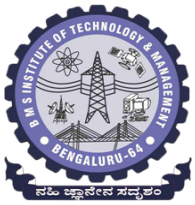
BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

YELAHANKA, BENGALURU - 560064



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROJECT BASED LEARNING

2020-21 Even Semester

Report of SSCD & WTA - 18CS61 & 18CS63 project work

"LUNG DISEASE DETECTION USING DEEP LEARNING"

Submitted By

Siddhanth Tripathi	USN: 1BY18CS164
Sinchana Shetty	USN: 1BY18CS165
Somil Jain	USN: 1BY18CS167
Vanshika Sharma	USN: 1BY18CS185

Under the guidance of

Dr. Manoj H M
Assistant Professor

2020-2021

INSTITUTE VISION

To emerge as one of the finest technical institutions of higher learning, to develop engineering professionals who are technically competent, ethical and environment friendly for betterment of the society.

INSTITUTE MISSION

Accomplish stimulating learning environment through high quality academic instruction, innovation and industry-institute interface.

DEPARTMENT VISION

To develop technical professionals acquainted with recent trends and technologies of computer science to serve as valuable resources for the nation/society.

DEPARTMENT MISSION

Facilitating and exposing the students to various learning opportunities through dedicated academic teaching, guidance and monitoring.

PROGRAM EDUCATIONAL OBJECTIVES

1. Lead a successful career by designing, analyzing and solving various problems in the field of Computer Science & Engineering.
2. Pursue higher studies for enduring edification.
3. Exhibit professional and team building attitude along with effective communication.
4. Identify and provide solutions for sustainable environmental development.

SSCD – 18CS61- Course Outcomes (COs) w.r.t this PBL	
CO #	CO DEFINED
	ASK YOUR FACULTY ABOUT THIS

WTA – 18CS63- Course Outcomes (COs) w.r.t this PBL	
CO #	CO DEFINED
	ASK YOUR FACULTY ABOUT THIS

Project to Program Outcomes (PO) Mapping

Project Name: Lung Disease Detection Using Deep Learning

COURSE	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
SSCD	✔	✔	✔	✔	✔			✔	✔	✔	✔	✔
WTA	✔	✔	✔	✔	✔			✔	✔	✔	✔	✔

Program outcomes (POs):	
PO1	**Engineering knowledge:** Apply the knowledge of Mathematics, Science, Engineering fundamentals and an engineering specialization to the solution of complex engineering problems
PO2	**Problem analysis:** Identify, formulate, review research literature, and analyse complex Engineering problems reaching substantiated conclusions using first principles of mathematics, Natural sciences and engineering sciences
PO3	**Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the Information to provide valid conclusions
PO5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern Engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for Sustainable development
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings
PO10	**Communication:** Communicate effectively on complex engineering activities with the engineering Community and with society at large, such as, being able to

	comprehend and write effective reports And design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance:** Demonstrate knowledge and understanding of the Engineering and management principles and apply these to one's own work, as a member and Leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Project to Program Specific Outcomes (PSO) Mapping

Project Name: Lung Disease Detection Using Deep Learning

COURSE	PSO1	PSO2
SSCD	✔	✔
WTA	✔	✔

Program Specific Outcomes (PSOs):	
PSO1	Analyze the problem and identify computing requirements appropriate to its solution.
PSO2	Apply design and development principles in the construction of software systems of varying complexity.

ABSTRACT

The World Health Organization (WHO) estimates that there are 300 million people who suffer from asthma, and that this disease causes around 250 thousand deaths per year worldwide . For the public health system, the early and correct diagnosis of any pulmonary disease is mandatory for timely treatment and prevents further death. From a clinical standpoint, diagnosis aid tools and systems are of great importance for the specialist and hence for the people's health. Thus, for pulmonary disease analysis and diagnosis, it is necessary to segment lung structures. It is worth noting that segmentation is an essential step in image systems for the accurate lung disease diagnosis, since it delimits lung structures in X-ray images. Indeed, image processing techniques can help computer diagnosis if lung region is accurately obtained

Learning about Lung Diseases and their characterization is one of the most interesting research topics in recent years. With the various uses of medical images in hospitals, pathologies, and diagnostic centers, the size of the medical image datasets is also expanding expeditiously to capture the diseases in hospitals. Though a lot of research has been done on this particular topic still this field is confusing and challenging. There are lots of techniques in literature to classify medical images. The main drawback of traditional methods is the semantic gap that exists between the low-level visual information captured by imaging devices and high-level semantic information perceived by a human being.

A publicly available dataset provided by National Institute of Health(NIH) called Chest X-Ray which consists of fifteen classes named Atelectasis, Cardiomegaly, Effusion, Infiltration, Mass, Nodule, Pneumonia, Pneumothorax, Consolidation, Edema, Emphysema, Fibrosis, Pleural Thickening, Hernia and No Finding images are used to train this model. This model gives a good result in multiclass classification.The average accuracy of 89.77% is achieved for the classification of different diseases. The comparative analysis shows the effectiveness of the proposed model. The proposed technique is best suited for classifying the multiclass medical images for different diseases.Each output unit will predict the probability of one of the fifteen diseases.

INTRODUCTION

Developments in artificial intelligence has taken place at an extraordinary pace in the past decade and it has found its way into countless applications from driverless cars to medical diagnosis. In recent years, artificial intelligence is being increasingly applied to diagnose obstructive lung diseases. Its primary goal has been to approximate human cognition in the analysis of complex medical data obtained from diagnostic tests for obstructive lung diseases. Deep learning is a type of machine learning where we learn from a large amount of data which involves artificial intelligence, neural networks and algorithms inspired by the human brain. It is the new perimeter for enterprise application and is a great assurance for development in almost every field.

We combine this knowledge and apply machine learning as a realworld tool deployed to solve the problems of medical big data, having wide complexity and difficulty in handling. Identifying and diagnosing diseases are one of the many healthcare challenges we apply this knowledge to. We will use Machine Learning as well as Deep Learning to process data as well as create models for diagnosing patients. My key point here will be: combining the processing of patient information with data from X-rays, using CNN with the well-known pre-trained model.

We aim to produce a taxonomy of the state-of-the-art machine learning based lung disease detection systems.Analysing the trends of recent work on the domain, we will identify the remaining issues.We will be describing the general processes of using machine learning to detect lung disease in X-Ray images.

Motivation

The world is changing so fast that the pressure on health is increasing, the adverse changes in climate, the environment, the lifestyle of humans, increase the risk as well as diseases for people. Lungs are one of the organs severely affected. More than 3 million people succumbed in 2020 to chronic obstructive pulmonary disease (COPD), caused mainly by smoking and pollution, while 400,000 people died from asthma. With so many lung diseases people can get, here is just one example of diseases we can save if we find them out earlier. With the technology, machines and computer power, the earlier identification of diseases, particularly lung disease, can be helped to detect earlier and more accurately, which can save many many people as well as reduce the pressure on the system. The health system has not developed in time with the development of the population. With the power of computers as well as the large amount of data being released to the public, this is a good time to contribute to solving this problem. The aim is to contribute more to the community by reducing medical costs, helping those who are not able to pay for medical expenses.

Existing System

In the diagnosis of radiological images, detection and classification processes are important.For assisting radiologists' diagnosis, computer-aided diagnosis (CAD) systems include two types of CAD algorithms such as a computer-aided detection (CADe) that detect abnormal lesion, and a computer aided diagnosis (CADx) that differentiate abnormal lesion into benign or malignant.In previous CAD algorithms, we used image features that could detect and classify lung abnormalities such as lung nodules or diffuse lung disease patterns. These image features are useful for computer-aided classification on lung diseases. However, to define such image features is a difficult task due to complicated image patterns of lung diseases. Machine learning technique has dramatically improved the state-of-the art in pattern recognition in the fields of speech and vision.

Limitations of Existing System

In spite of launching the first CAD system for detecting lung nodules or affected lung cells in the late 1980s, those efforts were not enough. This is because there were many inadequate computational resources for the implementation of advanced image processing techniques at that time. Lung disease detection using basic image processing techniques is also time consuming. After the successful invention of GPU and CNN, the performance of CAD (for lung disease diagnosing) and decision support arrangement got a high boost. Many studies propose many deep learning models in order to detect lung cancer and other lung diseases. The work focuses on the detection of multiple diseases. A 3D deep CNN is proposed with multiscale prediction strategies in order to detect the lung nodules from segmented images.

However, the work cannot classify disease types and the multiscale prediction approaches are applied for small nodules. A full CNN is proposed for the reduction of false positive rate in classifying the lung nodules. This method can only analyze the nature of the CT scan images in order to reduce the probability of wrong diagnosis. A machine learning method is also proposed where several transfer learning methods such as DenseNet121, AlexNet, Inception V3, etc., are used for pneumonia diagnosis. However, the parameter tuning for their implemented methods are very complex.

Proposed System

The datasets used by the surveyed works are reported in this section. Tables 5–8 show the summary of datasets used for tuberculosis, pneumonia, lung cancer and COVID-19 detection, respectively. This is done to provide readers with relevant information on the datasets. Note that only public datasets are included in the tables because they are available to the public, whereas private datasets are inaccessible without permission. According to Table 5, among the twelve datasets used for tuberculosis detection works, five of them do not contain tuberculosis medical images: JSRT dataset, Indiana dataset, NIH-14 dataset, LDOCTCXR and RSNA pneumonia dataset. JSRT dataset contains lung cancer images, while the Indiana and NIH-14 datasets contain multiple different diseases. LDOCTCXR and RSNA pneumonia datasets both contain pneumonia and normal lung images. These five datasets were used for transfer learning in several studies. Models were first trained to identify abnormalities in chest X-ray, and then they were trained to identify tuberculosis. The India, Montgomery and Shenzhen datasets contain X-ray images of tuberculosis; ImageCLEF 2018 and ImageCLEF 2019 datasets contain CT images of tuberculosis; and the Belarus dataset contains both X-ray and CT images of tuberculosis. Two of the datasets contain sputum smear microscopy images of tuberculosis: the TBimages dataset and ZiehlNeelsen Sputum smear Microscopy image DataBase. For detection works related to pneumonia, only four public datasets are available, as shown in Table 6. All four datasets contain X-ray images only. Even though the number of datasets is low, the number of images within these datasets is high. Future studies utilising these datasets should have sufficient data.

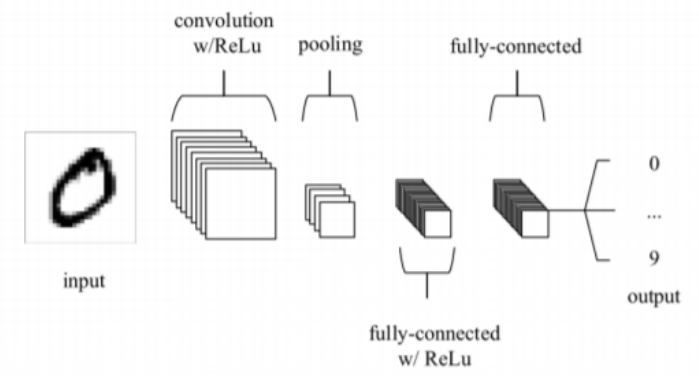


Figure 9. Example of a CNN structure.

System Requirement Specifications (Functional & Non-Functional)

Functional Requirements

Python based Deep Learning libraries will be exploited for the development and experimentation of the project. Tools such as Anaconda Python, visual studio and python libraries will be utilized for this process. Training will be conducted for training a probabilistic modeling and deep learning approach for disease prediction. We can use medical hardware devices for capturing the real data or test the results on real-time data.

Non-Functional Requirements

- A non-functional requirement tells us about the system's behavior.
- This also specifies how are the system's quality characteristics or quality attributes.
- The system is highly reliable.
- Resource consumption is quite low.
- We can add more resources to our project without disturbing the current scenario.

