):
    image=Image.open(image)
    image=image.resize((128,128))
    image_arr = np.array(image)

    image_arr = image_arr / 255
    explanation = explainer.explain_instance(image_arr.astype('double'), model.predict,
                                    top_labels=10, hide_color=0, num_samples=1000)
    temp, mask = explanation.get_image_and_mask(explanation.top_labels[0], positive_only=True, num_features=10,
hide_rest=False,min_weight=0.5)
    fig,ax = plt.subplots(figsize=(7,7))
    ax.imshow(mark_boundaries(temp, mask))
    ax.axis('off')
    plt.axis('off')
    plt.savefig('diaa.png',transparent=True)
    plt.show()
```

Now let’s talk about our results for our three models using confusion matrix.

```
preds = model.predict(test_gen)
y_pred = np.argmax(preds, axis=1)
cm = confusion_matrix(test_gen.classes, y_pred)
sns.heatmap(cm,cmap= "Blues", linecolor = 'black' , linewidth = 1 , annot = True, fmt='')
```

- **Resnet50**

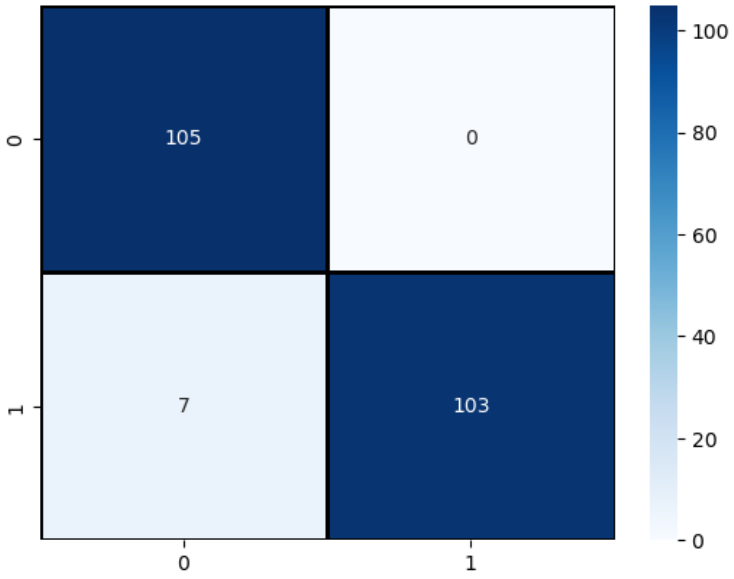


Fig 5.3(Enet confusion matrix)

- **VGG16**

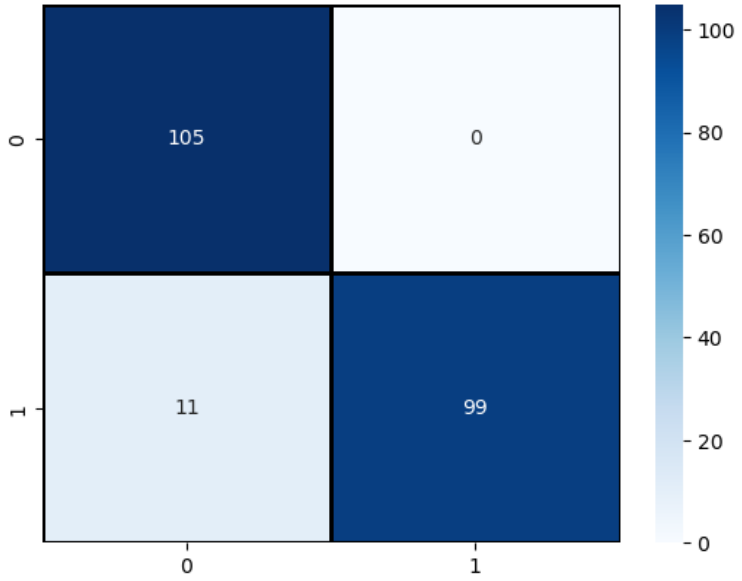


Fig 5.4(VGG 16 confusion matrix)

- **ENETB4**

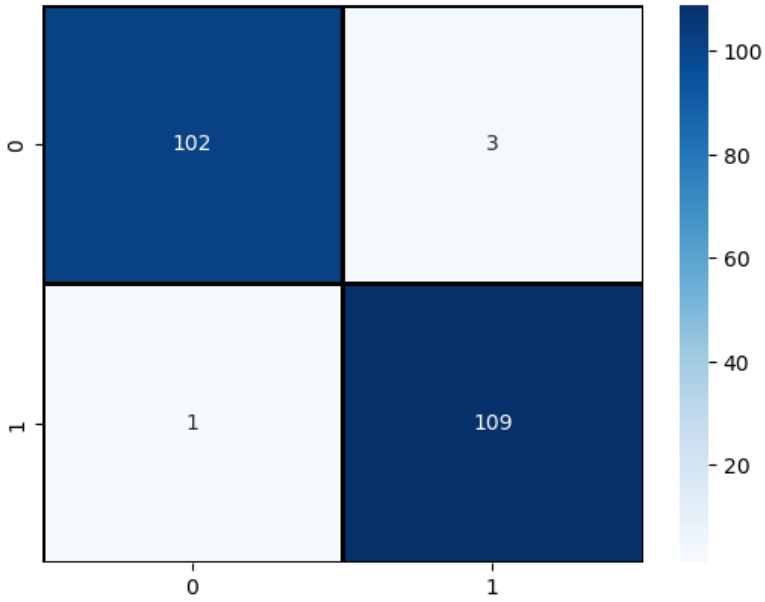


Fig 5.5(resnet50 confusion matrix)

Now let's evaluate two graphs to see is our model learning in good way or not. By plotting training vs validation loss and plotting training vs validation AUC

