Ain Shams University
Faculty of Computer & Information Sciences
Computer Science Department

Abnormality Detection and Localization in Chest X-Rays

July 2021

[Sponsor Logo if exists] [ITIDA Logo if exists]

Ain Shams University
Faculty of Computer & Information Sciences
Computer Science Department

Abnormality Detection and Localization in Chest X-Rays

By:

Mahmoud Khaled Abd Elfattah [CS]
Mahmoud Khaled Salem [CS]
Mahmoud El-Sayed Mohamed [CS]
Mahmoud Mohamed Hassan [CS]
Mohamed Adel Abd Elkhalek [CS]

Under Supervision of :

[**Dr. Ahmed Salah**]
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

[**TA. Aya Nasser**]
Scientific Computing Department
Faculty of Computer and Information Sciences,
Ain Shams University.

Acknowledgement

All graduate is to ALLAH for the strength and blessing in completing our graduation project. Profound appreciation to Dr. Ahmed Salah for his supervision starting from formulating the problem, constructive suggestions and fulfillment the project. we would like to express our sincere thanks and graduate without her continuous support to achieve the successful completion of this project. sincere thanks and appreciation to our families, we couldn’t complete our Project without their continuous support and help.

Abstract

Chest X-Rays (CXRs) are widely used for diagnosing abnormalities in the chest area. Automatically detecting these abnormalities with high accuracy could greatly enhance real world diagnosis processes. Lack of standard publicly available dataset and benchmark studies, however, makes it difficult to compare and establish the best detection methods. In order to overcome these difficulties, we have used the publicly available NIH chest X-Ray Dataset and studied the performance of known deep convolutional network (DCN) architectures on different abnormalities. We employed heat maps obtained from occlusion sensitivity as a measure of localization in the CXRs. We find that the same DCN architecture doesn't perform well across all abnormalities. Shallow features or earlier layers consistently provide higher detection accuracy compared to deep features. We have also found ensemble models to improve classification significantly compared to single model. Combining this insight, we report the highest accuracy on chest X-Ray abnormality detection on this dataset.

We applied the techniques developed along the way to the problem of tuberculosis detection on a different disease and achieved the highest accuracy on that task. Our localization experiments using these trained classifiers show that for spatially spread-out abnormalities like Effusion and Edema, the network can localize the abnormalities successfully most of the time. We believe that through deep learning-based classification and localization, we will discover many more interesting features in medical image diagnosis that are not considered traditionally.

We worked on NIH dataset (National Institutes of Health - Clinical Center) contain 112,120 chest x-ray images consist 14 –diseases each images have resolution 1024 * 1024 (Public dataset)

NIH Dataset Link

We worked on 6 classes (5 diseases and normal) and achieved Average AUC 92.5

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CPU	Central processing unit
DL	Deep learning
GPU	Graphics processing unit
NLP	Natural language processing
AUC	Area Under curve
CXR	Chest X-Ray
DCN	Deep convolutional Network

1- Introduction

1.1 Motivation

With millions of diagnostic examinations performed annually, chest X-rays are an important and accessible clinical imaging tool for the detection of many diseases. Advances in machine learning (ML) and deep learning present an exciting opportunity to create new tools to help low experienced radiologists to detect abnormality with higher accuracy.

1.2 Problem Definition

Faulty determination of X-ray may lead to wrong treatment.

1.3 Objective

1. Develop an automated model that can interpret the X-ray image and detect various diseases.
2. Localize region of detected abnormality findings in chest X-ray image.
3. Maximize the number of diseases and maintaining the high accuracy.

1.4 Time Plan Figure [1]

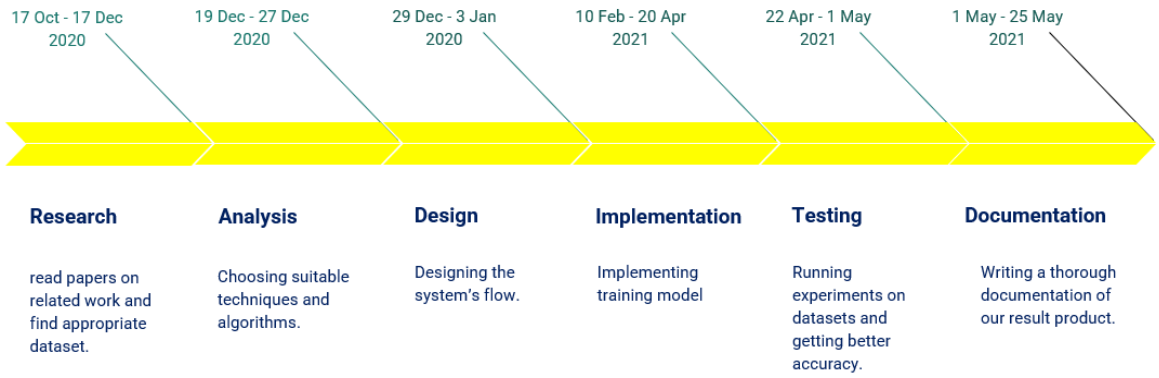


Figure 1 Time plan

1.5 Document Organization

Chapter 2 – Background

Background on Abnormality Detection and Localization in Chest X-Rays and related work of the any different architecture that works on Chest X-Rays.

Chapter 3 – Analysis and Design

-System overview where we discuss in detail the system architecture of the DenseNet and how it works with our model.

-Showing System Analysis and Design (use case , class diagram , sequence diagram , Database diagram)

Chapter 4 – Implementation and Testing

-Identify Environments and Libraries that we use, our dataset and experiments and results that we achieve.

Chapter 5 – User Manual

-Explaining how to use our application in detail steps.

Chapter 6 – Conclusion and Future Work

- What we conclude from our research and the final we wish to do in the future.

Chapter 7 – References

-all paper and researches that helps to fully understand the problem and last improves happens.

Chapter 2

Background

2- Background

- Chest radiograph interpretation is critical for the detection of thoracic diseases, which affect millions of people worldwide each year. This time-consuming task typically requires expert radiologists to read the images, leading to fatigue-based diagnostic error and lack of diagnostic expertise in areas of the world where radiologists are not available.
- There exist many people work in this filed some of them work in Heart disease to detect abnormality in the XRAY and some of other papers work in lung diseases but not all of them just only 2 or 3 diseases.
- Some of existing systems are make binary classification to detect normality and abnormality in the image this system takes an image and run on it some deep learning algorithms and the output of them only if the image has some abnormality, and some other could detect and tell us what is the disease in the image but the only deal with 2 or 3 diseases and some other give us the probability of each disease.
- We choose 5 diseases Edema, Pleural-Thickening, Consolidation, Atelectasis and Effusion according to medical specialist recommendation.
- deep learning approaches have been able to achieve expert-level performance in medical image (XRAY) interpretation tasks, powered by large network architectures and fueled by the emergence of large labeled datasets.
- We used different version of denseNet Deep Learning model consists of 121-layer, 161-layer, 169 layer and 201 layer and we choose the model with 121 layers according too a paper was tested them all and 121 layer was the best.

- Any deep learning model consist of many layer and each one of them has his own work to do and combine them together to give us in the end the final model that's we have
- The layers of CNN model that's used in DenstNet are Convolution Layers, Pooling Layers, Transition Layers, Flattening and Full Connection
    - **Convolution Layers:**
        - Convolution is an orderly procedure where two sources of information are intertwined; it's an operation that changes a function into something else. Convolutions have been used for a long time typically in image processing to blur and sharpen images, but also to perform other operations. (e.g. enhance edges and emboss) CNNs enforce a local connectivity pattern between neurons of adjacent layers.
        - In deep learning, a convolutional neural network (CNN or ConvNet) is a class of deep neural networks, that are typically used to recognize patterns present in images but they are also used for spatial data analysis, computer vision, natural language processing, signal processing, and various other purposes The architecture of a Convolutional Network resembles the connectivity pattern of neurons in the Human Brain and was inspired by the organization of the Visual Cortex. This specific type of Artificial Neural Network gets its name from one of the most important operations in the network: convolution
    - **Pooling Layers**
        - A pooling layer is a new layer added after the convolutional layer. Specifically, after a nonlinearity (e.g. ReLU) has been

applied to the feature maps output by a convolutional layer; for example, the layers in a model may look as follows:

1- **Input Image**
2- **Convolutional Layer**
3- **Nonlinearity**
4- **Pooling Layer**

- The addition of a pooling layer after the convolutional layer is a common pattern used for ordering layers within a convolutional neural network that may be repeated one or more times in a given model.
- The pooling layer operates upon each feature map separately to create a new set of the same number of pooled feature maps.
- Pooling involves selecting a pooling operation, much like a filter to be applied to feature maps. The size of the pooling operation or filter is smaller than the size of the feature map; specifically, it is almost always 2×2 pixels applied with a stride of 2 pixels.
- This means that the pooling layer will always reduce the size of each feature map by a factor of 2, e.g. each dimension is halved, reducing the number of pixels or values in each feature map to one quarter the size. For example, a pooling layer applied to a feature map of 6×6 (36 pixels) will result in an output pooled feature map of 3×3 (9 pixels).
- The pooling operation is specified, rather than learned. Two common functions used in the pooling operation are:

- **Average Pooling:** Calculate the average value for each patch on the feature map.
- **Maximum Pooling (or Max Pooling):** Calculate the maximum value for each patch of the feature map.
- The result of using a pooling layer and creating down sampled or pooled feature maps is a summarized version of the features detected in the input. They are useful as small changes in the location of the feature in the input detected by the convolutional layer will result in a pooled feature map with the feature in the same location. This capability added by pooling is called the model's invariance to local translation.