Proposed Methodology

1. The Basic Process to Apply Deep Learning for Lung Disease Detection

In this section, the process of how deep learning is applied to identify lung diseases from medical images is described. There are mainly three steps: image preprocessing, training and classification. Lung disease detection generally deals with classifying an image into healthy lungs or disease-infected lungs. The lung disease classifier, sometimes known as a model, is obtained via training. Training is the process in which a neural network learns to recognise a class of images. Using deep learning, it is possible to train a model that can classify images into their respective class labels. Therefore, to apply deep learning for lung disease detection, the first step is to gather images of lungs with the disease to be classified. The second step is to train the neural network until it is able to recognise the diseases. The final step is to classify new images. Here, new images unseen by the model before are shown to the model, and the model predicts the class of those images. The overview of the process is illustrated in Figure 2.

1.1. Image Acquisition Phase

The first step is to acquire images. To produce a classification model, the computer needs to learn by example. The computer needs to view many images to recognise an object. Other types of data, such as time series data and voice data, can also be used to train deep learning models. In the context of the work surveyed in this paper, the relevant data required to detect lung disease will be images. Images that could be used include chest X-ray, CT scan, sputum smear microscopy and histopathology image. The output of this step is images that will later be used to train the model.

1.2. Preprocessing Phase

The second step is preprocessing. Here, the image could be enhanced or modified to improve image quality. Contrast Limited Adaptive Histogram Equalisation (CLAHE) could be performed to increase the contrast of the images . Image modification such as lung segmentation and bone elimination could be used to identify the region of interest (ROI), whereby the detection of the lung disease can then be performed on the ROI. Edge detection could also be used to provide an alternate data representation . Data augmentation could be applied to the images to increase the

amount of available data. Feature extraction could also be conducted so that the deep learning model could identify important features to identify a certain object or class. The output of this step is a set of images whereby the quality of the images is enhanced, or unwanted objects have been removed. The output of this step is images that were enhanced or modified that will later be used in training.

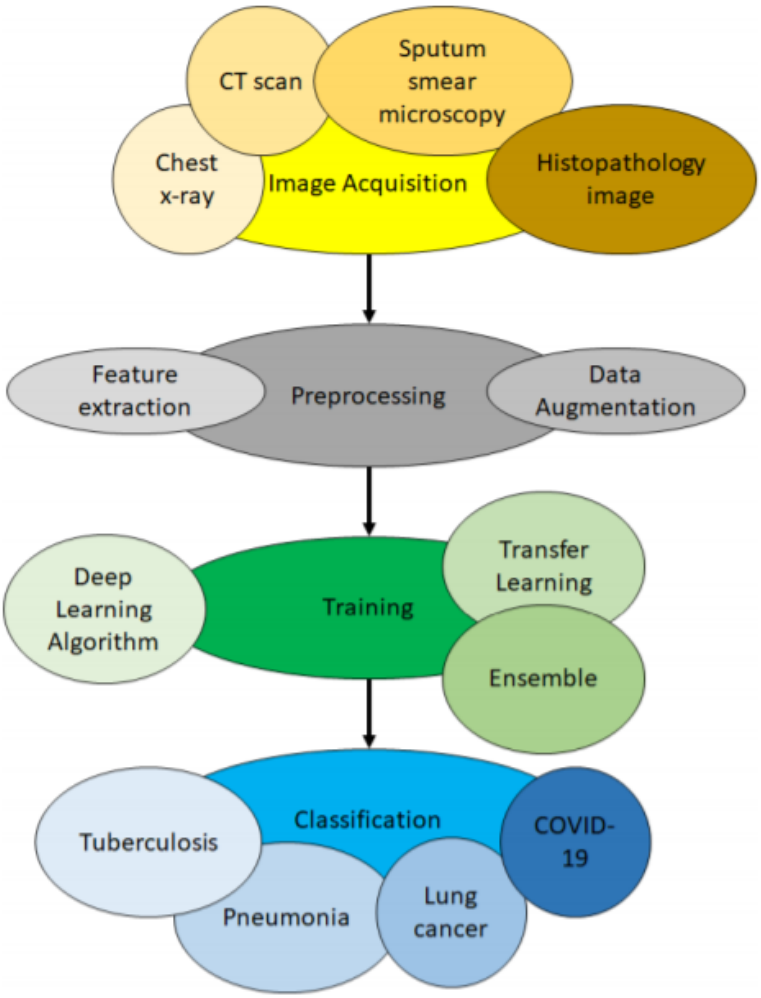


Figure 2. Overview of using deep learning for lung disease detection.

1.3. Training Phase

In the third step, namely training, three aspects could be considered. These aspects are the selection of deep learning algorithms, usage of transfer learning and usage of an ensemble. There are numerous deep learning algorithms, for example deep belief network (DBN), multilayer perceptron neural network (MPNN), recurrent neural network (RNN) and the aforementioned

CNN. Different algorithms have different learning styles. Different types of data work better with certain algorithms. CNN works particularly well with images. Deep learning algorithm should be chosen based on the nature of the data at hand. Transfer learning refers to the transfer of knowledge from one model to another. Ensemble refers to the usage of more than one model during classification. Transfer learning and ensemble are techniques used to reduce training time, improve classification accuracy and reduce overfitting. Further details concerning these two aspects could be found. The output of this step is models generated from the data learned.

1.4. Classification Phase

In the fourth and final step, which is classification, the trained model will predict which class an image belongs to. For example, if a model was trained to differentiate X-ray images of healthy lungs and tuberculosis-infected lungs, it should be able to correctly classify new images (images that are never seen by the model before) into healthy lungs or tuberculosis-infected lungs. The model will give a probability score for the image. The probability score represents how likely an image belongs to a certain class. At the end of this step, the image will be classified based on the probability score given to it by the model.

2. The Taxonomy of State-Of-The-Art Work on Lung Disease Detection Using Deep Learning

In this section, a taxonomy of the recent work on lung disease detection using deep learning is presented, which is the first contribution of this paper. The taxonomy is built to summarise and provide a clearer picture of the key concepts and focus of the existing work. Seven attributes were identified for inclusion in the taxonomy. These attributes were chosen as they were imminent and can be found in all the articles being surveyed. The seven attributes included in the taxonomy are image types, features, data augmentation, types of deep learning algorithms, transfer learning, the ensemble of classifiers and types of lung diseases. Sections 4.1–4.7 describe each attribute in detail, whereby the review of relevant works is provided. Section 4.8 describes the datasets used by the works surveyed. Figure 3 shows the taxonomy of state-of-the-art lung disease detection using deep learning.

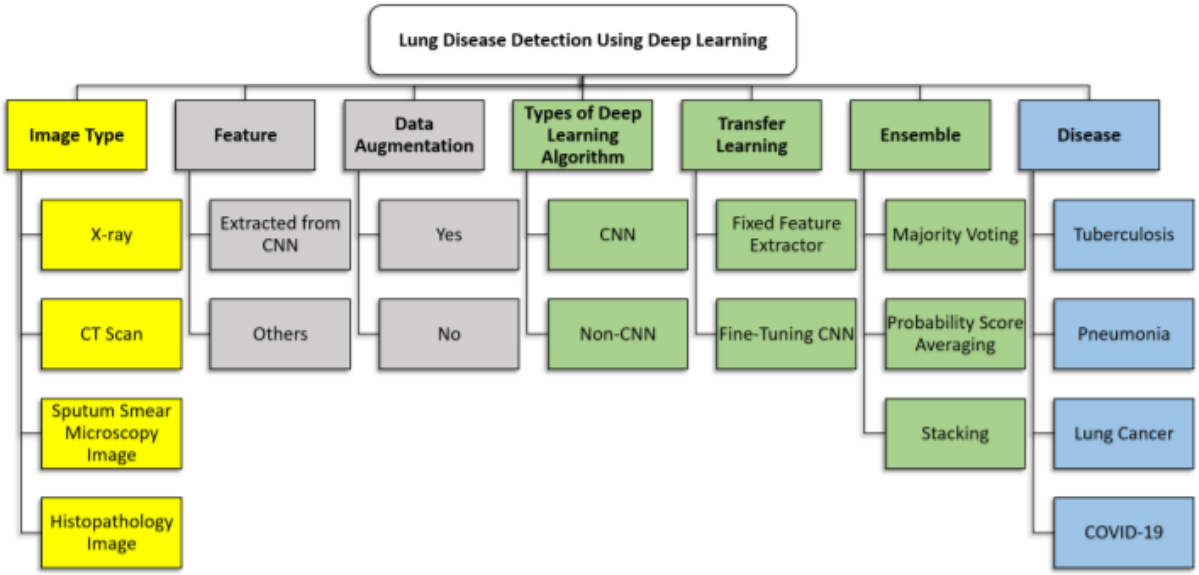


Figure 3. Taxonomy of lung disease detection using deep learning.

2.1. Image Type

In the papers surveyed, four types of images were used to train the model: chest X-ray, CT scans, sputum smear microscopy images and histopathology images. These images are described in detail in Sections 4.1.1–4.1.4. It should be noted that there are other imaging techniques exist such as positron emission tomography (PET) and magnetic resonance imaging (MRI) scans. Both PET and MRI scans could also be used to diagnose health conditions and evaluate the effectiveness of ongoing treatment. However, none of the papers surveyed used PET or MRI scans.

2.1.1. Chest X-rays

An X-ray is a diagnostic test that helps clinicians identify and treat medical problems [17]. The most widely performed medical X-ray procedure is a chest X-ray, and a chest X-ray produces images of the blood vessels, lungs, airways, heart and spine and chest bones. Traditionally, medical X-ray images were exposed to photographic films, which require processing before they can be viewed. To overcome this problem, digital X-rays are used [18]. Figure 4 shows several examples of chest X-ray with different lung conditions taken from various datasets.

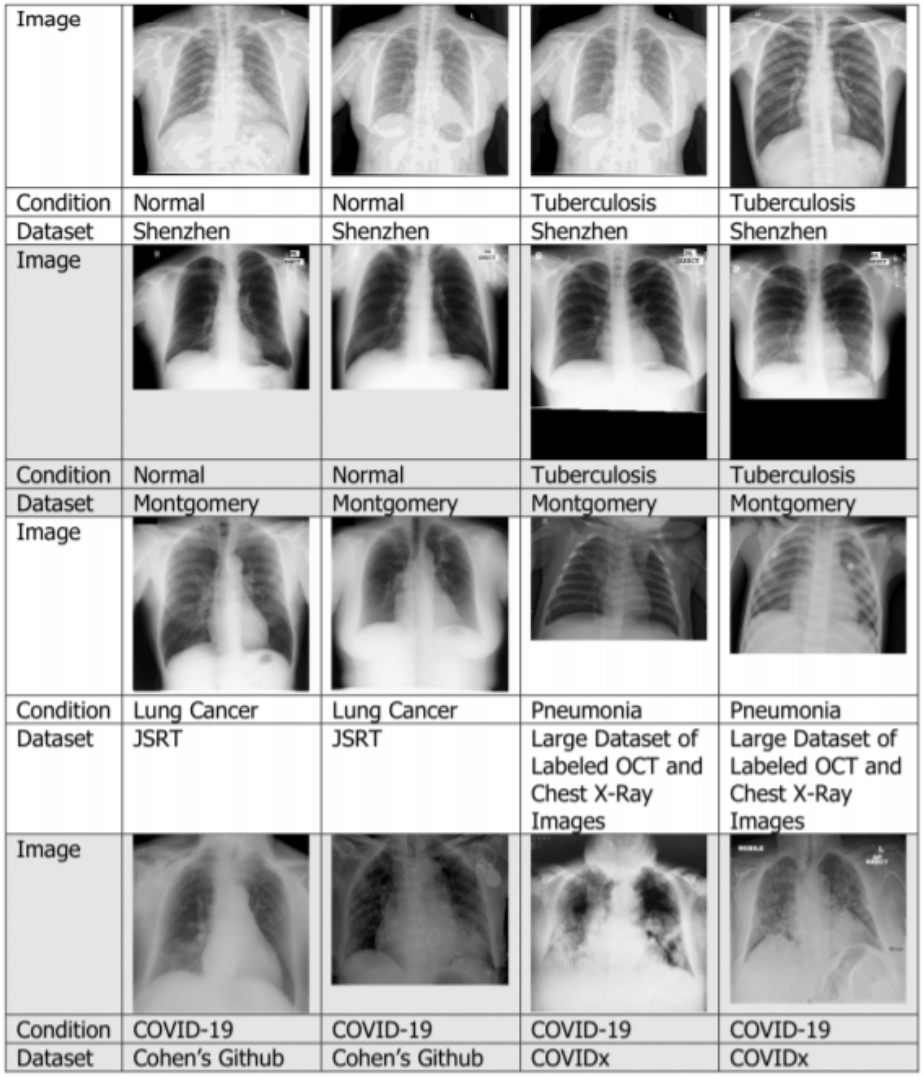


Figure 4. Examples of chest X-ray images.

