Deep SaliencyNet - A Saliency Mapping Deep Learning Model

by

Jugal Kamlesh Panchal

(Under the Direction of Tianming Liu)

Abstract

Computer Vision with Deep Learning presents an intriguing combination in the field of Medical Image Processing and Analysis. Recent advancement in neural networks has made high-quality research plausible. Many models are developed, targeting specific application, within the Deep Learning community. However, very little has been explored for mapping the saliency information using the gaze information.

We present a Convolutional Neural Network (CNN), targeting a specific application, which foretells the saliency information on the image. Reading and interpreting a chest x-ray is a challenging task. We record the gaze data of radiologist who is deciphering the chest x-ray, using an eye tracker device, that acts as the ground truth. We perform various operations in the convolution neural network on every chest x-ray image in the dataset. Plotting the saliency information on the chest x-ray for easier interpretation is the eventual goal of this thesis project.

Index words: Saliency mapping, Chest X-Ray Interpretation, Deep Learning, Convolution Neural Network, Eye Tracker, Medical Image Analysis, TensorFlow

Deep SaliencyNet - A Saliency Mapping Deep Learning Model

by

Jugal Kamlesh Panchal

B.E., University of Mumbai, 2014

A Thesis Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Master of Science

Athens, Georgia

2018

Deep SaliencyNet - A Saliency Mapping Deep Learning Model

by

Jugal Kamlesh Panchal

Dedication

With Gratitude to my Late Grandfather, Thakorlal Bhagwandas Panchal

(May 11th, 1938 - August 10th, 2017)

Acknowledgments

Firstly, I would like to thank my major professor, Dr. Tianming Liu, for his avid guidance, motivation, and supervision. Your knowledge and resources have helped me at every step in my research and thesis. I would also like to thank Dr. Hamid Arabnia and Dr. WenZhan Song for being part of my thesis committee. It is Dr. Arabnia who inflamed my interest in the image processing and computer vision domain of computer science when I studied Image Processing class taught by him.

I would like to thank Vyom Srivastava for helping me in the initial stages of the thesis project set-up. I would also like to thank Wenyu Li, a visiting research scientist from China, for simplifying some of the concepts in TensorFlow that accelerated my work. I would like to acknowledge all the faculty members at the Department of Computer Science at the University of Georgia.

I would like to thank Vinamra Jain for all the valuable advice and encouragement. I would like to thank Anurag Joshi, Nirav Kuber, Chirag Jain, Lovina Dmello, Pooja Panchal, and Juhi Furia for their unfading support.

Lastly, but the most important, I am grateful to my family: my parents and my grandma for showing immense support during the hard times and motivating me to keep moving forward, no matter what!

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Saliency information is defined as valuable or noticeable information. Mapping such information is an important topic in the field of computer vision, especially when focusing on medical images. In a colored image, some pixels have a unique color in comparison to its background, which in turn forms different kinds of object or shapes. When these objects or shapes attract the focus of the human eyes, the image is considered to contain areas of saliency. The same logic works while plotting a saliency map on a grayscale image, where a sudden difference in the level of gray is the area of saliency. The goal of mapping the saliency on an image is to ease the analysis process and hence save time. In this chapter, we will discuss the problem description in Chapter 1.1 and the motivation behind the thesis project in Chapter 1.2 and Chapter 1.3 will have the thesis outline.

1.1 Problem Description

Application of machine learning in the field of medicine has improved health care immensely over the past 5 years. According to the Pharmaceuticals Research and Manufacturers of America (PhRMA), over 120 medicines are developed to treat lungs cancer and the rate of deaths are expected to be more than 158,000 because missing out on interpreting certain diseases in the chest [1]. Reading a chest x-ray, even for a professional radiologist, is often a daunting task. Challenges include apices, bone abnormalities, the cardiac shadow or a reactive effusion, diaphragm, division of the lungs, etc. Very less is explored in ways of examining or interpreting chest x-rays, especially when you consider the above-mentioned obstacles. Other hindrances to interpretation also include the incorrect amount of exposure and an adequate position of the human subject while capturing the image. Hence, considering these aspects, interpretation becomes a demanding and challenging task.