- **Transition Layers**
  - The authors refer to the layers between the dense blocks as transition layers which do the convolution and pooling.
  - we know that the transition layers used in the DenseNet architecture consist of a batch-norm layer, 1x1 convolution followed by a 2x2 average pooling layer.
  - Given that the transition layers are pretty easy
- **Flattening**

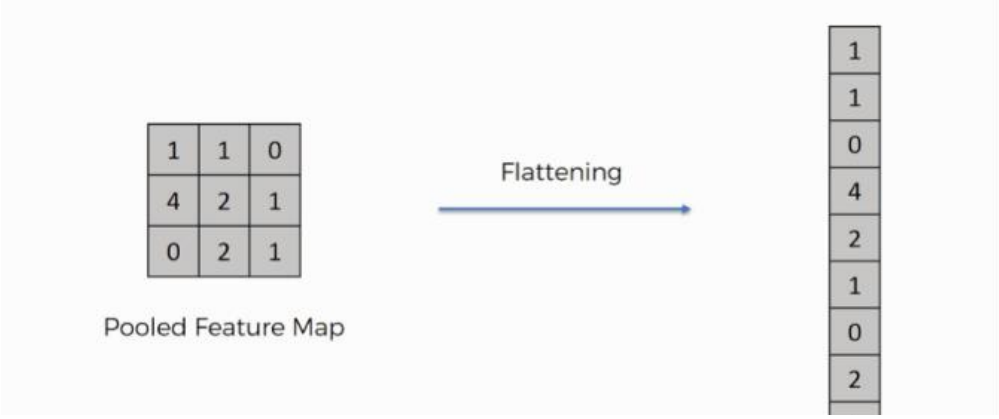


Figure 2 Flattening [1]

- After finishing the previous two steps, we're supposed to have a pooled feature map by now. As the name of this step implies, we are literally going to flatten our pooled feature map into a column like in the image [3].
- The reason we do this is that we're going to need to insert this data into an artificial neural network later on.
- As you see in the image above, we have multiple pooled feature maps from the previous step.

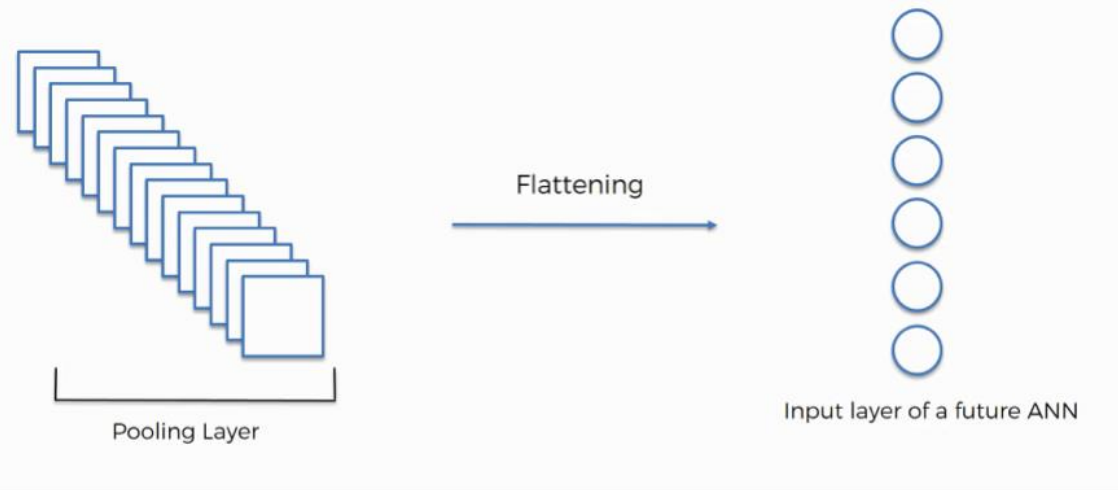


Figure 3 Flattening [2]

- What happens after the flattening step is that you end up with a long vector of input data that you then pass through the artificial neural network to have it processed further like the image [3]

- **Full Connection**
  - Here's where artificial neural networks and convolutional neural networks collide as we add the former to our latter. It's here that the process of creating a convolutional neural network begins to take a more complex and sophisticated turn.

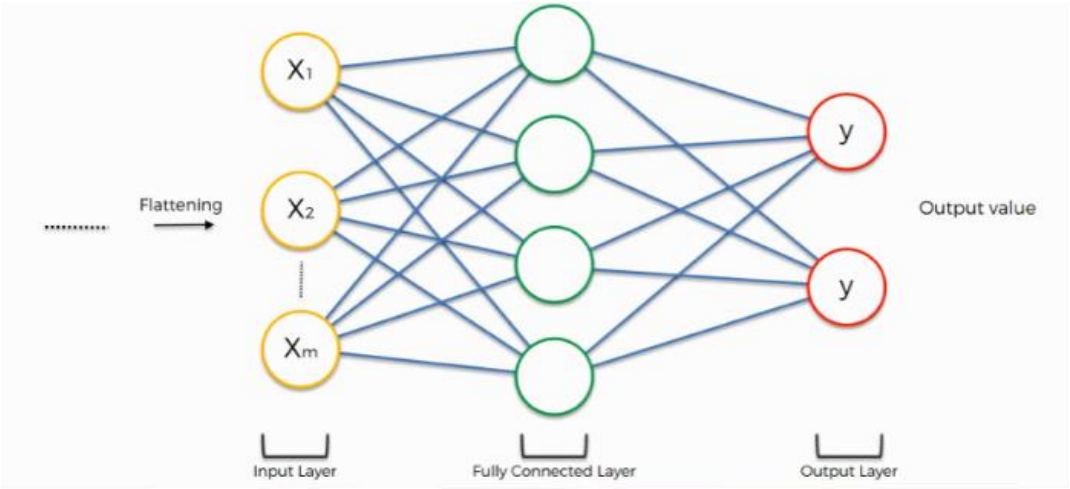


Figure 5Full Connection [1]

- As you see from the image [4] , we have three layers in the full connection step:
  - Input layer
  - Fully-connected layer
  - Output layer
- Notice that when we discussed artificial neural networks, we called the layer in the middle a “hidden layer” whereas in the convolutional context we are using the term “fully-connected layer.”

- **The Full Connection Process**
  - As we said in the previous, the input layer contains the vector of data that was created in the flattening step. The features that we distilled throughout the previous steps are encoded in this vector.
  - At this point, they are already sufficient for a fair degree of accuracy in recognizing classes. We now want to take it to the next level in terms of complexity and precision.

❖ **What is the aim of this step?**

- The role of the artificial neural network is to take this data and combine the features into a wider variety of attributes that make the convolutional network more capable of classifying images, which is the whole purpose from creating a convolutional neural network.
- We can now look at a more complex example than the one at the beginning of the tutorial.

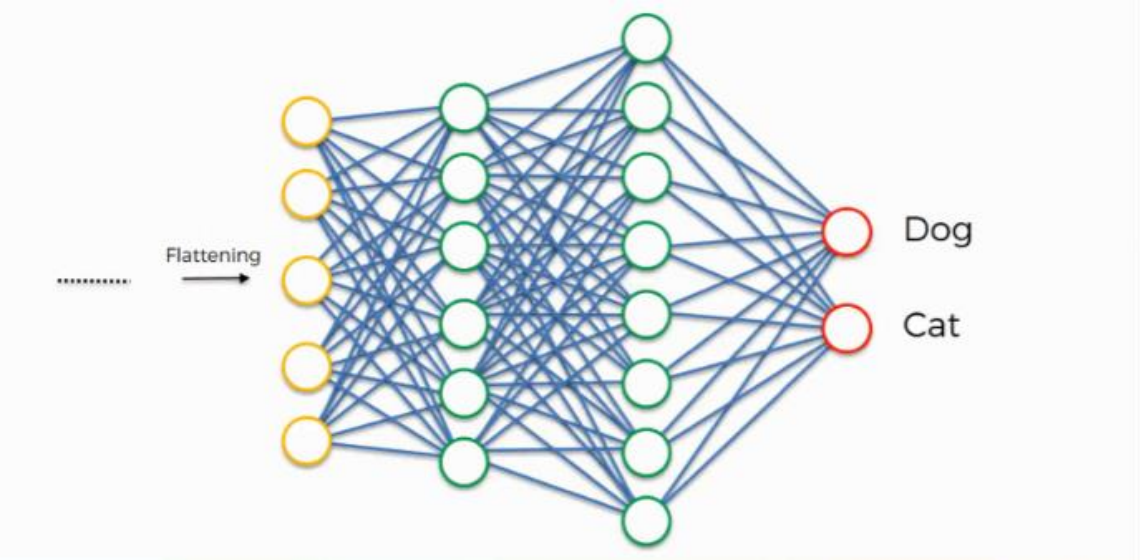


Figure 6 Full Connection [2]

- We'll explore how the information is processed from the moment it is inserted into the artificial neural network and until it develops its classes (dog, cat) like the image [5].
- At the very beginning, as you know by now, we have an input image which we convolve, pool, flatten, and then pass through the artificial neural network.
- By the end of this channel, the neural network issues its predictions. Say, for instance, the network predicts the figure in the image to be a dog by a probability of 80%, yet the image

actually turns out to be of a cat. An error has to be calculated in this case.

- In the context of artificial neural networks, we call this calculation a “cost function” or a mean squared error, but as we deal with convolutional neural networks, it is more commonly referred to as a “loss function.” We use the cross-entropy function in order to achieve that.
- The cross-entropy function and mean squared errors will be discussed in detail in a separate tutorial, so, if you're interested in the topic, you can stick around for the extra tutorials at the end of this section.
- For now, all you need to know is that the loss function informs us of how accurate our network is, which we then use in optimizing our network in order to increase its effectiveness. That requires certain things to be altered in our network.
- These include the weights (the blue lines connecting the neurons, which are basically the synapses), and the feature detector since the network often turns out to be looking for the wrong features and has to be reviewed multiple times for the sake of optimization.
- Just as we said when discussing artificial neural networks, the information is then conveyed in the opposite direction as you see in the figure below. As we work to optimize the network, the information keeps flowing back and forth over and over until the network reaches the desired state.

- As you're always reminded, this process is explained here in an intuitive manner, but the science and the mathematics behind it are more complex, of course.