Among the papers surveyed, the majority of them used chest X-rays. For example, X-rays were used for tuberculosis detection [19], pneumonia detection [20], lung cancer detection [14] and COVID-19 detection [21].

2.1.2. Sputum Smear Microscopy Images

Sputum is a dense fluid formed in the lungs and airways leading to the lungs. To perform sputum smear examination, a very thin layer of the sputum sample is positioned on a glass slide [27]. Among the papers surveyed, only five used sputum smear microscopy image [28–32]. Figure 6 shows examples of sputum smear microscopy images.

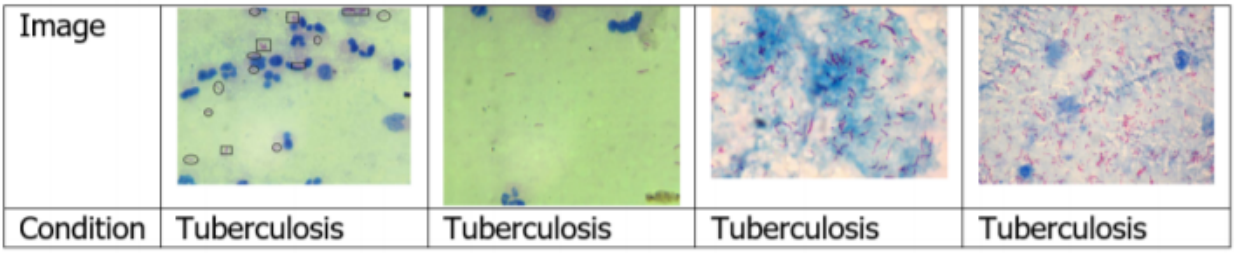


Figure 6. Examples of sputum smear microscopy images.

2.1.3. Histopathology Images

Histopathology is the study of the symptoms of a disease through microscopic examination of a biopsy or surgical specimen using glass slides. The sections are dyed with one or more stains to visualise the different components of the tissue [33]. Figure 7 shows a few examples of histopathology images. Among all the papers surveyed, only Coudray et al. [34] used histopathology images.

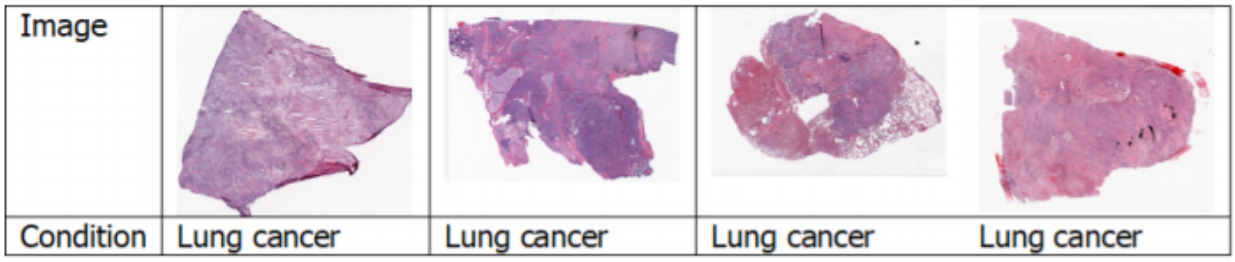


Figure 7. Examples of histopathology images.

2.2. Features

In computer vision, features are significant information extracted from images in terms of numerical values that could be used to solve specific problem [35]. Features might be in the form of specific structures in the image such as points, edges, colour, sizes, shapes or objects. Logically, the types of images affect the quality of the features. Feature transformation is a process that creates new features using the existing features. These new features may not have the same representation as to the original features, but they may have more discriminatory power in a different space than the original space. The purpose of feature transformation is to provide a

more useful feature for the machine learning algorithm for object identification. The features used in the surveyed papers include: Gabor, GIST, Local binary patterns (LBP), Tamura texture descriptor, colour and edge direction descriptor (CEDD) [36], Hu moments, colour layout descriptor (CLD) edge histogram descriptor (EHD) [37], primitive length, edge frequency, autocorrelation, shape features, size, orientation, bounding box, eccentricity, extent, centroid, scale-invariant feature transform (SIFT), regional properties area and speeded up robust features (SURF) [38]. Other feature representations in terms of histograms include pyramid histogram of oriented gradients (PHOG), histogram of oriented gradients (HOG) [39], intensity histograms (IH), shape descriptor histograms (SD), gradient magnitude histograms (GM), curvature descriptor histograms (CD) and fuzzy colour and texture histogram (FCTH). Some studies even performed lung segmentations before training their models (e.g., [13,14,36]). From the literature, a majority of the works surveyed used features that are automatically extracted from CNN. CNN can automatically learn and extract features, discarding the need for manual feature generation [40].

2.3. Data Augmentation

In deep learning, it is very important to have a large training dataset, as the community agrees that having more images can help improve training accuracy. Even a weak algorithm with a large amount of data can be more accurate than a strong algorithm with a modest amount of data [41]. Another obstacle is imbalanced classes. When doing binary classification training, if the number of samples of one class is a lot higher than the other class, the resulting model would be biased [6]. Deep learning algorithms perform optimally when the amount of samples in each class is equal or balanced. One way to increase the training dataset without obtaining new images is to use image augmentation. Image augmentation creates variations of the original images. This is achieved by performing different methods of processing, such as rotations, flips, translations, zooms and adding noise [42]. Figure 8 shows various examples of images after image augmentation. Data augmentation can also help increase the amount of relevant data in the dataset. For example, consider a car dataset with two labels, X and Y. One subset of the dataset contains images of cars of label X, but all the cars are facing left. The other subset contains images of cars of label Y, but all the cars are facing right. After training, a test image of a label Y car facing left is fed into the model, and the model labels that the car as X. The prediction is

wrong as the neural network search for the most obvious features that distinguish one class from another. To prevent this, a simple solution is to flip the images in the existing dataset horizontally such that they face the other side. Through augmentation, we may introduce relevant features and patterns, essentially boosting overall performance.

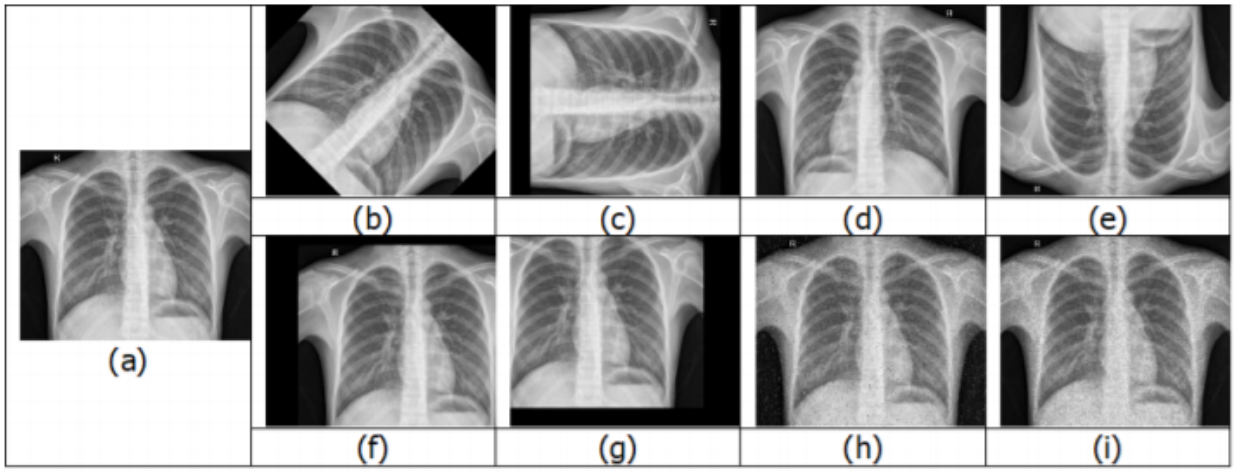


Figure 8. Examples of image augmentation: (**a**) original; (**b**) 45° rotation; (**c**) 90° rotation; (**d**) horizontal flip; (**e**) vertical flip; (**f**) positive x and y translation; (**g**) negative x and y translation; (**h**) salt and pepper noise; and (**i**) speckle noise.

Data augmentation also helps prevent overfitting. Overfitting refers to a case where a network learns a very high variance function, such as the perfect modelling of training results. Data augmentation addresses the issue of overfitting by introducing the model with more diverse data [43]. This diversity in data reduces variance and improves the generalisation of the model. However, data augmentation cannot overcome all biases present in a small dataset [43]. Other disadvantages of data augmentation include additional training time, transformation computing costs and additional memory costs.

Code

In [1]:
```
import numpy as np
import pandas as pd
import os
from glob import glob
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

Data visualizations

In [2]:
```
df = pd.read_csv('kaggle/input/data/Data_Entry_2017.csv')

df.head()
```

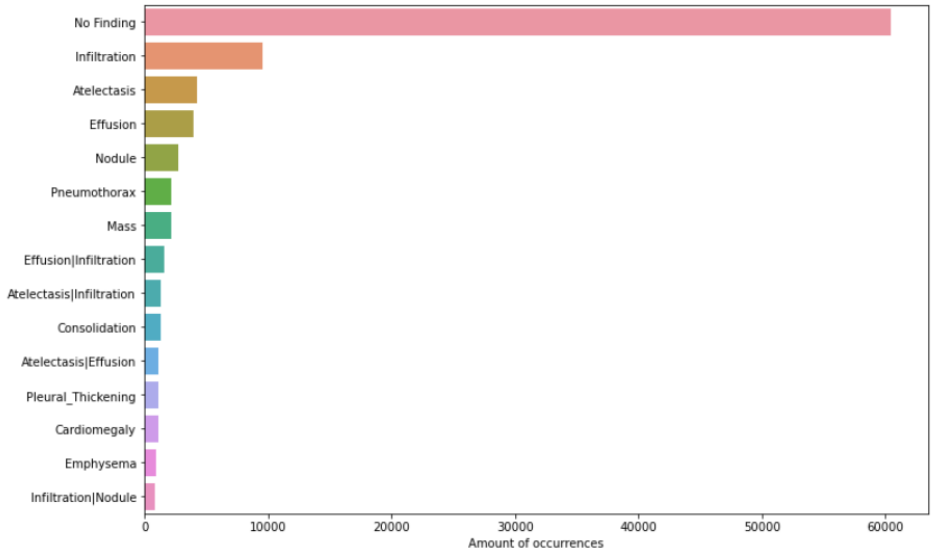
Out[2]:

	Image Index	Finding Labels	Follow-up #	Patient ID	Patient Age	Patient Gender	View Position	OriginalImage[Width	Height]	OriginalImagePixelSpacing[x	y]	Unr
0	00000001_000.png	Cardiomegaly	0	1	058Y	M	PA	2682	2749	0.143	0.143	
1	00000001_001.png	Cardiomegaly|Emphysema	1	1	058Y	M	PA	2894	2729	0.143	0.143	
2	00000001_002.png	Cardiomegaly|Effusion	2	1	058Y	M	PA	2500	2048	0.168	0.168	
3	00000002_000.png	No Finding	0	2	081Y	M	PA	2500	2048	0.171	0.171	
4	00000003_000.png	Hernia	0	3	081Y	F	PA	2582	2991	0.143	0.143	

In [3]:
```
labels_count = df['Finding Labels'].value_counts()[:15]
plt.subplots(figsize=(12, 8))
ax = sns.barplot(labels_count, labels_count.index)
ax.set(xlabel='Amount of occurrences')
```

```
C:\Users\91805\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword ar
gs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[3]: [Text(0.5, 0, 'Amount of occurrences')]