High-resolution Computed Tomography (CT scan) and Magnetic Resonance Imaging (MRI) like technologies are used for clarity of the results of the radiologist's examination. These modern techniques are time-consuming and very expensive to set-up. Top IT companies including Google, Amazon, and researchers across the medical image community are trying to find an efficient and effective way of interpreting chest x-rays to combat the downfalls of CT and MRI scans. By focusing on using technology in the interpretation process, researchers have made new discoveries, which in turn are currently being used to further technological advancements. Using the eye-tracking sensor technology to monitor how a radiologist interprets a chest x-ray remains unexplored.

1.2 Motivation

Despite the advancement in medical imaging technologies, reading and interpreting the chest images is still a difficult and challenging task [2]. Recording the gaze data from the interpretation and using that data to plot the saliency on the chest x-ray image is an untouched field. Motivated by prediction of eye fixation on the various resolution of images [10], we introduce the saliency mapping using the visual gaze data captured by the eye-tracking device. This method of using an eye-tracking sensor makes advancement in the traditional technique of analyzing and interpreting chest x-rays, which is currently being used by the medical industry. To implement the needed changes, the installation of the eye-tracking sensor connected to the local computer would be the only hardware required. This method will only need the installation of the eye-tracking sensor which would be connected to the local computer and no other hardware would be required.

Radiologists view the x-ray images with their naked eyes for most of the time. During the time of interpretation, an eye-tracker sensor can capture the gaze data of the radiologist. To model a neural network technique for mapping the important area of interest on the chest x-ray is the basic idea behind this thesis.

1.3 Thesis Outline

In Chapter 2 we discuss the Background of the thesis, the basics of the neural networks (Chapter 2.1) and convolutional neural network (Chapter 2.2) and also understanding the TensorFlow framework (Chapter 2.3), which is highly used for building a neural network and also Keras (Chapter 2.4). Following, in chapter 3 we discuss the Dataset we used. This involves the chest x-ray dataset ChestX-ray8 (Chapter 3.1), details of capturing the eye tracker's data (Chapter 3.2) and plotting said data onto the chest x-ray to generate the dataset needed for passing the images to the neural network. In chapter 4, we discuss the architecture of our Convolutional Neural Network (CNN) along with the description of the layers (Chapter 4.1 and Chapter 4.2) and how they contribute to the entire network. The results and experimentations (Chapter 4.3) along with the summary (Chapter 4.4) are discussed later. Finally, we discuss the future work and conclude the thesis document.

Chapter 2

Background

Artificial neural networks or just neural networks are one of the important paradigms of the machine learning approach that is gaining popularity quickly and helping researchers and computer scientists around the world to make the machine learning field robust. In this chapter, we shall discuss the details of the technology behind this thesis project. In section 2.1, we shall discuss details of the deep learning technology and some of its application. Then, in section 2.2, we look at a more focused approach of the Deep Learning called the Deep Neural Networks or Neural Networks for short, and focus more on a specific type of neural networks called the Convolutional Neural Networks or CNNs. To implement these CNNs requires a tool, called TensorFlow. Section 2.3 shall include the details about TensorFlow which is an open-sourced programming library available across various platforms. When working with the TensorFlow for CNNs, there is a well-focused library, called Keras. This library is also open-sourced and has capabilities to dataflow programs on top of TensorFlow in Python programming language. Details and explanation of how we use this Keras library in our project to implement the CNN are covered in section 2.4.

2.1 Deep Learning

A buzzword today, 'Deep Learning' was originally was termed in the 1940's as a technology that could simulate the neural system of the human brain. Deep learning is a subset of machine learning that is capable of solving complex problems. Machine learning is the ability of a program to understand and infer future predictions.

2.1.1 Machine Learning

Machine learning is a type of artificial intelligence technique that is engineered from a biological inspiration to give machines, robots, or computer programs to make decisions and act according to specific inputs. The goal here is for a machine or a program to mimic the human mind. Machine learning and artificial intelligence (AI) are both the terms that are used interchangeably by computer scientists and industry experts. It is because of machine learning many labor jobs are being replaced by technology that can perform the tasks with intelligence. By becoming more reliable for efficiency and effectiveness at work, many industries are using machine learning to complete tasks. Knowledge representation, reasoning, and abstract thinking are the core topics of machine learning. Past experience or data is an essential element of the machine learning field. The data is used in conjunction with machine learning algorithms to make decisions and analysis based on the input data. on what they learn from the input data. When data comes into play, machine learning can be classified into two categories: supervised learning and unsupervised learning.