❖ **Class Recognition**

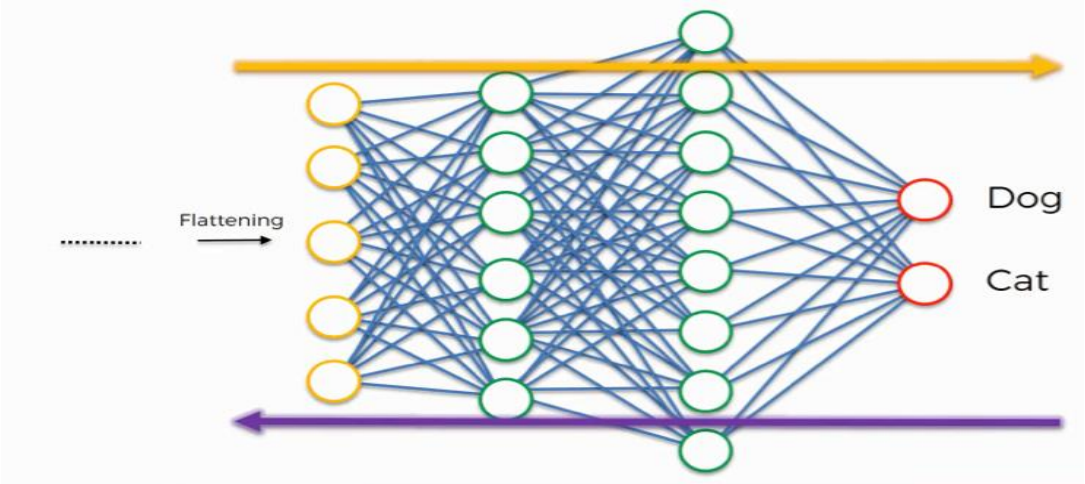


Figure 7 Class Recognition [1]

- In order to understand how it will play out, we need to check out the weights placed on each synapse linking to this class so that we can tell which attributes/features are most relevant to it like image [6].
- This full connection process practically works as follows:
  - The neuron in the fully-connected layer detects a certain feature; say, a nose.
  - It preserves its value.
  - It communicates this value to both the “dog” and the “cat” classes.
  - Both classes check out the feature and decide whether it's relevant to them.

- In our example, the weight placed on the nose-dog synapse is high (1.0), which means that the network is confident that this is a dog's nose.
- Since the information is constantly flowing in both directions, the “cat” class takes note of this and understands that since this is a dog's nose, then it simply can't be a cat's nose. Even if at first it would have considered the signal saying “small rounded nose” because this might be a cat's as well as a dog's nose, now it dismisses this feature.
- This happens gradually as it receives the same reading multiple times. The dog class on its part will start focusing more on the attributes carrying the highest weight (the three thick purple lines), and it will ignore the rest.
- The same process simultaneously occurs with the cat, enabling it to pick out its own priority features. What we end up with is what you see in the figure [7]. As this process goes on repeat for thousands of times, you find yourself with an optimized neural network.

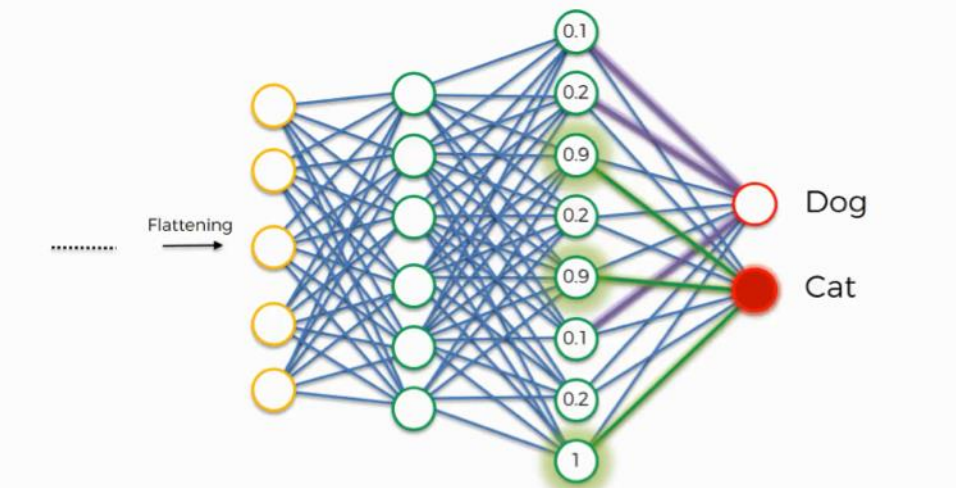


Figure 8 Class Recognition[2]

Chapter 3

Analysis and Design

3- Analysis and Design

3.1 System Overview

3.1.1 System Architecture:

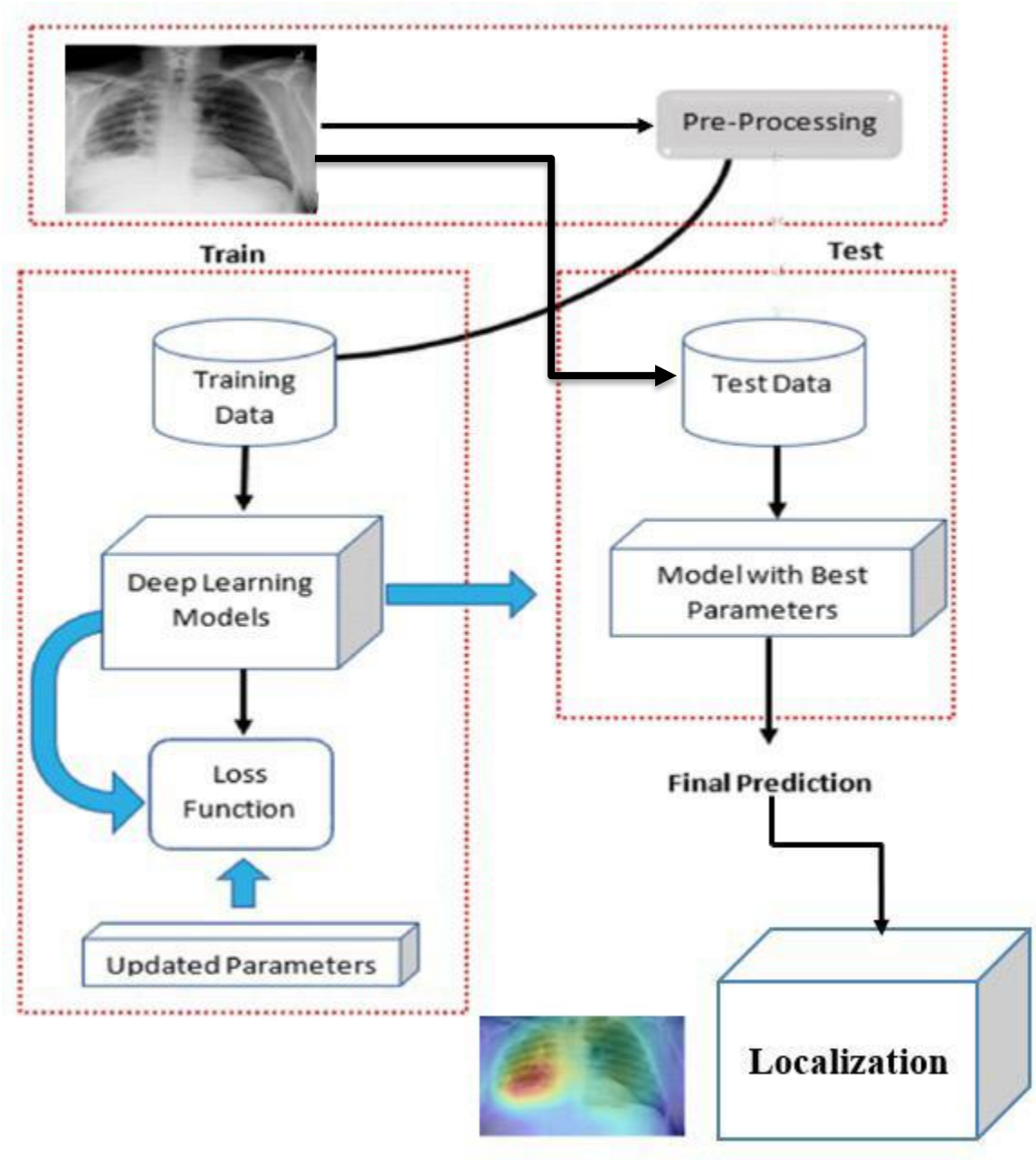


Figure 9 System Architecture

In the system architecture we have more than on phase

Pre-Processing Phase:

1. Image enter as an input for our system
2. Apply some pre - processing on it:
    - normalization
    - flip horizontal augmentation
    - resize scale to 224
3. Send the image after made the edit by the pre - processing on it
4. Then send the output image to the DensNet model to apply on it
5. After this the image send to another model to apply the heat map on it

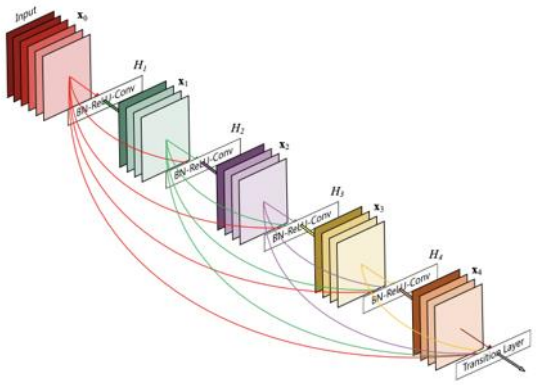
3.1.2 Network Architecture:

DenseNet Architecture

DenseNet is one of the new discoveries in neural networks for visual object recognition. In a normal feedforward network an output from a layer is passed on to only the next occurring layer and as the network goes deep it would have access to only the higher-level features but not those from lover levels closer to input. In Denset every subsequent layer receives input from all the previous layers in the chain as can be observed from above figure. All these values are concatenated so that information is not lost. The concatenated feature map is passed through a composite function consisting of Batch Normalization,

Relu and 3x3 Convolution.

The output is then passed on to every subsequent layer and the process is repeated.