Small label preparation

In [4]:
```
df['Finding Labels'] = df['Finding Labels'].map(lambda x: x.replace('No Finding', ''))
y = []
for x in df['Finding Labels'].unique():
    splitted = x.split('|')
    y = np.append(y, splitted)

y = [x for x in y if len(x) > 0]
y = np.unique(y)
print('All labels ({}): {}'.format(len(y), y))
```

```
All labels (14): ['Atelectasis' 'Cardiomegaly' 'Consolidation' 'Edema' 'Effusion'
 'Emphysema' 'Fibrosis' 'Hernia' 'Infiltration' 'Mass' 'Nodule'
 'Pleural_Thickening' 'Pneumonia' 'Pneumothorax']
```

In [5]:
```
for label in y:
    df[label] = df['Finding Labels'].map(lambda x: 1.0 if label in x else 0.0)

df.head()
```

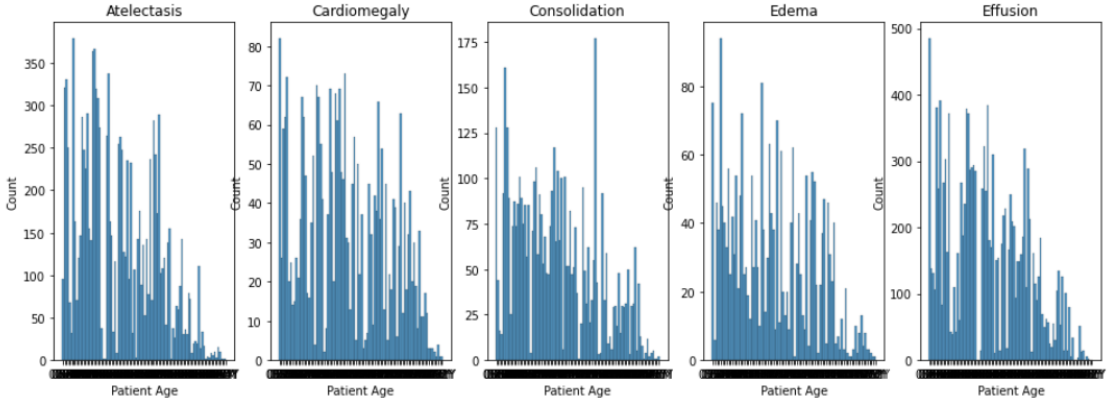
Out[5]:

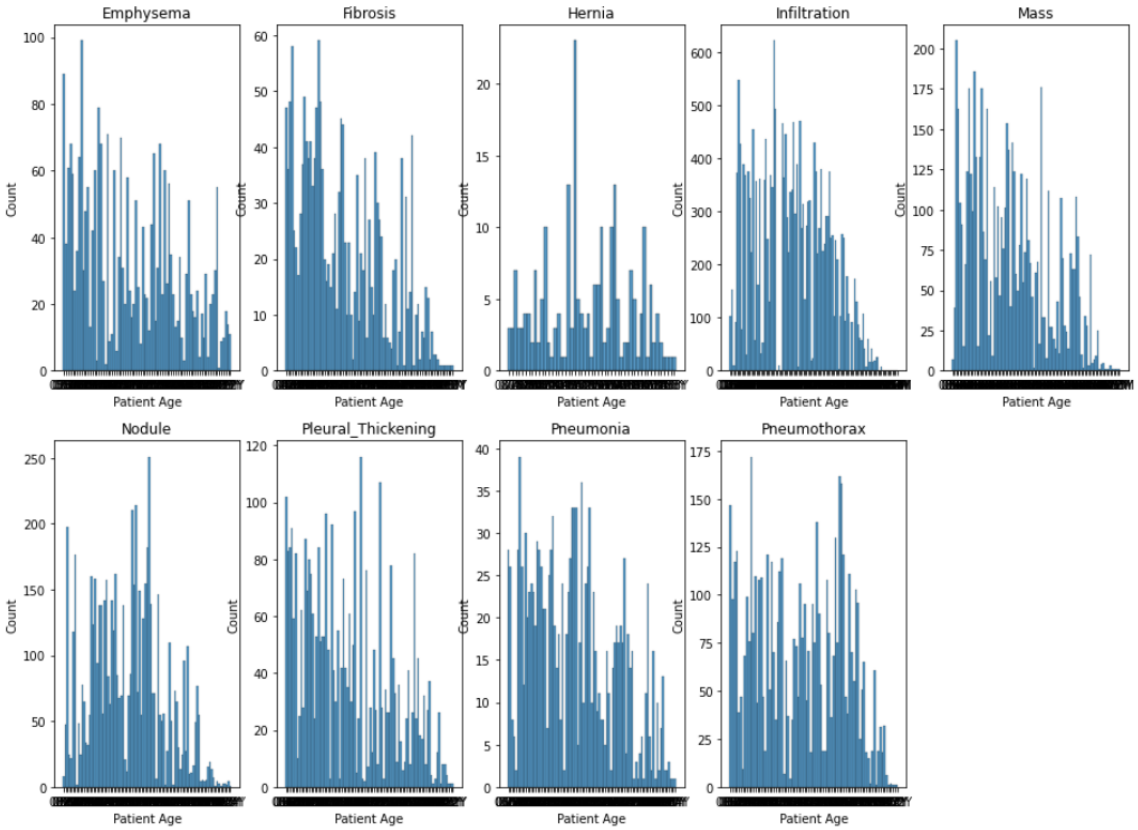
	Image Index	Finding Labels	Follow-up #	Patient ID	Patient Age	Patient Gender	View Position	OriginalImage[Width	Height]	OriginalImagePixelSpacing[x	...	Effusio
0	00000001_000.png	Cardiomegaly	0	1	058Y	M	PA	2682	2749	0.143	...	0.
1	00000001_001.png	Cardiomegaly|Emphysema	1	1	058Y	M	PA	2894	2729	0.143	...	0.
2	00000001_002.png	Cardiomegaly|Effusion	2	1	058Y	M	PA	2500	2048	0.168	...	1.
3	00000002_000.png		0	2	081Y	M	PA	2500	2048	0.171	...	0.
4	00000003_000.png	Hernia	0	3	081Y	F	PA	2582	2991	0.143	...	0.

5 rows × 26 columns

More data visualizations

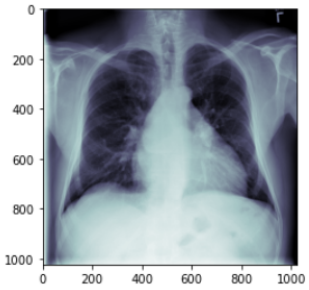
In [6]:
```
plt.figure(figsize=(16, 18))
for i, column in enumerate(df[y]):
    plt.subplot(3, 5, i+1)
    plt.title(column)
    sns.histplot(df[df[column].where(df[column] == 1.0).notnull()]['Patient Age'])
```





Display example images

In [7]:
```
import matplotlib.image as mpimg

img = mpimg.imread('kaggle/input/data/images_001/images/00000001_000.png')
imgplot = plt.imshow(img, cmap='bone')
plt.show()
```



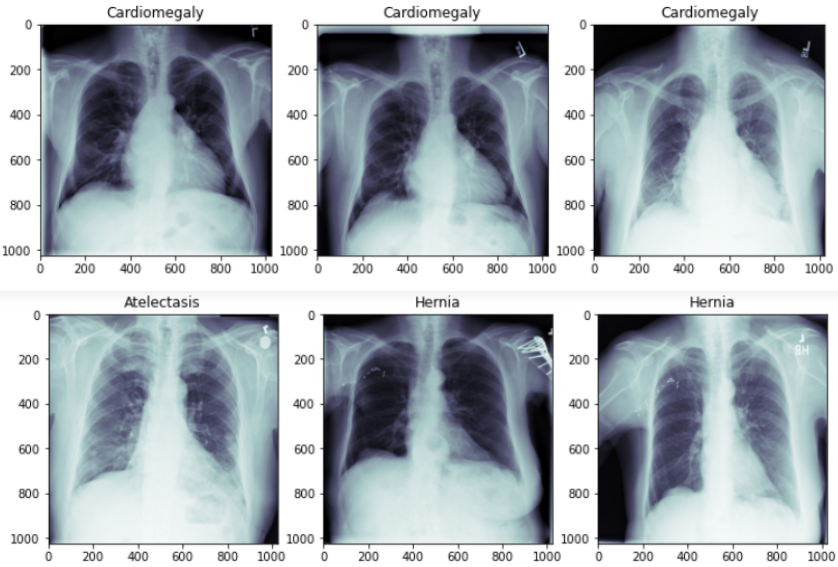
In [8]:
```
def get_corresponding_label(image_index):
    return y[df.loc[df['Image Index'] == image_index][y].values.argmax()]

get_corresponding_label('00000001_000.png')
```

Out[8]: 'Cardiomegaly'

In [8]:
```
def get_corresponding_label(image_index):
    return y[df.loc[df['Image Index'] == image_index][y].values.argmax()]

get_corresponding_label('00000001_000.png')
```

Out[8]: 'Cardiomegaly'

In [9]:
```
def show_images(path='kaggle/input/data', extension='/images_001/images', labeled=False, max_images=6):
    amount = 0
    fig = plt.figure(figsize=(12, 8))

    for file in os.listdir(path + extension):
        if file.endswith('.png'):
            if amount == max_images:
                break

            img = mpimg.imread(os.path.join(path + extension, file))
            plt.subplot(231+amount)
            if labeled:
                plt.title(get_corresponding_label(file))
            imgplot = plt.imshow(img, cmap='bone')

            amount += 1

show_images(labeled=True)
```



Prepare the training data

This will store every result of the given data in one single vector. Basically, we store what the model will try to predict.

In [10]:
```
#df['disease_vec'] = df.apply(lambda x: [x[y].values], 1).map(lambda x: x[0])
#df['disease_vec'].head()
```

In [11]:
```
df.shape
```

Out[11]: (112120, 26)

In [16]:
```
df = df[df['Patient Age'] < '100Y']
df_image_paths = {os.path.basename(x): x for x in
                  glob(os.path.join('kaggle/input/data', 'images*', '*', '*.png'))}
print('Scans found:', len(df_image_paths), ', Total Headers', df.shape[0])
df['path'] = df['Image Index'].map(df_image_paths.get)
df['Patient Age'] = df['Patient Age'].map(lambda x: int(x[:-1]))
```

```
Scans found: 112120 , Total Headers 112104
```

In [17]:
```
new_df = df.sample(30000)
```

In [18]:
```
import cv2
def read_img(img_path):
    img = cv2.imread(img_path)
    img = cv2.resize(img, (128, 128))
    return img

from tqdm import tqdm
train_img = []
for img_path in tqdm(new_df['path'].values):
    train_img.append(read_img(img_path))
```

```
100%|██████████| 30000/30000 [46:13<00:00, 10.82it/s]
```

In [19]:
```
X = np.array(train_img, np.float32)/255
```

In [20]:
```
Y = new_df[y].values
```

In [21]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(
    X,
    Y,
    test_size=0.25,
    random_state=2018
)
print('train', X_train.shape[0], 'val', X_test.shape[0])
```

```
train 22500 val 7500
```

In [22]:
```
del X
del Y
```

In [23]:
```
X_train.shape
```

Out[23]:
```
(22500, 128, 128, 3)
```

In [24]:
```
Y_train.shape
```

Out[24]:
```
(22500, 14)
```

In [28]:
```
from keras.applications.mobilenet import MobileNet
from keras.layers import GlobalAveragePooling2D, Dense, Dropout, Flatten
from tensorflow.keras.models import Sequential

base_mobilenet_model = MobileNet(
    input_shape=(128, 128, 3),
    include_top=False,
    weights=None
)
model = Sequential()
model.add(base_mobilenet_model)
model.add(GlobalAveragePooling2D())
model.add(Dropout(0.5))
model.add(Dense(512))
model.add(Dropout(0.5))
model.add(Dense(len(y), activation='sigmoid'))

model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['binary_accuracy', 'mae']
)