- Supervised Learning: The machine learning task that generates the output decision based on the input, which can be given in the form of labeled data, is called as supervised learning
- Unsupervised Learning: The machine learning task that generates the output decision without having any previous reference or labeled data and has to learn on-the-go is called as unsupervised learning

Deep learning with neural networks falls into the category of the supervised learning. To generate a decision or an output in the deep learning environment, the input data is passed into a structure called as multi-layered artificial neural networks. Each of the layers in the network contains a node which will hold the data. At every layer, there is a calculation based on what the program is aimed to achieve. Once the task at one layer is completed the data is passed over to the next layer to perform another last. This is why it is called a multi-layered.

At the calculations in the last layer, we get the output of the network. One such example of the multi-layered artificial neural network is as shown in figure 2.1 :

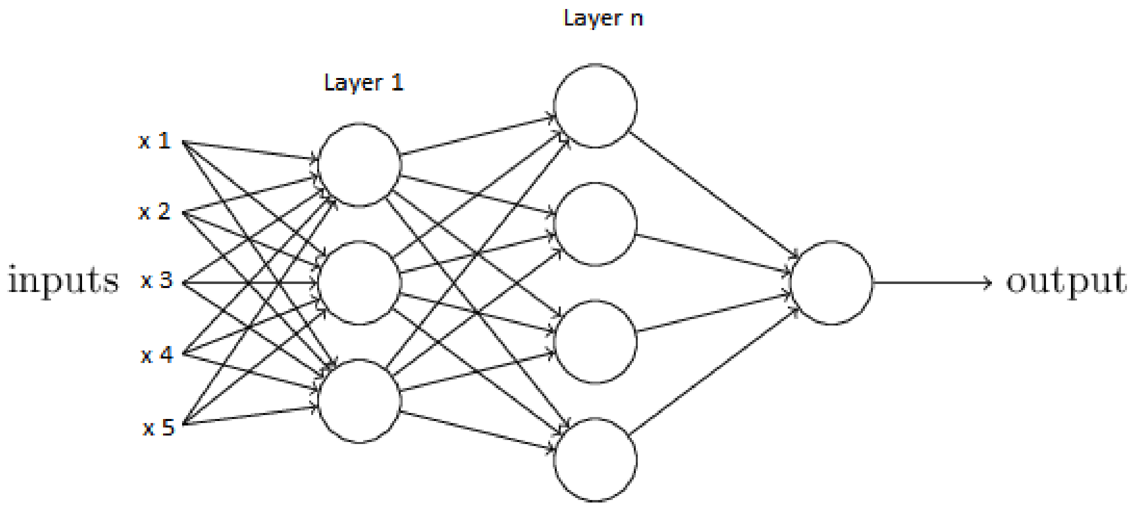


Figure 2.1: Sample of a multi-layered artificial neural network.

[1]

In the above figure, we see the inputs x1,x2,.., x5) are passed into the network. At layer 1 there is some intended task that is performed and the output of this layer is fed as the input of the next layer and this process continues based on the number of layers. After the final layer, we get the output. Some of the applications of deep learning include language recognition, autonomous vehicle, image captioning, image recognition and many more.

[1]Part of the pictorial representation in figure 2.1 is used from 'Concept of Deep Learning type artificial intelligence' from website: https://stonewashersjournal.com/2016/01/29/deeplearningchess/