DenseNet Structure

$$a^{[l]} = g\left([a^{[0]}, a^{[1]}, a^{[2]}, \dots\dots\dots\dots, a^{[l-1]}\right)$$

Figure 10 DenseNet Structure

But simple concatenation can lead to very huge number of parameters in a deep network. So the architecture is divided into multiple dense blocks each separated by a block called transition block. In dense block the information flows in the way described in above paragraph. Transition block is the place where down sampling occurs to prevent blowing up. The transition layer consists of batch normalization layers followed by 1x1 convolution layer which is followed by 2x2 average pooling layer as shown in the fig [11]

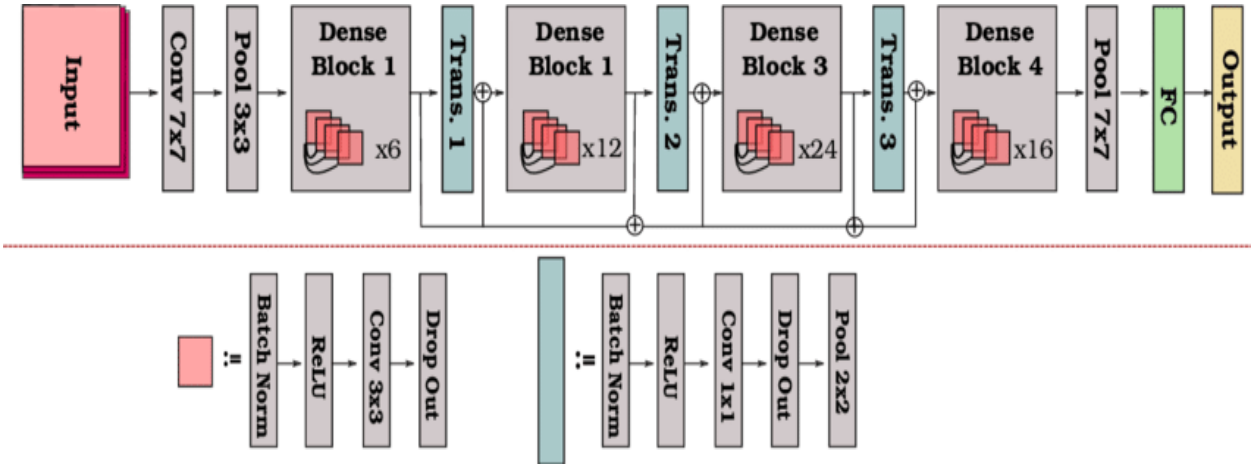


Figure 11 DensNet Architecture

Details of each phase of DenseNet Model:

Layers	Output Size	DenseNet-121
Convolution	112x112	7x7 conv, stride2
Pooling	56x56	3x3 max pool, stride 2
Dense Block 1	56x56	[1x1 conv ; 3x3 conv] x 6
Transition Layer 1	56x56	1x1 conv
	28x28	2x2 average pool, stride 2
Dense Block 2	28x28	[1x1 conv ; 3x3 conv] x 12
Transition Layer 2	28x28	1x1 conv
	14x14	2x2 average pool, stride 2
Dense Block 3	7x7	[1x1 conv ; 3x3 conv] x24

Figure 12 details of each phase of our DensNet Model

Localization Phase:

- We get last conv. Layer from trained model
- Feed image to last layer of the model
- Get classification weight from trained model
- Get feature map matrix from last conv. Layer
- Multiply feature map with weight of classification
- apply it by adding on x-ray images

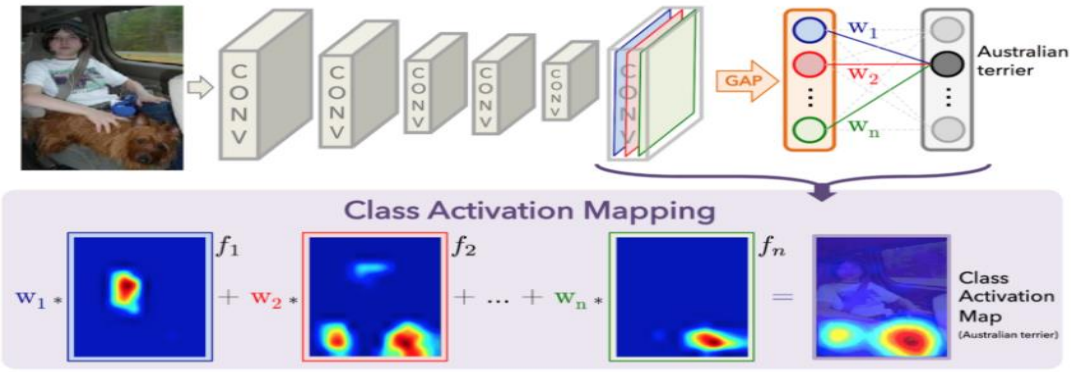


Figure 13 Localization phase Arch.

3.1.2 System Users

A. Intended Users:

- **Radiologist**: All that is required of the specialist is to insert an image into the system, and the system will produce an image of the expected disease

- **Chest Doctors**: All that is required of the doctor is to insert an image into the system, and the system will produce an image of the expected disease

B. User Characteristics

To be specialists in XRAY and they can understand and analyze it and to be specialists in treating chest diseases so that they can prescribe treatment to the patient after the disease is diagnosed through the system