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenet_1.00_128 (Function (None, 4, 4, 1024)        3228864
_________________________________________________________________
global_average_pooling2d (Gl (None, 1024)              0
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 512)               524800
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 14)                7182
=================================================================
Total params: 3,760,846
Trainable params: 3,738,958
Non-trainable params: 21,888
_________________________________________________________________
```

In [31]:
```
from keras.callbacks import ModelCheckpoint, LearningRateScheduler, EarlyStopping, ReduceLROnPlateau

weight_path="{}_weights.best.hdf5".format('xray_class')

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

early = EarlyStopping(
    monitor="val_loss",
    mode="min",
    patience=3
)

callbacks_list = [checkpoint, early]
```

In [33]:
```
early_stops = EarlyStopping(patience=3, monitor='val_accuracy')

model.fit(
    x=X_train,
    y=Y_train,
    batch_size=100,
    epochs=5,
    validation_split=0.3,
    callbacks=[early_stops]
)
```

```
Epoch 1/5
158/158 [==============================] - 383s 2s/step - loss: 0.1818 - binary_accuracy: 0.9483 - mae: 0.0927 - val_loss: 0.23
56 - val_binary_accuracy: 0.9489 - val_mae: 0.0593
WARNING:tensorflow:Early stopping conditioned on metric `val_accuracy` which is not available. Available metrics are: loss,bina
ry_accuracy,mae,val_loss,val_binary_accuracy,val_mae
Epoch 2/5
158/158 [==============================] - 386s 2s/step - loss: 0.1788 - binary_accuracy: 0.9484 - mae: 0.0918 - val_loss: 0.19
86 - val_binary_accuracy: 0.9489 - val_mae: 0.0704
WARNING:tensorflow:Early stopping conditioned on metric `val_accuracy` which is not available. Available metrics are: loss,bina
ry_accuracy,mae,val_loss,val_binary_accuracy,val_mae
Epoch 3/5
158/158 [==============================] - 383s 2s/step - loss: 0.1765 - binary_accuracy: 0.9484 - mae: 0.0913 - val_loss: 0.18
65 - val_binary_accuracy: 0.9489 - val_mae: 0.0755
WARNING:tensorflow:Early stopping conditioned on metric `val_accuracy` which is not available. Available metrics are: loss,bina
ry_accuracy,mae,val_loss,val_binary_accuracy,val_mae
Epoch 4/5
158/158 [==============================] - 383s 2s/step - loss: 0.1738 - binary_accuracy: 0.9485 - mae: 0.0903 - val_loss: 0.18
97 - val_binary_accuracy: 0.9468 - val_mae: 0.1204
WARNING:tensorflow:Early stopping conditioned on metric `val_accuracy` which is not available. Available metrics are: loss,bina
ry_accuracy,mae,val_loss,val_binary_accuracy,val_mae
Epoch 5/5
158/158 [==============================] - 353s 2s/step - loss: 0.1727 - binary_accuracy: 0.9485 - mae: 0.0900 - val_loss: 0.18
42 - val_binary_accuracy: 0.9479 - val_mae: 0.0831
WARNING:tensorflow:Early stopping conditioned on metric `val_accuracy` which is not available. Available metrics are: loss,bina
ry_accuracy,mae,val_loss,val_binary_accuracy,val_mae
```

Out[33]: <tensorflow.python.keras.callbacks.History at 0x23ee08a8c10>

In [34]:
```
for c_label, s_count in zip(y, 100*np.mean(Y_test, 0)):
    print('%s: %2.2f%%' % (c_label, s_count))
```

In [34]:
```
for c_label, s_count in zip(y, 100*np.mean(Y_test, 0)):
    print('%s: %2.2f%%' % (c_label, s_count))
```

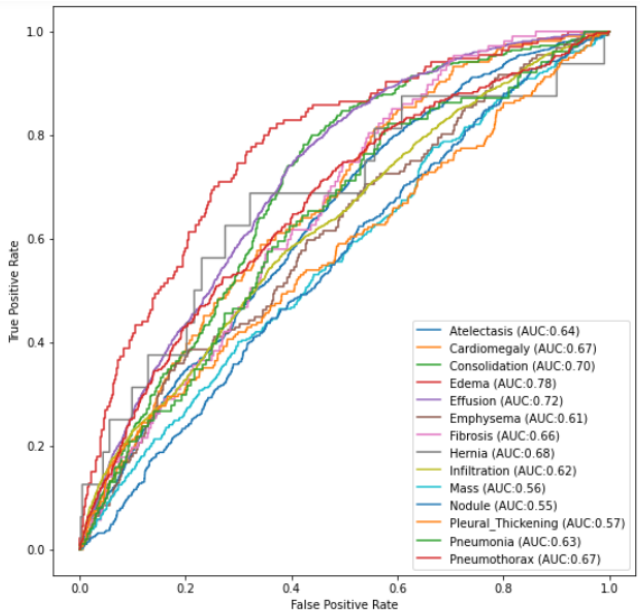
```
Atelectasis: 10.53%
Cardiomegaly: 2.75%
Consolidation: 4.37%
Edema: 1.79%
Effusion: 12.71%
Emphysema: 2.08%
Fibrosis: 1.43%
Hernia: 0.21%
Infiltration: 17.57%
Mass: 4.83%
Nodule: 5.56%
Pleural_Thickening: 2.89%
Pneumonia: 1.35%
Pneumothorax: 4.80%
```

In [35]:
```
pred_Y = model.predict(X_test, batch_size=32, verbose=True)
```

```
235/235 [==============================] - 26s 109ms/step
```

ROC Curves

In [36]:
```
from sklearn.metrics import roc_curve, auc

fig, c_ax = plt.subplots(1, 1, figsize = (9, 9))

for (idx, c_label) in enumerate(y):
    fpr, tpr, thresholds = roc_curve(Y_test[:, idx].astype(int), pred_Y[:, idx])
    c_ax.plot(fpr, tpr, label = '%s (AUC:%0.2f)'  % (c_label, auc(fpr, tpr)))
c_ax.legend()
c_ax.set_xlabel('False Positive Rate')
c_ax.set_ylabel('True Positive Rate')
fig.savefig('barely_trained_net.png')
```



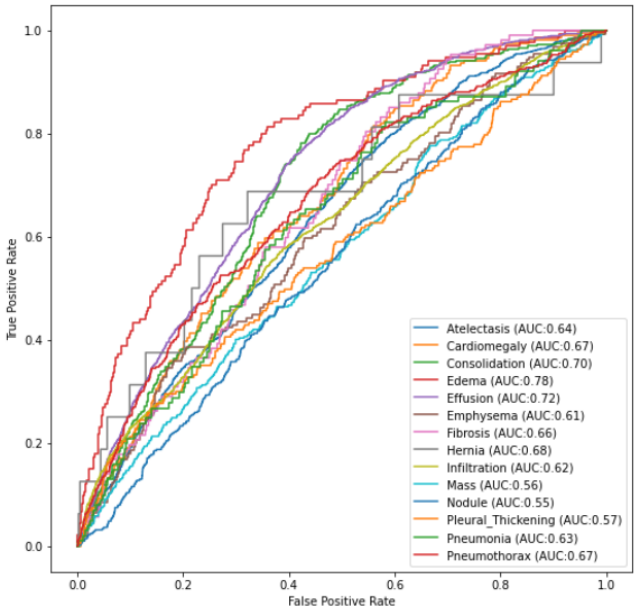
Continue training

In [37]:
```
early_stops = EarlyStopping(patience=3, monitor='val_acc')

model.fit(x=X_train, y=Y_train, batch_size=100, epochs=8, validation_split=0.3, callbacks=[early_stops])
```