```
def create_charts(cnn, cnn_model):
    ## DEFINE ##
    ## Define 1:  train & validation loss
    train_loss = cnn_model.history['loss']
    val_loss = cnn_model.history['val_loss']

    ## Define 2: train & validation AUC
    train_auc_name = list(cnn_model.history.keys())[3]
    val_auc_name = list(cnn_model.history.keys())[1]
    train_auc = cnn_model.history[train_auc_name]
    val_auc = cnn_model.history[val_auc_name]

    ## Define 3: y_pred & y_true
    y_true = test_gen.classes
    Y_pred = cnn.predict_generator(test_gen, steps = len(test_gen))
    y_pred = (Y_pred > 0.5).T[0]
    y_pred_prob = Y_pred.T[0]

    ## PLOT ##
    fig = plt.figure(figsize=(13, 10))

    ## PLOT 1: TRAIN VS. VALIDATION LOSS
    plt.subplot(2,2,1)
    plt.title("Training vs. Validation Loss")
    plt.plot(train_loss, label='training loss')
    plt.plot(val_loss, label='validation loss')
    plt.xlabel("Number of Epochs", size=14)
    plt.legend()

    ## PLOT 2: TRAIN VS. VALIDATION AUC
    plt.subplot(2,2,2)
    plt.title("Training vs. Validation AUC Score")
    plt.plot(train_auc, label='training auc')
    plt.plot(val_auc, label='validation auc')
    plt.xlabel("Number of Epochs", size=14)
    plt.legend()
```

And these are the results on the three models.

- **Resnet50**

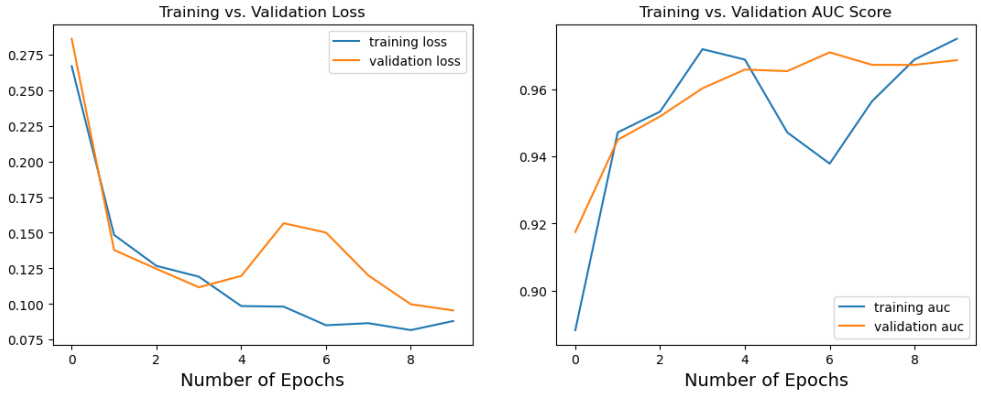


Fig 5.6(resnet 50 evaluation graph)

- **VGG16**

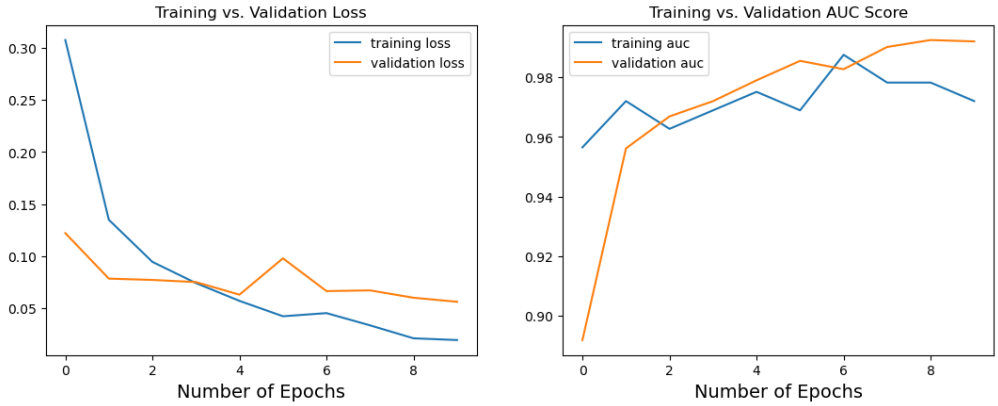


Fig 5.7(vgg16 evaluation graph)

- **ENETB4**

Fig 5.8(enet evaluation graph)

To see the precision and recall of models and also f1 score we can use classification report to see them for each model.

- **Resnet50**

	precision	recall	f1-score	support
NORMAL	0.94	1.00	0.97	105
PNEUMONIA	1.00	0.94	0.97	110
accuracy			0.97	215
macro avg	0.97	0.97	0.97	215
weighted avg	0.97	0.97	0.97	215

Fig 5.9(resnet50 classification report)

- **VGG16**

	precision	recall	f1-score	support
NORMAL	0.91	1.00	0.95	105
PNEUMONIA	1.00	0.90	0.95	110
accuracy			0.95	215
macro avg	0.95	0.95	0.95	215
weighted avg	0.95	0.95	0.95	215

Fig 5.10(vgg16 classification report)

- **ENETB4**

	precision	recall	f1-score	support
NORMAL	0.99	0.97	0.98	105
PNEUMONIA	0.97	0.99	0.98	110
accuracy			0.98	215
macro avg	0.98	0.98	0.98	215
weighted avg	0.98	0.98	0.98	215

Fig 5.11(enet classification report)

- From these graphs we could say from our three models we can say that Enetb4 model is the most effective one which it gives us high accuracy and also more generalized for new data
- And this is a comparison between them.

Arch name	VGG16	Resnet50	ENETB4
Accuracy	0.9604	0.9688	0.9921
F1 SCORE	0.95	0.97	0.98
Acuuracy on test	0.948	0.967	0.981

table 5.1(compare accuracy between the three models)

- So, we now ready to use this model for the next task is to deploy it into website using flask and HTML and CSS
- Our website uses a database to store all the information which we take from the user and but it on the table shows on ERD.
- By saving all these data we can go to future work we mentioned it in separate section to discuss in detail what we are planning in the future
- The last thing that we need to know is that our model is still a model with accuracy and there can be a wrong prediction so we will tell the user this thing on the page of prediction.
- And we will advise the user to see the doctor as soon as it possible and the doctor can check our boundaries which why our model predict it as pneumonia so he will decide with data of symptoms stored also in the database if he need to take action or the model prediction is wrong .