3.2 System Analysis & Design

3.2.1 Use Case Diagram

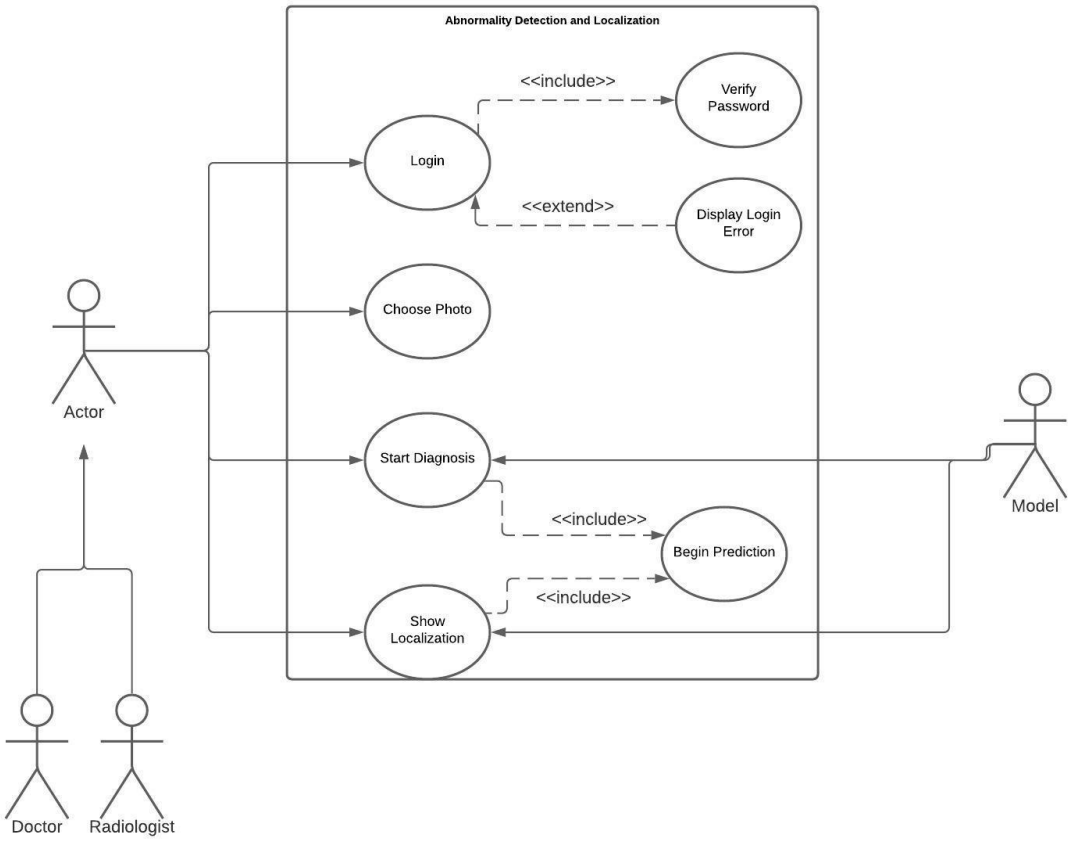


Figure 14 Use case Diagram

3.2.2 Class Diagram

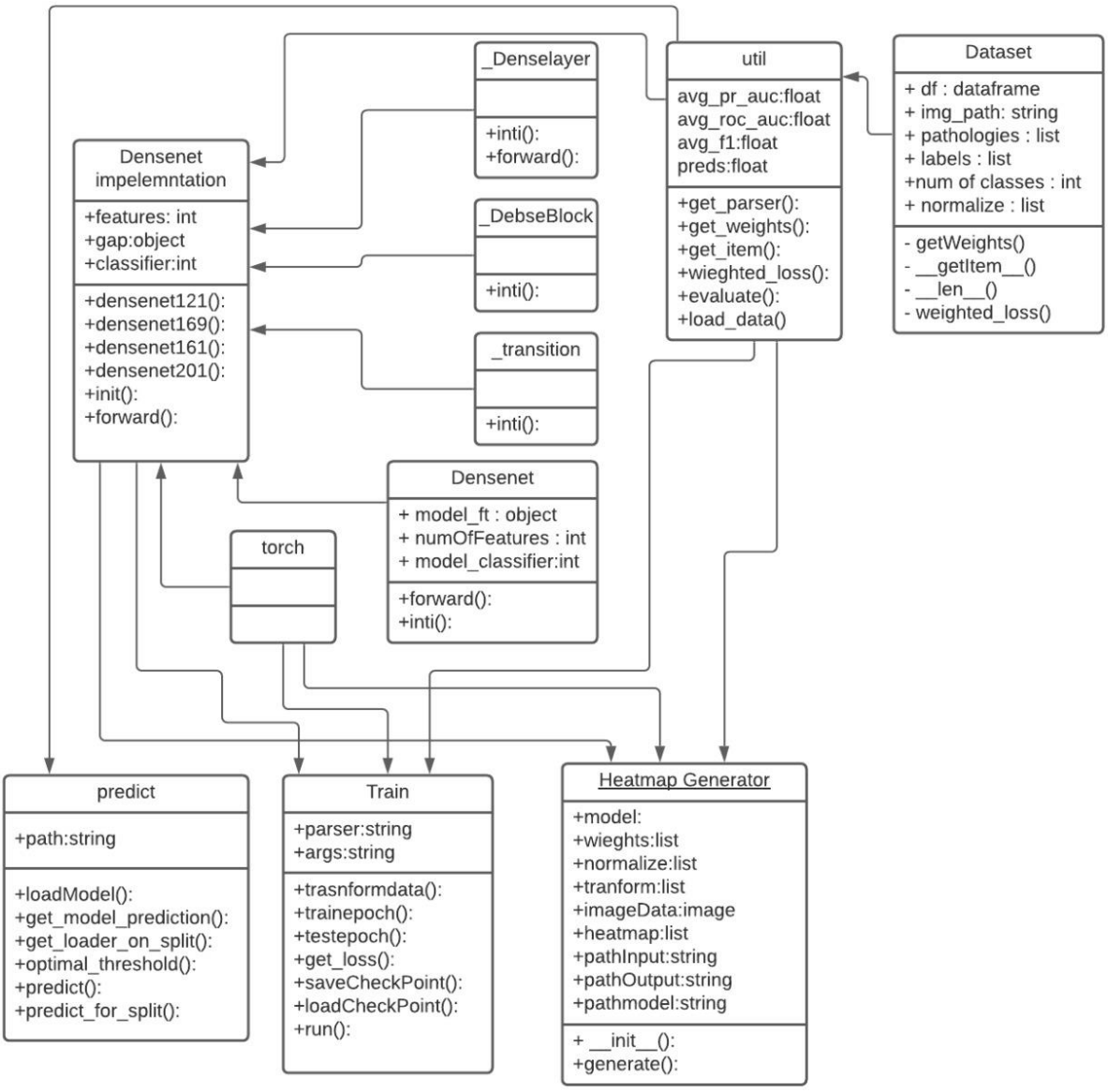


Figure 15 -class diagram

1-Densenet

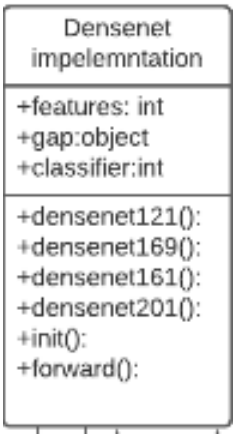


Figure 16 class diagram 2

- download pretrained densenet models with different layers number
- implement Densenet class structure like DenseLayer and DenseBlock to be used with classififcation
- DenseLayer applies 2D Batch Normalization, ReLu activation function and 2D convolutin.
- DenseBlock creates a blocks of DenseLayers according to number of layers in model.
- Transition applies 2D Batch Normalization, ReLu activation function, 2D convolutin and 2D Average pooling
- init(): initializes the model with convolutional layer, denseblock, final batch normalization, and classification layer

2-util

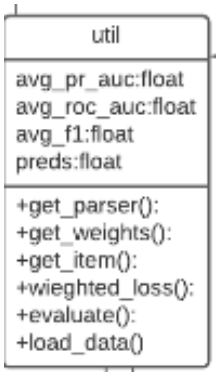


Figure 17 class diagram 3

- creates Argument Parser which used for creating commands to run the model
- creates Dataset class which prepares the dataset for training
- evaluates each pathology with its presence in dataset and computes metrics (auc, roc, F1, accuracy) for each pathology

3-Dataset

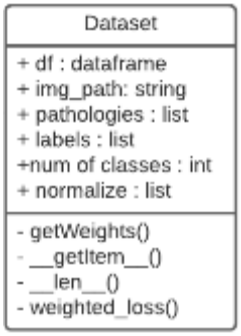


Figure 18 class diagram 3

- reads csv file containing information about pathologies
- creates data frame of dataset
- gets the path of training images
- gets pathologies names from dataset file
- gets labels of each dataset sample(0 for absence and 1 for presence)
- count the number of different pathologies
- apply data normalization to dataset
- splits the data into train and test
- get_weights(): checks if gpu is available and returns weights and calculates random loss
- __getitem__(): gets specific item(x-ray image)
- __len__(): return the length of dataset

4-Train

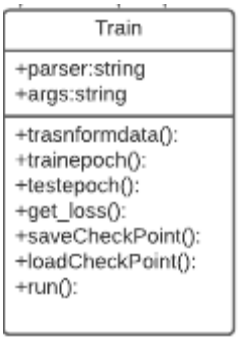


Figure 19 class diagram 4

- initializes an object of Densenet class
- transform_data(): applies data transformation for dataset to prepare it for training process
- train_epoch(): start model training and calculates batch losses and returns the train loss
- test_epoch(): evaluates the model and , calculate the validation loss and returns it
- save checkpoint for each epoch so we can load it later to continue learning process in case of any failure during training process
- run():
- checks if gpu is available for training, calculate train criterion and Val criterion, select the optimize to train with, then start training, and calc train and valid loss for each epoch and print the final best valid loss

5-Predicit

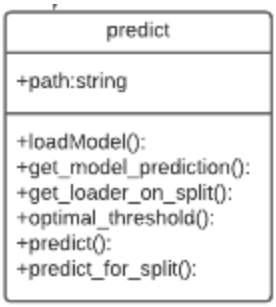


Figure 20 class diagram 5

- load_model(): loads pretrained model
- get_model_predictions(): returns the predictions of the model on validation set or testing set
- optimal_threshold_compute(): computes precision, recall, and computes optimal threshold of each pathology
- predict_for_split(): applies prediction method according to dataset split choice(valid, test) and returns the probabilities and thresholds
- predict(): gets parameters of model, creates a path to save the prediction result, calculates probabilities and thresholds from predict_for_split() function, and return them.

6-Heatmap

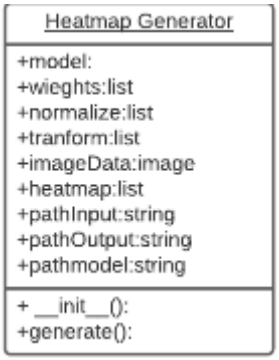


Figure 21 class diagram 6

- __init__(): loads the pretrained model with its features and weights, applies transforms for the image(resize, normalize)
- generate(): loads the image, pass it to the model and generates a heatmap to localize the area of abnormality detected.

3.2.3 Sequence Diagram

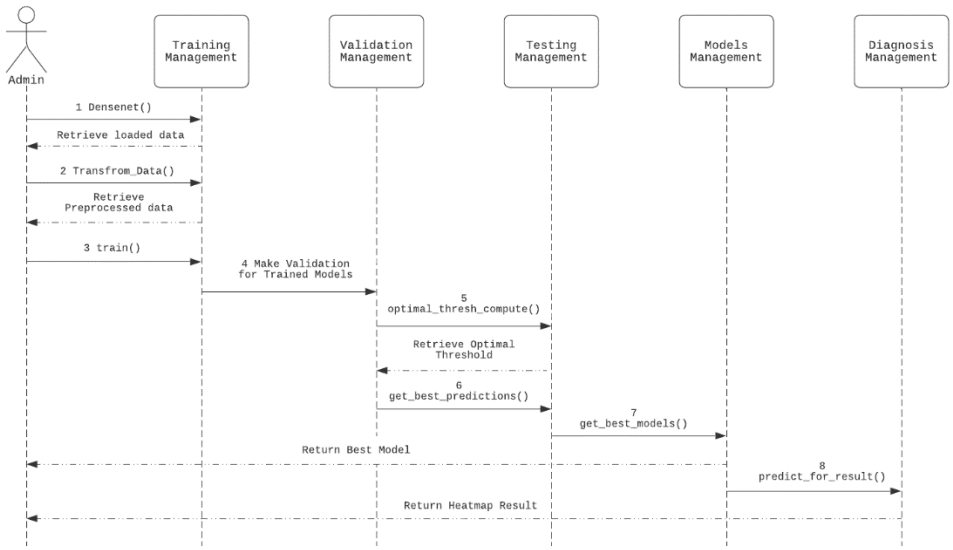


Figure 22 sequence diagram

3.2.4 Database Diagram

A dataset with the image's names of XRAY and each of them have the diseases name that image have.

info	
image path	string
Pleural-Thickening	boolean
Edema	boolean
Consolidation	boolean
Atelectasis	boolean
Effusion	boolean
Normal	boolean

Figure 23 Database diagram

Chapter 4

Implementation and Testing

4- Implementation and Testing

4.1- Environments and Libraries

- **Python (PyCharm)**
  Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

- **Google Colab**
  Collaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

- **PyTorch**

  PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab (FAIR). It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface.

  A number of pieces of deep learning software are built on top of PyTorch, including Tesla Autopilot, Uber’s Pyro, Hugging Face’s Transformers, PyTorch Lightning, and Catalyst.