```
Epoch 1/8
158/158 [==============================] - 269s 2s/step - loss: 0.1707 - binary_accuracy: 0.9484 - mae: 0.0896 - val_loss: 0.17
61 - val_binary_accuracy: 0.9483 - val_mae: 0.0854
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,binary_ac
curacy,mae,val_loss,val_binary_accuracy,val_mae
Epoch 2/8
158/158 [==============================] - 269s 2s/step - loss: 0.1685 - binary_accuracy: 0.9485 - mae: 0.0884 - val_loss: 0.17
85 - val_binary_accuracy: 0.9482 - val_mae: 0.0947
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,binary_ac
curacy,mae,val_loss,val_binary_accuracy,val_mae
Epoch 3/8
158/158 [==============================] - 270s 2s/step - loss: 0.1666 - binary_accuracy: 0.9486 - mae: 0.0882 - val_loss: 0.17
91 - val_binary_accuracy: 0.9461 - val_mae: 0.1006
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,binary_ac
curacy,mae,val_loss,val_binary_accuracy,val_mae
Epoch 4/8
158/158 [==============================] - 269s 2s/step - loss: 0.1652 - binary_accuracy: 0.9485 - mae: 0.0877 - val_loss: 0.19
90 - val_binary_accuracy: 0.9479 - val_mae: 0.0687
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,binary_ac
curacy,mae,val_loss,val_binary_accuracy,val_mae
Epoch 5/8
158/158 [==============================] - 270s 2s/step - loss: 0.1642 - binary_accuracy: 0.9487 - mae: 0.0870 - val_loss: 0.18
05 - val_binary_accuracy: 0.9488 - val_mae: 0.0905
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,binary_ac
curacy,mae,val_loss,val_binary_accuracy,val_mae
Epoch 6/8
158/158 [==============================] - 269s 2s/step - loss: 0.1614 - binary_accuracy: 0.9488 - mae: 0.0865 - val_loss: 0.19
17 - val_binary_accuracy: 0.9482 - val_mae: 0.0762
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,binary_ac
curacy,mae,val_loss,val_binary_accuracy,val_mae
Epoch 7/8
158/158 [==============================] - 270s 2s/step - loss: 0.1579 - binary_accuracy: 0.9489 - mae: 0.0848 - val_loss: 0.19
05 - val_binary_accuracy: 0.9461 - val_mae: 0.0840
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,binary_ac
curacy,mae,val_loss,val_binary_accuracy,val_mae
Epoch 8/8
158/158 [==============================] - 270s 2s/step - loss: 0.1542 - binary_accuracy: 0.9492 - mae: 0.0835 - val_loss: 0.19
74 - val_binary_accuracy: 0.9481 - val_mae: 0.0770
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,binary_ac
curacy,mae,val_loss,val_binary_accuracy,val_mae
```

Out[37]: <tensorflow.python.keras.callbacks.History at 0x23ee0c83700>

In [38]:
```
from sklearn.metrics import roc_curve, auc
fig, c_ax = plt.subplots(1, 1, figsize = (9, 9))
for (idx, c_label) in enumerate(y):
    fpr, tpr, thresholds = roc_curve(Y_test[:, idx].astype(int), pred_Y[:, idx])
    c_ax.plot(fpr, tpr, label = '%s (AUC:%0.2f)'  % (c_label, auc(fpr, tpr)))
c_ax.legend()
c_ax.set_xlabel('False Positive Rate')
c_ax.set_ylabel('True Positive Rate')
fig.savefig('barely_trained_net.png')
```

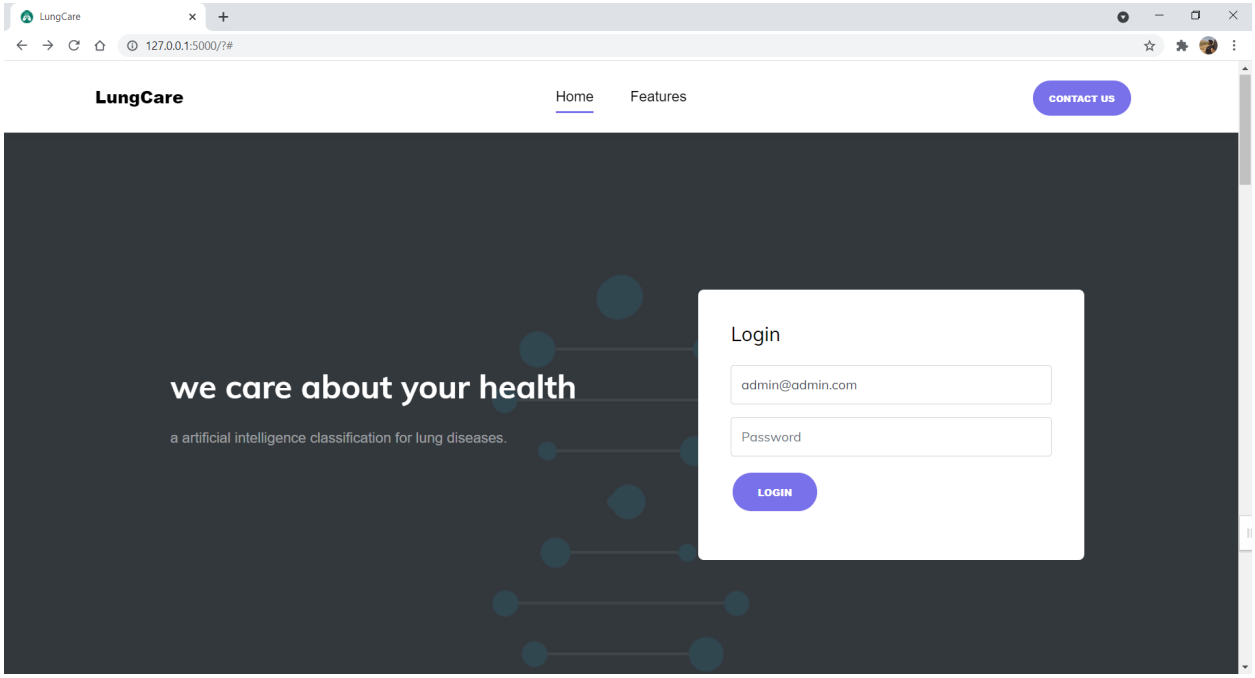


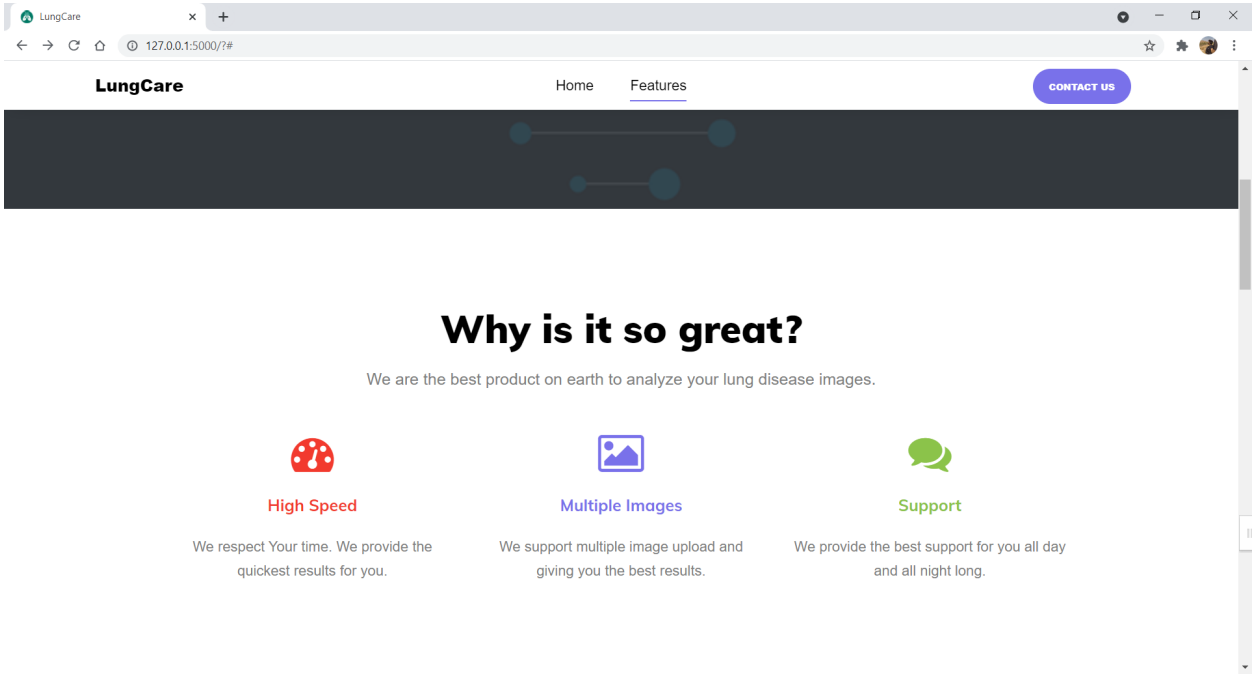
Save the model to disk

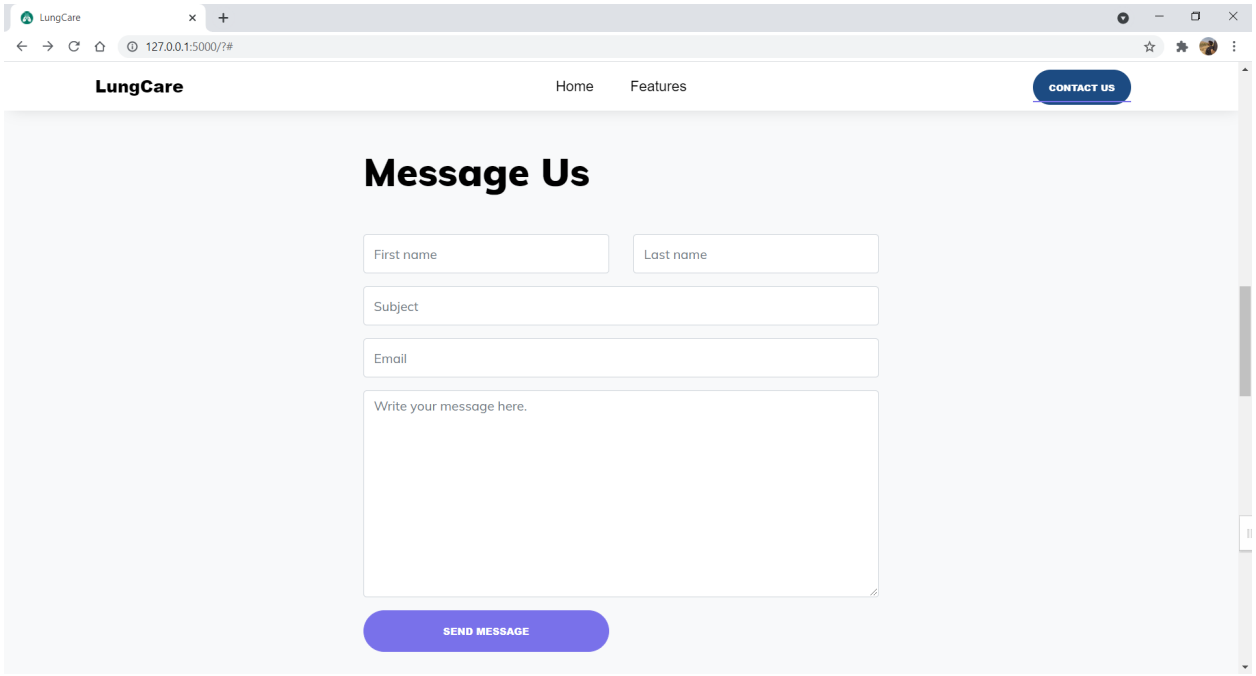
In [39]:
```
model_json = model.to_json()
with open("multi_disease_model.json", "w") as json_file:
    json_file.write(model_json)