3- What issues we got and how we solved them.

- We had some issues which we learn from it and also fix it we will discuss it in brief way.

1) We have a problem with data, it was imbalanced and also had wrong labels and we talked about it in the implementation section you can check it.

2) We have a problem that's called overfitting which means that the model learn the data too much so if we upload any new data we will get wrong results , we were suffering from this problem but it was invisible because when we evaluate accuracy we find test is very good around 99% but when we tried the same image with little noise we surprised that our model predicted it in wrong way.
   So how we fixed it.
   - For this problem we added some noise to training data by adding rotation etc.
   - We also used more complex architecture at first, we were using efficient net B0 this version gives us 99% accuracy, but we found that it doesn't generalize the data.
   - So, preferred to use efficient net B4 which has more parameters so it's more complex and we found that is very good with new data.

3) The last problem we found was that when we were using rescale method on augmentation the validation accuracy stuck at global minimum so decided not to rescale photos and make it original, so we get high accuracy.

4- Future work:

Website future work

- In this section we will talk about future of the website
- We will add on our website access to user to retrieve all his predictions from his name just write his name and we will retrieve all data from the table on the database.
- And also, we will add a button on the website for users to download our database as excel sheet for some reasons:

    - First this will make our dataset useful for future training because we will have a big data with new photos from users and also other features like medical history and symptoms you can train your machine learning model using this feature and also use CNN for images classification.
    - Second this will make it a way for health organizations to make analysis on our data and they will get results of what symptoms always come with pneumonia and in which ages and also which gender have this disease more than another one and another medical analysis.
    - This will reduce the numbers of infected people because we have a tool to make sure which symptoms to care for most.

References:

Cleveland clinic (information)
https://my.clevelandclinic.org/health/diseases/4471-pneumonia

paper 1 by (V. Sirish Kaushik Anand Nayyar Gaurav Kataria Rachna Jain)
https://www.researchgate.net/publication/340961287_Pneumonia_Detection_Using_Convolutional_Neural_Networks_CNNs

paper 2 by (Patrik Szepesi [a], László Szilágyi)
https://www.sciencedirect.com/science/article/pii/S0208521622000742

architectures information from (MathWorks)
https://www.mathworks.com/help/deeplearning/ref

callbacks in keras
https://www.kdnuggets.com/2019/08/keras-callbacks-explained-three-minutes.html

Data of the project from Kaggle
https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

explainer model (LIME)
https://github.com/marcotcr/lime/tree/master

Dropout: A Simple Way to Prevent Neural Networks from Overfitting (Nitish Srivastava Geoffrey Hinton Alex Krizhevsky Ilya Sutskever Ruslan Salakhutdinov)
https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

loss function and optimizers and etc. API keras
https://keras.io/api/