4.2- Dataset

- We worked on NIH dataset (National Institutes of Health - Clinical Center).
- NIH Dataset Link
- 112,120 chest x-ray images consist of normal and abnormal (14 – diseases) each images have resolution 1024 * 1024 (Public dataset)
- our Dataset contain diseases (Large data), so we take combination of 5 diseases according to:
    1. Chest radiologist advice
       (similar– dis-similar) diseases.
    2. arrange diseases according to which have largest number of images

1. Dataset represented in binary form as show in Figure [24]

	A	B	C	D	E	F	G
1	Path	Pleural-Th	Edema	Consolida	Atelectasi	Effusion	Normal
2	images/00000005_002.png	0	0	0	0	0	1
3	images/00000005_003.png	0	0	0	0	0	1
4	images/00000005_004.png	0	0	0	0	0	1
5	images/00000011_004.png	0	0	0	0	1	0
6	images/00000013_016.png	1	0	0	0	1	0
7	images/00000015_000.png	0	0	0	1	0	0
8	images/00000017_001.png	0	0	0	0	1	0
9	images/00000017_002.png	0	0	0	0	1	0
10	images/00000025_000.png	0	0	0	0	1	0
11	images/00000035_000.png	0	0	0	0	0	1
12	images/00000039_002.png	0	0	0	0	0	1
13	images/00000039_003.png	0	0	0	0	0	1
14	images/00000039_004.png	0	0	0	0	1	0
15	images/00000040_000.png	0	0	0	0	0	1
16	images/00000042_000.png	0	0	0	0	0	1
17	images/00000042_004.png	0	0	0	0	0	1

Figure 25 Dataset representation

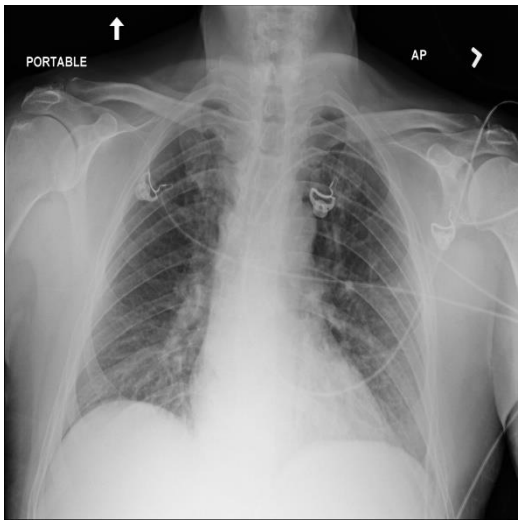


Figure Dataset representation

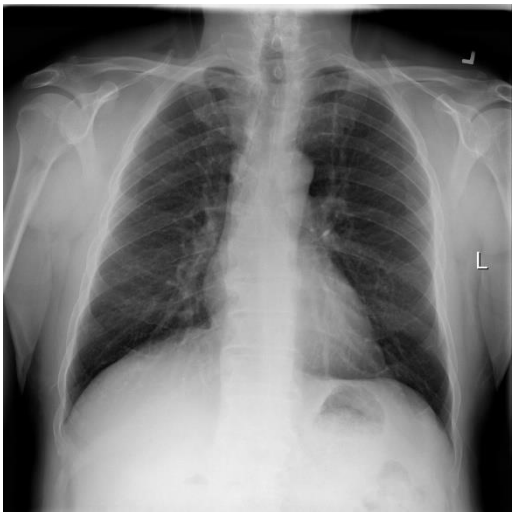


Figure Dataset representation

4.3- Experiments

The diseases we use in our experiments are:

1	Pleural-Thickening
2	Edema
3	Consolidation
4	Atelectasis
5	Effusion
6	Cardiomegaly
7	Hernia
8	Pneumothorax
9	Nodule
10	Infiltration

We take 3 different combinations of those diseases according to Radiologist advice. The result is shown in Exp 5.

- **Experiment 1**, we compare between Densenet121 and Densenet161 and the accuracy in Densenet121 is better than Densenet161 by 2%

So, we choose Densenet121

Exp.	model	optimizer	Learning rate	scale	N. of Diseases	N. of Images	Average AUC
1	Densenet161	Adam	0.0001	224	5	26,100	0.78
2	Densenet121	Adam	0.0001	224	5	26,100	0.80

Table 1- Exp1 result

- **Experiment 2**, we try to change Optimizer and compare between Adam optimizer and rmsprop optimizer and recognize that rmsprop is better by 0.01%

So, we choose **rmsprop optimizer**

Exp.	model	optimizer	Learning rate	scale	N. of Diseases	N. of Images	Average AUC
1	Densenet121	Adam	0.0001	224	5	22,100	0.923
2	Densenet121	rmsprop	0.0001	224	5	22,100	0.924

Table 2 Exp2 Result

-Experiment 3, we change learning rate between 0.0001 and 0.0001 and 0.001 and realize that 0.001 is better between 2%-3%

So, we choose **0.0001 Learning rate**

Exp.	model	optimizer	Learning rate	scale	N. of Diseases	N. of Images	Average AUC
1	Densenet121	Adam	0.00001	224	5	22,100	0.90
2	Densenet121	Adam	0.0001	224	5	22,100	0.923
3	Densenet121	Adam	0.001	224	5	22,100	0.894

Table 3Exp 3 result

- Experiment 4, we change image scale between 512 and 244 and realize that 244 is better between by 1.5%

So, we choose **scale 244**

Exp.	model	optimizer	Learning rate	scale	N. of Diseases	N. of Images	Average AUC
1	Densenet121	rmsprop	0.0001	512	5	22,100	0.91
2	Densenet121	rmsprop	0.0001	224	5	22,100	0.925

Table 4 Exp4 result

- **Experiment 5, we worked on different combination of disease**
  1. Infiltration, Atelectasis, Effusion, Nodule, Pneumothorax
  2. Cardiomegaly, Hernia, Edema, Effusion, Pneumothorax.
  3. Pleural-Thickening, Edema, Consolidation, Atelectasis, Effusion.

So, we choose **3rd combination**

Exp.	model	optimizer	Learning rate	scale	N. of Diseases	N. of Images	Average AUC
1	DesnseNet121	Adam	0.0001	224	5	26,100	0.80
2	DesnseNet121	Adam	0.0001	224	5	38,900	0.84
3	DesnseNet121	Adam	0.0001	224	5	22,100	0.923

Table 5 Exp 5 result

-Overall Experiments & Results

In this table we show result of all experiment together , in the end it show best result : Average AUC : 0.925 used DenseNet121 layer , rmsprop as optimizer and scale image to 224.

Exp.	model	optimizer	Learning rate	scale	N. of Diseases	N. of Images	Average AUC
1	DesnseNet161	Adam	0.0001	224	5	26,100	0.78
2	DesnseNet121	Adam	0.0001	224	5	26,100	0.80
3	DesnseNet121	Adam	0.0001	224	6	38,900	0.84
4	DesnseNet121	Adam	0.00001	224	5	22,100	0.90
5	DesnseNet121	rmsprop	0.00001	224	5	22,100	0.91
6	DesnseNet121	rmsprop	0.0001	512	5	22,100	0.91
7	DesnseNet121	Adam	0.0001	224	5	22,100	0.923
8	DesnseNet121	rmsprop	0.0001	224	5	22,100	0.924
9	DesnseNet121	rmsprop	0.0001	224	5	22,100	0.925

Table 6 overall Exp

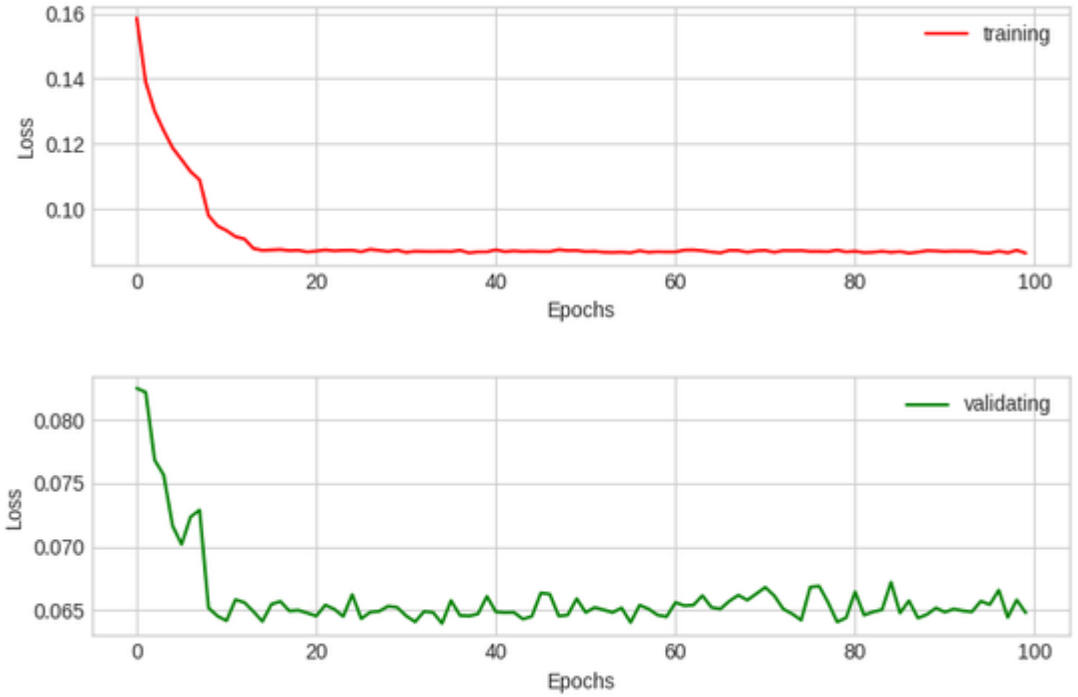


Figure 26 Train Graph

After apply all this experiments and compare between disease and take best 5 disease, we improved accuracy by mean of 3.5%

	Disease Name	Paper that we take	Our paper
1	Pleural-Thickening	79.8	77.89
2	Edema	92.4	95.92
3	Consolidation	89.3	95.92
4	Atelectasis	86.2	90.28
5	Effusion	90.1	95.65

Chapter 5

User Manual

5- User Manual

We have desktop application
Step 1 : Run the app
Step 2 : **Browse image** as shown in the figure