# serialize weights to HDF5
model.save_weights("multi_disease_model_weight.h5")
print("Saved model to disk")
```

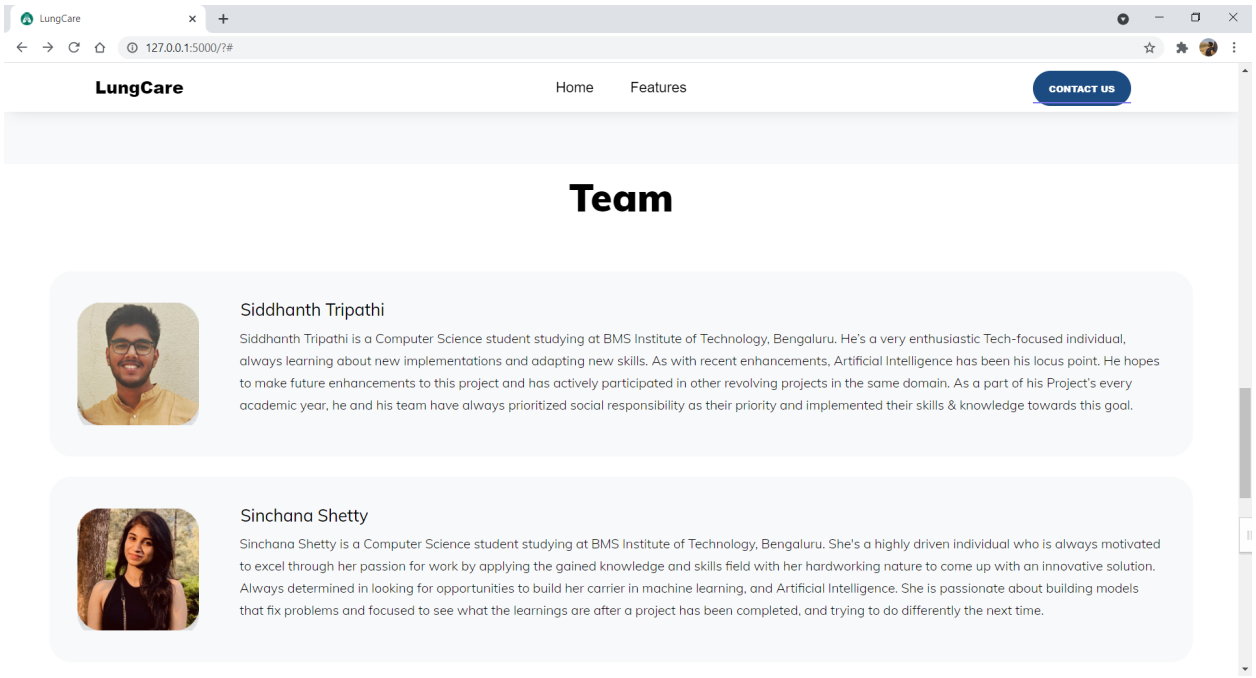
```
Saved model to disk
```

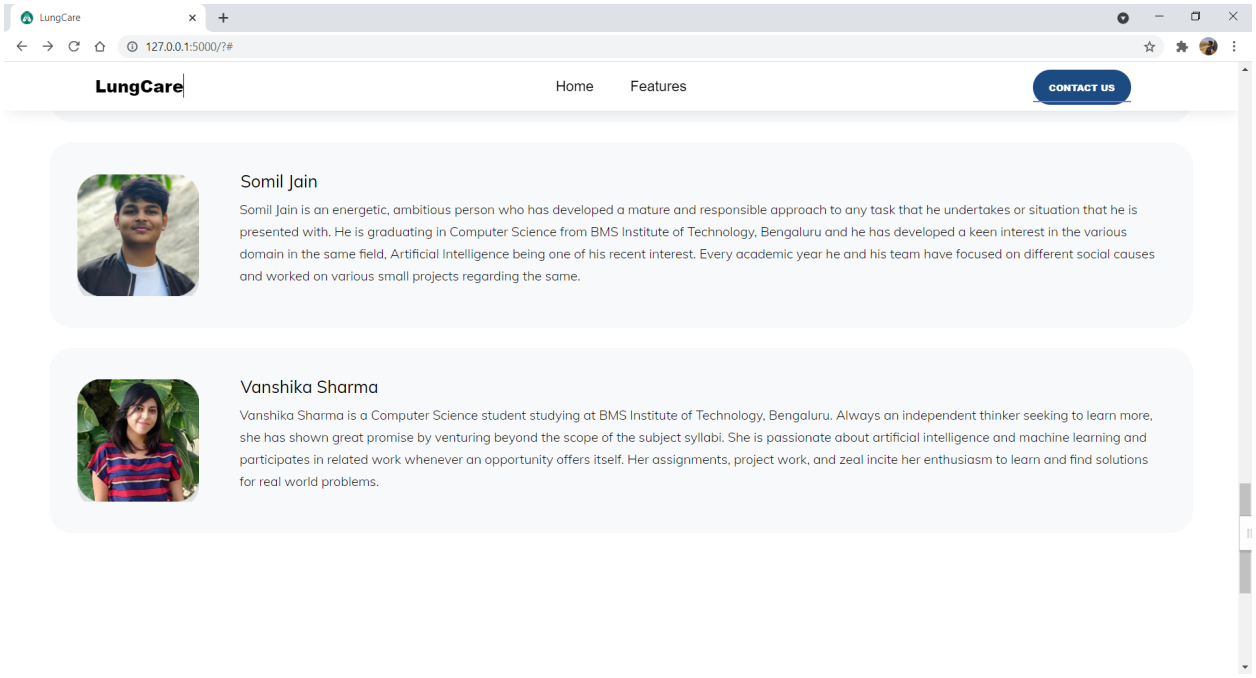
Screenshot

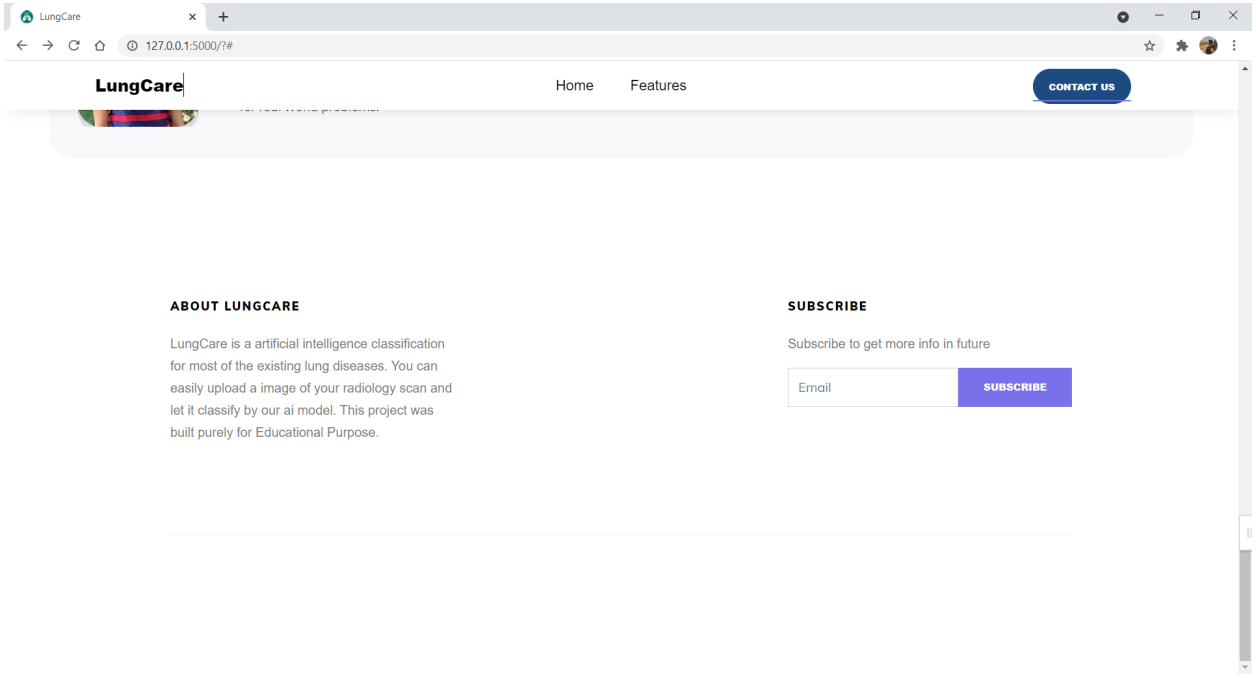

LungCare
127.0.0.1:5000/?#
LungCare
Home
Features
CONTACT US
we care about your health
a artificial intelligence classification for lung diseases.
Login
admin@admin.com
Password
LOGIN

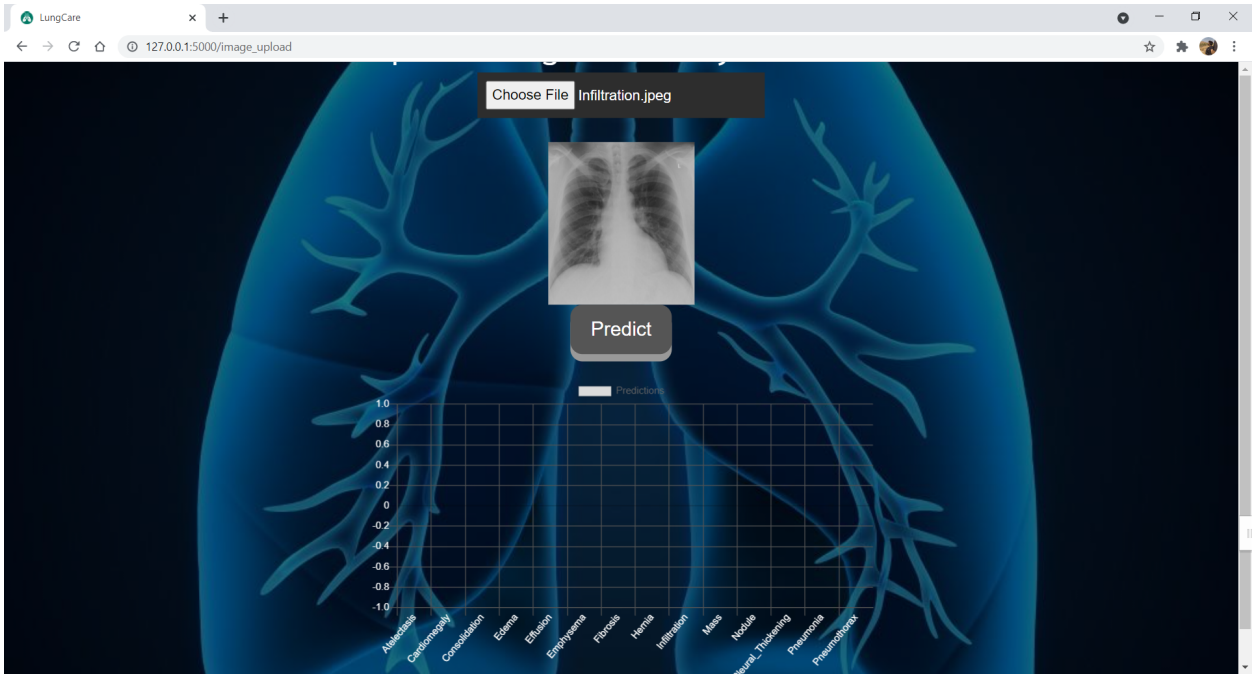

LungCare
127.0.0.1:5000/?#
LungCare
Home
Features
CONTACT US
Why is it so great?
We are the best product on earth to analyze your lung disease images.
High Speed
We respect Your time. We provide the quickest results for you.
Multiple Images
We support multiple image upload and giving you the best results.
Support
We provide the best support for you all day and all night long.

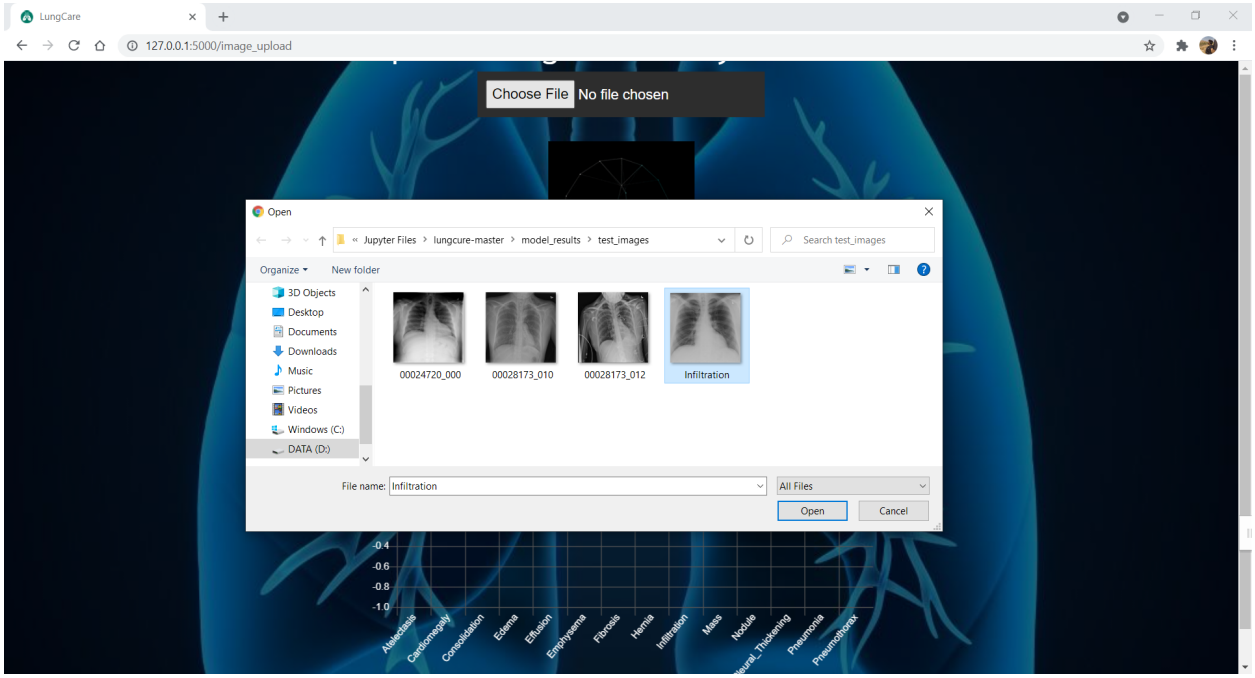

LungCare
127.0.0.1:5000/?#
LungCare
Home
Features
CONTACT US
Message Us
First name
Last name
Subject
Email
Write your message here.
SEND MESSAGE

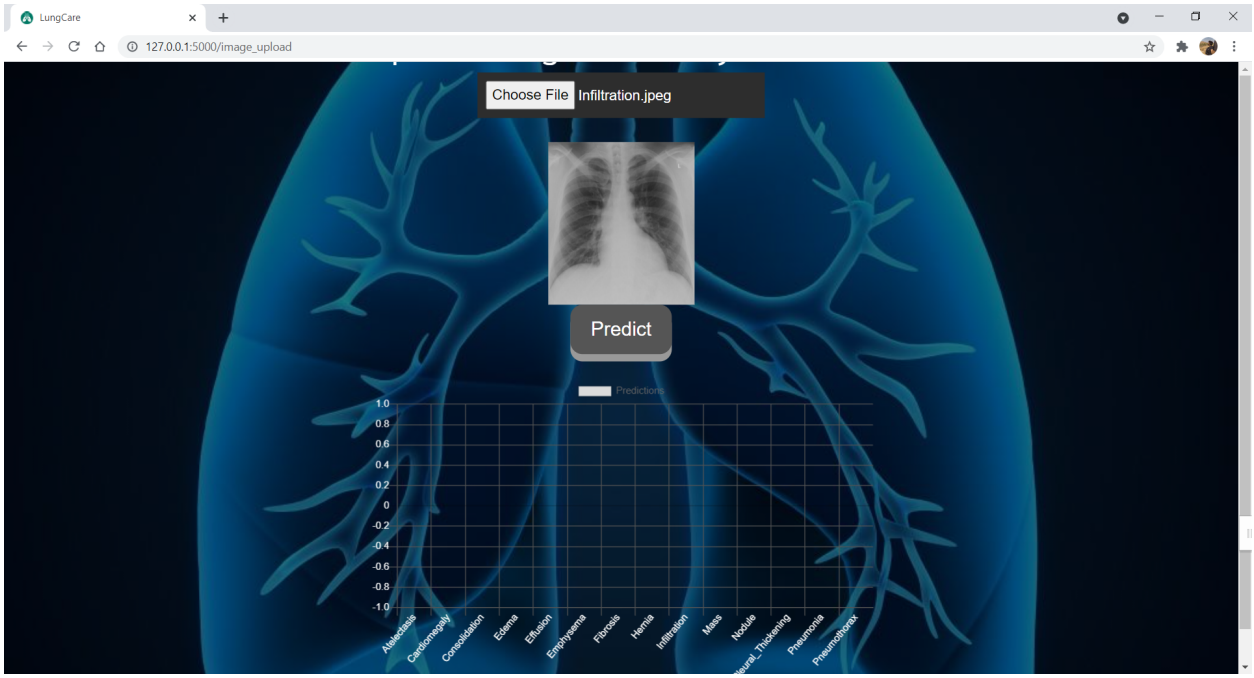

LungCare
127.0.0.1:5000/?#
LungCare
Home
Features
CONTACT US
Team
Siddhanth Tripathi
Siddhanth Tripathi is a Computer Science student studying at BMS Institute of Technology, Bengaluru. He's a very enthusiastic Tech-focused individual, always learning about new implementations and adapting new skills. As with recent enhancements, Artificial Intelligence has been his locus point. He hopes to make future enhancements to this project and has actively participated in other revolving projects in the same domain. As a part of his Project's every academic year, he and his team have always prioritized social responsibility as their priority and implemented their skills & knowledge towards this goal.
Sinchana Shetty
Sinchana Shetty is a Computer Science student studying at BMS Institute of Technology, Bengaluru. She's a highly driven individual who is always motivated to excel through her passion for work by applying the gained knowledge and skills field with her hardworking nature to come up with an innovative solution. Always determined in looking for opportunities to build her carrier in machine learning, and Artificial Intelligence. She is passionate about building models that fix problems and focused to see what the learnings are after a project has been completed, and trying to do differently the next time.

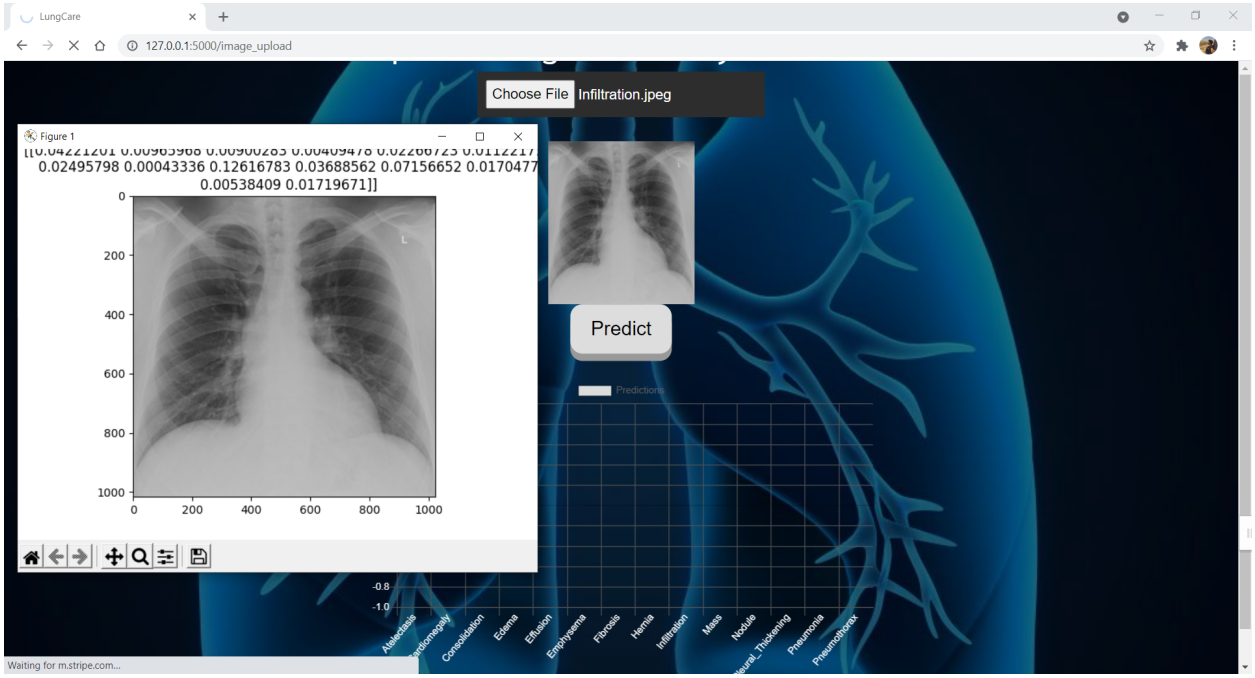

LungCare
127.0.0.1:5000/?#
LungCare
Home
Features
CONTACT US
Somil Jain
Somil Jain is an energetic, ambitious person who has developed a mature and responsible approach to any task that he undertakes or situation that he is presented with. He is graduating in Computer Science from BMS Institute of Technology, Bengaluru and he has developed a keen interest in the various domain in the same field, Artificial Intelligence being one of his recent interest. Every academic year he and his team have focused on different social causes and worked on various small projects regarding the same.
Vanshika Sharma
Vanshika Sharma is a Computer Science student studying at BMS Institute of Technology, Bengaluru. Always an independent thinker seeking to learn more, she has shown great promise by venturing beyond the scope of the subject syllabi. She is passionate about artificial intelligence and machine learning and participates in related work whenever an opportunity offers itself. Her assignments, project work, and zeal incite her enthusiasm to learn and find solutions for real world problems.


LungCare
127.0.0.1:5000/?#
LungCare
Home
Features
CONTACT US
ABOUT LUNGCARE
LungCare is a artificial intelligence classification for most of the existing lung diseases. You can easily upload a image of your radiology scan and let it classify by our ai model. This project was built purely for Educational Purpose.
SUBSCRIBE
Subscribe to get more info in future
Email
SUBSCRIBE

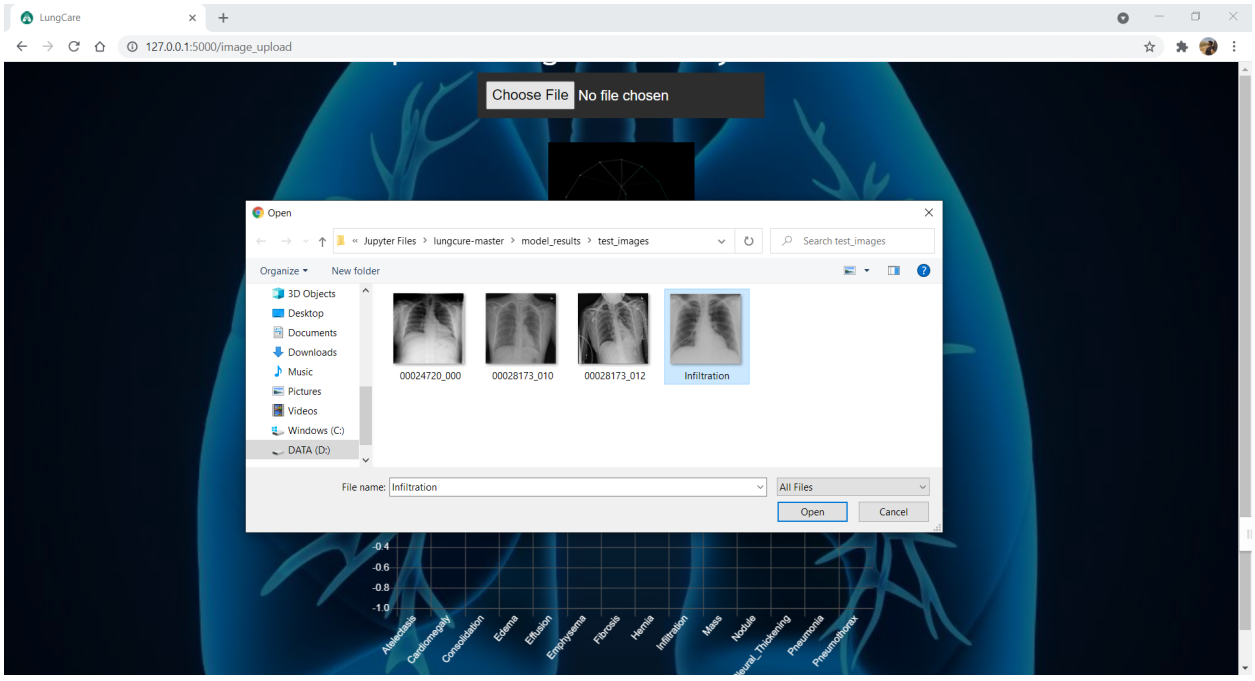

LungCare
127.0.0.1:5000/image_upload
Choose File
Infiltration.jpeg
Predict
Predictions
1.0
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1.0
Atelectasis
Cardiomegaly
Consolidation
Edema
Effusion
Emphysema
Fibrosis
Hernia
Infiltration
Mass
Nodule
Pneumonia
Pneumothorax


LungCare
127.0.0.1:5000/image_upload
Choose File
No file chosen
Open
Jupyter Files
lungcure-master
model_results
test_images
Search test_images
Organize
New folder
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
DATA (D:)
00024720_000
00028173_010
00028173_012
Infiltration
File name:
Infiltration
All Files
Open
Cancel


LungCare
127.0.0.1:5000/image_upload
Choose File
Infiltration.jpeg
Predict
Predictions
1.0
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1.0
Atelectasis
Cardiomegaly
Consolidation
Edema
Effusion
Emphysema
Fibrosis
Hernia
Infiltration
Mass
Nodule
Pneumonia
Pneumothorax


LungCare
127.0.0.1:5000/image_upload
Choose File
Infiltration.jpeg
Figure 1
0.02495798 0.00043336 0.12616783 0.03688562 0.07156652 0.0170477
0.00538409 0.01719671]]
0
200
400
600
800
1000
Predict
Predictions
Waiting for m.stripe.com...


LungCare
127.0.0.1:5000/predict
Choose File
No file chosen
Predict
Predictions
14
12
10
8
6
4
2
0
Atelectasis
Cardiomegaly
Consolidation
Edema
Effusion
Emphysema
Fibrosis
Hernia
Infiltration
Mass
Nodule
Pneumonia
Pneumothorax


LungCare
127.0.0.1:5000/image_upload
Choose File
No file chosen
Open
Jupyter Files
lungcure-master
model_results
test_images
Search test_images
Organize
New folder
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
DATA (D:)
00024720_000
00028173_010
00028173_012
Infiltration
File name:
Infiltration
All Files
Open
Cancel
-0.4
-0.6
-0.8
-1.0
Atelectasis
Cardiomegaly
Consolidation
Edema
Effusion
Emphysema
Fibrosis
Hernia
Infiltration
Mass
Nodule
Pneumonia
Pneumothorax

References

1. https://medicalxpress.com/news/2017-08-lung-diseases-million.html
2. Andrew Ward, Nicholas Bambos. Quantum Annealing Assisted Deep Learning for Lung Cancer Detection. http://cs231n.stanford.edu/reports/2017/pdfs/534.pdf
3. Albert Chon, Niranjan Balachandar. Deep Convolutional Neural Networks for Lung Cancer Detection. http://cs231n.stanford.edu/reports/2017/pdfs/518.pdf
4. Kingsley Kuan, Mathieu Ravaut, Gaurav Manek, Huiling Chen, Jie Lin, Babar Nazir, Cen Chen, Tse Chiang Howe, Zeng Zeng, Vijay Chandrasekhar. Deep Learning for Lung Cancer Detection: Tackling the Kaggle Data Science Bowl 2017 Challenge. https://arxiv.org/abs/1705.09435
5. https://www.nature.com/articles/srep46479
6. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5569872/
7. https://stephenson.pure.elsevier.com/en/publications/computer-aided-lung-cancer-diagnosiswith-deep-learning-algorithm
8. https://www.researchgate.net/publication/301651102_Computer_aided_lung_cancer_diagnosis_ with_deep_learning_algorithms
9. Matrix capsules with EM routing. https://openreview.net/forum?id=HJWLfGWRb¬eId=HJWLfGWRb 10. Sara Sabour, Nicholas Frosst, Geoffrey E Hinton. Dynamic Routing Between Capsules. https://arxiv.org/abs/1710.09829
10. NIH sample Chest X-rays dataset, https://www.kaggle.com/nih-chest-xrays/sample
11. NIH full Chest X-rays dataset, https://www.kaggle.com/nih-chest-xrays/data
12. Max Jaderberg, Karen Simonyan, Andrew Zisserman, Koray Kavukcuoglu. Spatial Transformer Networks. https://arxiv.org/abs/1506.02025