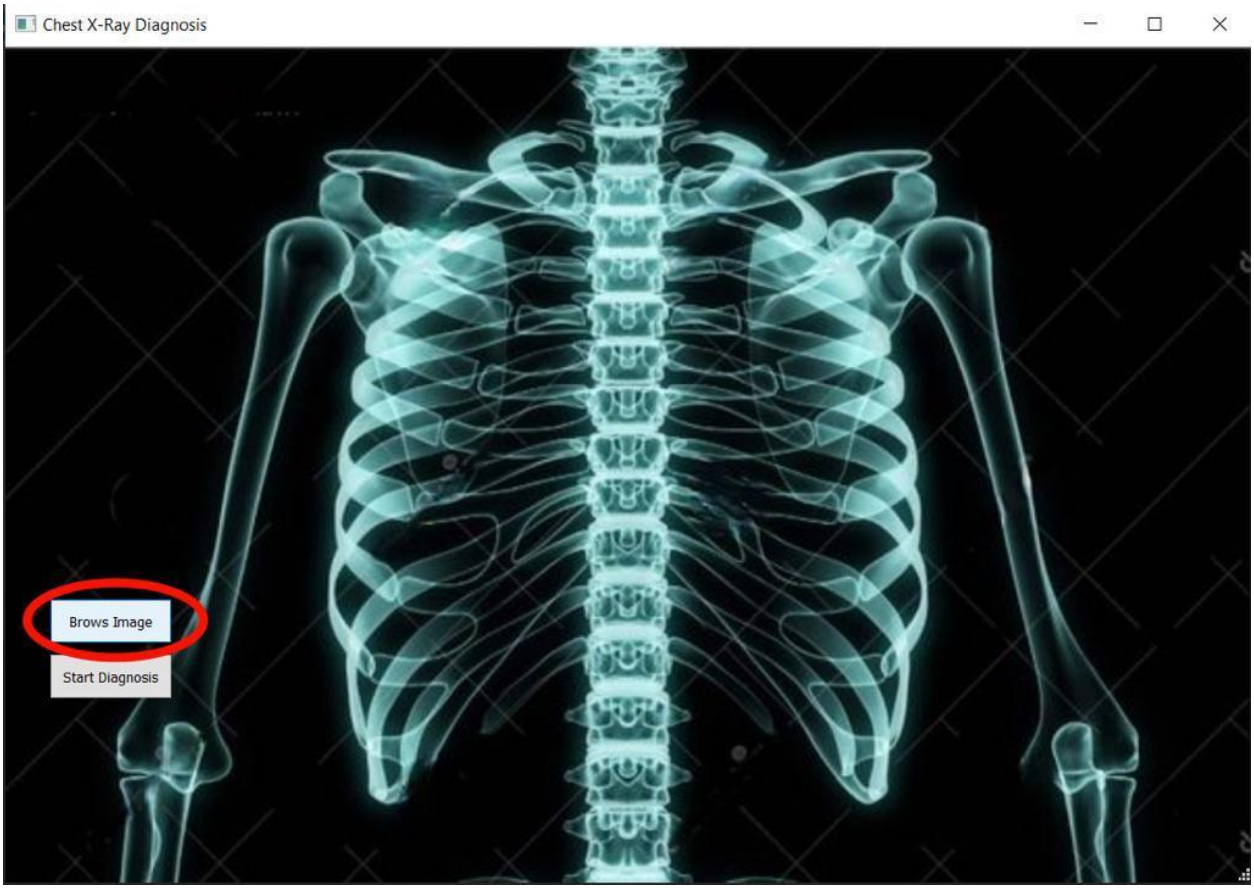


Figure 27user manual 1

Step 3 : **<u>select image for diagnose</u>** as shown in the figure

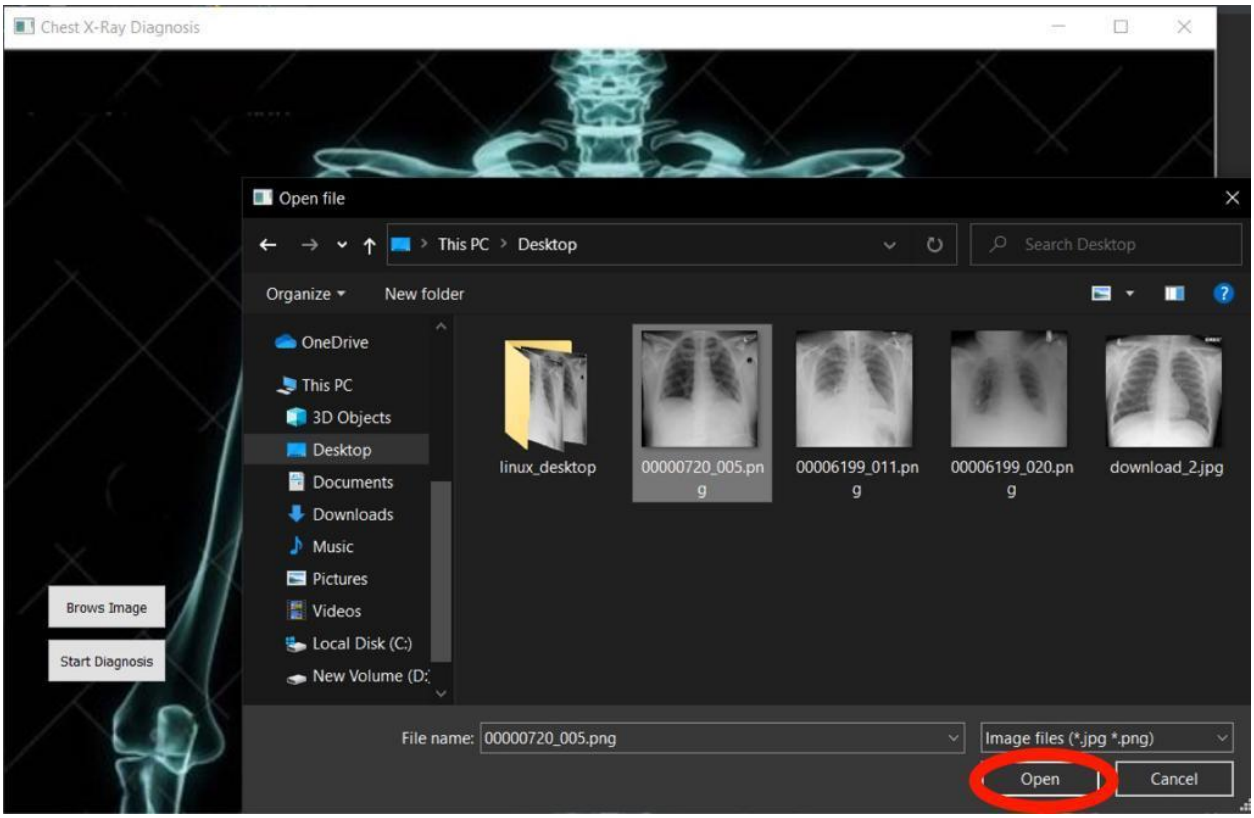


Figure 28 user manual 2

Step 4: **Press diagnose** as shown in the figure

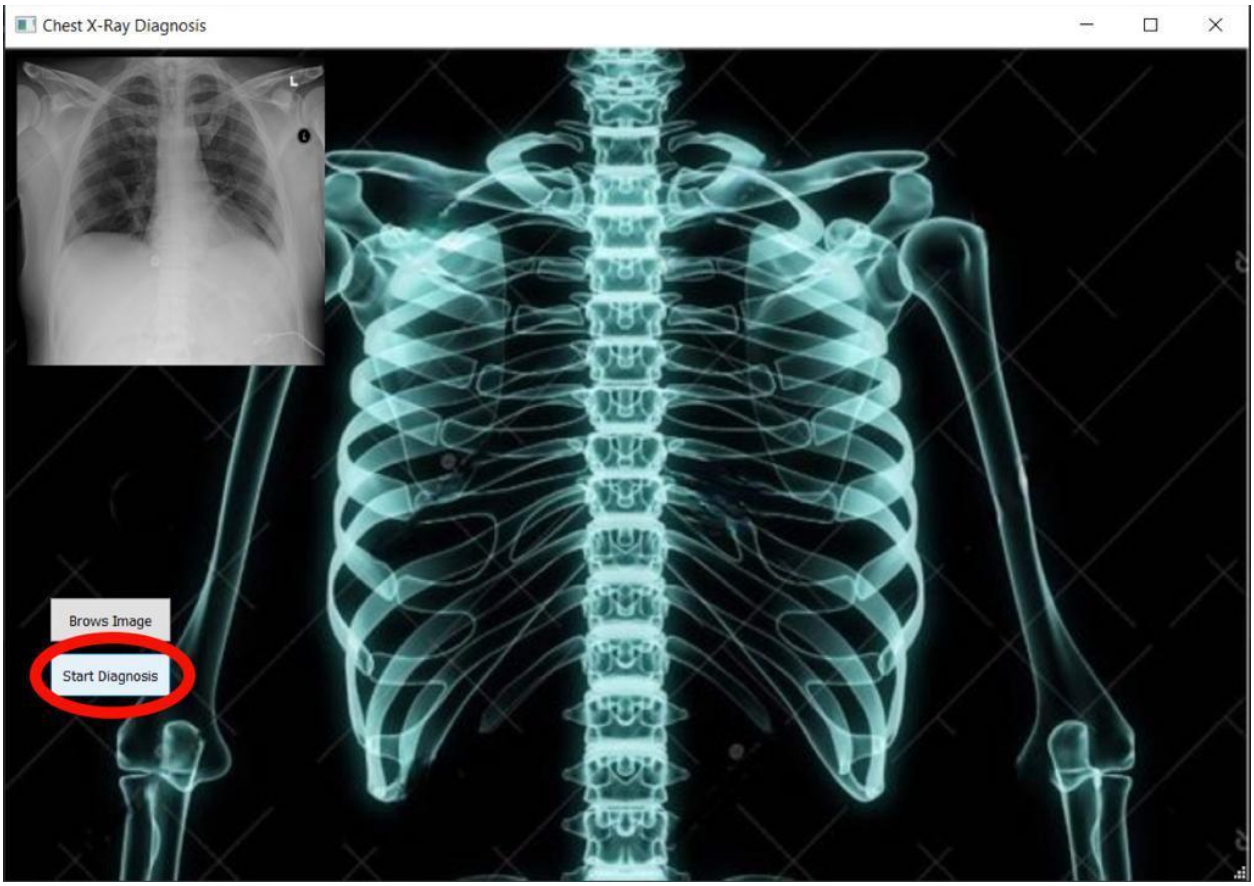


Figure 29 manual user 3

Result will be shown as this

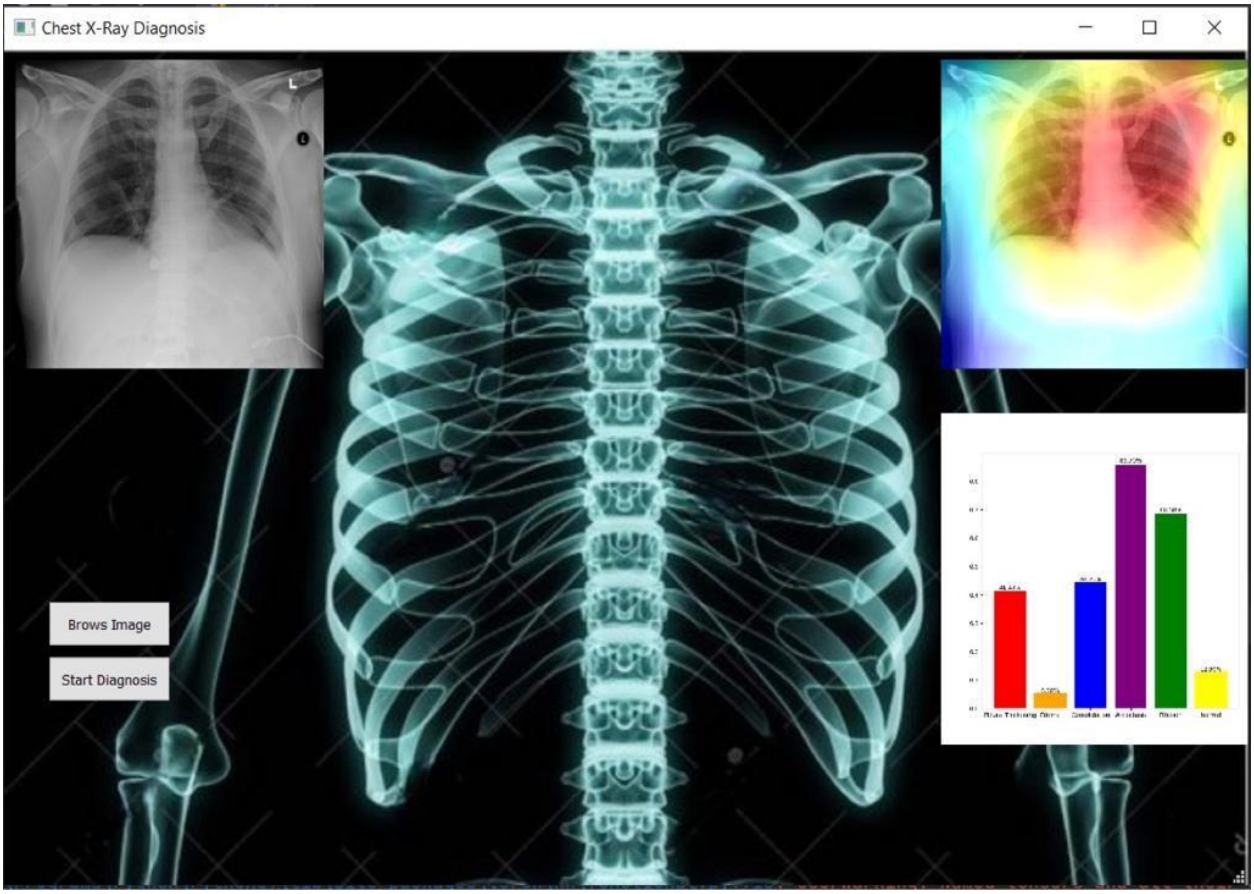


Figure 30user manual 4

Chapter 6

Conclusion and Future Work

6- Conclusion and Future Work

6.1 Conclusion

We present a deep learning algorithm that performs comparably to practicing board-certified radiologists in the detection of multiple thoracic pathologies in frontal-view chest radiographs. This technology may have the potential to improve healthcare delivery and increase access to chest radiograph expertise for the detection of a variety of acute diseases. Further studies are necessary to determine the feasibility of these outcomes in a prospective clinical setting.

discussed the application of deep learning models for the identification of identify some of diseases affecting the chest and we the convolutional neural network help us in this because the convolutional neural network has a long time get the best result for the same cases and their exist a lot of model in this technology but we use one model called DensNet and this model have more than one version but we choose the version called DensNet-121 and get us a good result because of the way of this model get the feature from the X-Ray this technique is to divide the model layers to some of blocks and each block get the input from the all previous block and send it to the all next blocks and because of this we can get more information from the image to help the model to get better result and also we try to make a localization for the diseases location in the X-Ray and we get a good result but of course the localization need some enhancement and we use the output image that's the model get and send it to the localization model and the final image must be colored image the darker red color refer to that is the diseases place.

The data set we choose to work with have 14 diseases but we choose only 5 and this happened because of our resources are limited and the 5 diseases are Pleural-Thickening, Edema, Consolidation, Atelectasis and Effusion and we choose them depend on some of experiences we made on the data.

In our survey we find more than one data set but we choose NIH chest X-Ray Dataset because it was a public dataset and we asked some expert and they say it's a good dataset for our project and our idea.

After all this things we got a good accuracy for each disease we can predict the Pleural-Thickening by 77.89% , Edema by 95.92 and this better than other model we find in the survey, Consolidation by 95.22 and also this better than other model we find in the survey, Atelectasis by 90.28 and also this better than other model we find in the survey, Effusion by 95.65 and also this better than other model we find in the survey, and we also can identify if the image have no disease and the average accuracy for our system is 92.5%

6.2 Future Work

We looking for to develop our model to work with maximum number of diseases and make the model work in any human organ and also need to get more accuracy to be more sure about the result we got not only but also need to add NLP to produce report for the users to make it easier and usable and helpful to be available for all people not only specialists.

Chapter 7

References

References

1. Tataru, Christine, et al. "Deep Learning for abnormality detection in Chest X-Ray images." *IEEE Conference on Deep Learning*. 2017.

2. Bhandari, Esha Dilipkumar, and Mahesh S. Badmera. "Chest Abnormality Detection from X-ray using Deep Learning." *Chest* 6.11 (2019).

3. Kieu, Phat Nguyen, et al. "Applying multi-CNNs model for detecting abnormal problem on chest x-ray images." *2018 10th International Conference on Knowledge and Systems Engineering (KSE)*. IEEE, 2018.

4. Rajarapollu, Prachi R., Debashis Adhikari, and Nutan V. Bansode. "Use of artificial neural network for abnormality detection in medical images." *Optimization in Machine Learning and Applications*. Springer, Singapore, 2020. 1-12.

5. Rakshit, Somnath, et al. "Deep Learning for Detection and Localization of Thoracic Diseases Using Chest X-Ray Imagery." *International Conference on Artificial Intelligence and Soft Computing*. Springer, Cham, 2019.

6. Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature. 2017 Feb; 542(7639):115–8. https://doi.org/10.1038/ nature21056 PMID: 28117445

7. Ehteshami Bejnordi B, Veta M, Johannes van Diest P, van Ginneken B, Karssemeijer N, Litjens G, et al. Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer. JAMA. 2017 12; 318(22):2199–210. https://doi.org/10.1001/jama.2017. 14585 PMID: 29234806

8. Cicero M, Bilbily A, Colak E, Dowdell T, Gray B, Perampaladas K, et al. Training and Validating a Deep Convolutional Neural Network for Computer-Aided Detection and Classification of Abnormalities on Frontal Chest Radiographs. Invest Radiol. 2017; 52(5):281–7. https://doi.org/10.1097/RLI.0000000000000341 PMID: 27922974

9. Bar Y, Diamant I, Wolf L, Lieberman S, Konen E, Greenspan H. Chest pathology detection using deep learning with non-medical training. In: 2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI). 2015. p. 294–7.

10. Maduskar P, Muyoyeta M,Pesce E, Ypsilantis P-P, Withey S, Bakewell R, Goh V, Montana G. Learning to detect chest radiographs containing lung nodules using visual attention networks. ArXiv171200996 Cs Stat [Internet]. 2017 Dec 4. Available from: http://arxiv.org/abs/1712.00996.

11. Guan Q, Huang Y, Zhong Z, Zheng Z, Zheng L, Yang Y. Diagnose like a Radiologist: Attention Guided Convolutional Neural Network for Thorax Disease Classification. ArXiv180109927 Cs [Internet]. 2018 Jan 30. Available from: http://arxiv.org/abs/1801.09927.

12. Wang X, Peng Y, Lu L, Lu Z, Bagheri M, Summers RM. ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. p. 3462–71.

13. Huang G, Liu Z, Maaten L v d, Weinberger KQ. Densely Connected Convolutional Networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. p. 2261–9.

14. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. ImageNet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. 2009. p. 248–55.

15. Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. Proc 3rd Int Conf Learn Represent ICLR [Internet]. 2014 Dec 22. Available from: http://arxiv.org/abs/1412.6980.

16. Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A. Learning Deep Features for Discriminative Localization. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016.p. 2921–9.

17. Cohen J. A Coefficient of Agreement for Nominal Scales. Educ Psychol Meas. 1960 Apr 1; 20(1):37–46.

18. Tibshirani R, Efron B. An introduction to the bootstrap [Internet]. CRC Press; 1994. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.473.2742.

19. Dunn OJ. Estimation of the Means of Dependent Variables. Ann Math Stat. 1958; 29(4):1095–111.

20. R Core Team. R: A Language and Environment for Statistical Computing [Internet]. R Foundation for Statistical Computing; 2017. Available from: https://www.R-project.org/.

21. Gamer M, Lemon J, Fellows I, Singh P. irr: Various Coefficients of Interrater Reliability and Agreement [Internet]. 2012. Available from: https://CRAN.R-project.org/package=irr.

22. Canty A, Ripley BD. boot: Bootstrap R (S-Plus) Functions. 2017.

23. Meyer MC. ConSpline: Partial Linear Least-Squares Regression using Constrained Splines [Internet]. 2017. Available from: https://CRAN.R-project.org/package=ConSpline/.

24. Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez J-C, et al. pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics. 2011 Mar 17; 12:77. https://doi.org/10.1186/1471-2105-12-77 PMID: 21414208

25. EkstrømCT. MESS: Miscellaneous Esoteric Statistical Scripts [Internet]. 2018. Available from: https://CRAN.R-project.org/package=MESS.

26. Wickham H. ggplot2: Elegant Graphics for Data Analysis [Internet]. Springer-Verlag New York; 2009. Available from: http://ggplot2.org.