2.2 Neural Networks

As discussed a bit in the above section, a neural network is a multi-layered structure that has nodes representing or storing data. Since then term artificial intelligence is giving the ability to machine for making decisions and to mimic the brain structure we have the artificial neural networks, the nodes in the artificial neural networks just act like the neurons of the brain. Biologically, the neurons are the cells in the brain that can receive any incoming information, process this information based on function or task and then transmit or pass the information to other neurons in the network. Since we are trying to copy this functionality of the brain, the node also is doing the similar kind of task in the artificial neural network. The neurons of the brain are called the nodes in the Neural Network. These nodes take in data in all different formats, process this data based on defined calculations and produce an output that can be also of any format. This data in the output can be sent to the nodes in the following layer. The figure below shows the above process. Here, there are n number of inputs that are represented as X1, X2, ... , Xn. There are weights assigned to every input value. These weights are used while performing the calculation. The calculation function is pre-defined for a particular layer. In the figure, we see the pre-defined calculation function is the summation. Hence, here all the weight will be multiplied by their respective input values and then will be summed up to give the final output. This is demonstrated in figure 2.2 :

[2]

Here, we only have represented 1 neuron in the layer. There are n number of neurons per layer that are used for the programming purpose for assigning the calculation per layer. Now, there can be n number of layers in the network as it is a multi-layered architecture. Hence, in a neural network, there can be 'n' number of neurons/nodes (defined with calculation) for 'z' number of layers in a network will construct a neural network. A neural network is called a deep neural network only when there is more than 1 layer in the architecture. This

[2]Concept of the diagram in figure 2.2 adopted from 'Single Neuron' website: https://blogs.cornell.edu/info2040/2015/09/08/neural-networks-and-machine-learning/

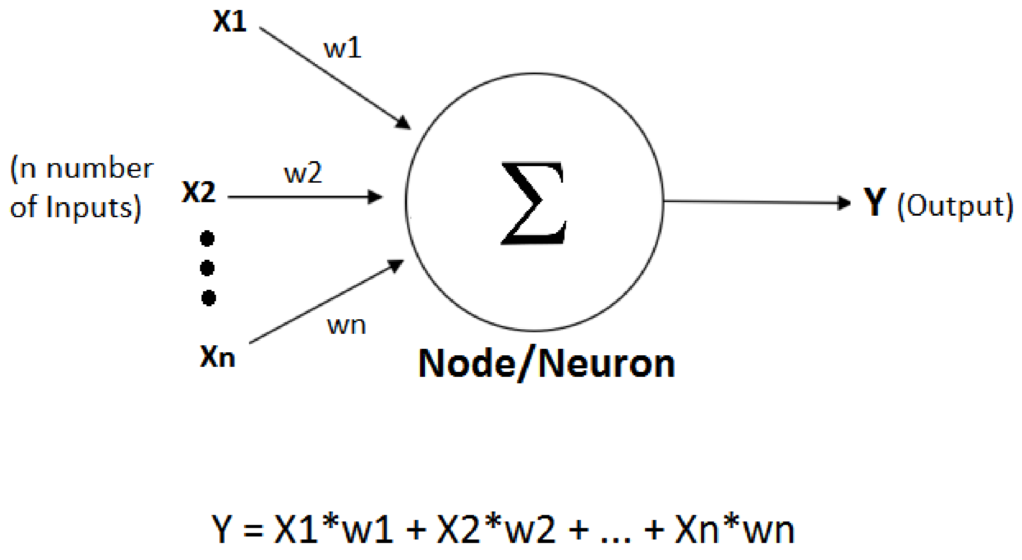


Figure 2.2: Node or the Neuron presentation with a set of input, set of weights assigned to each of the input, the pro-defined calculation function, and the output that is generated after the calculation.

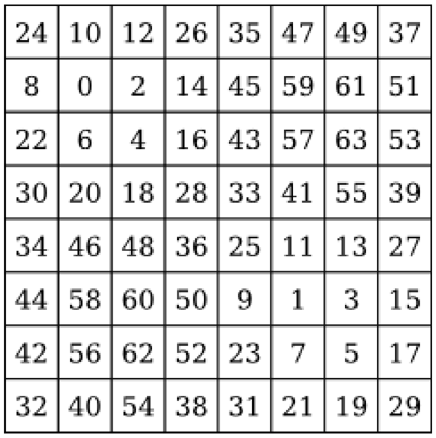
entire neural network needs to be programmed with calculations for each layer for achieving the goal, which is to generate an output that is a decision. In section 2.3, we will see how we write the program to construct such an architecture.

2.2.1 Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) is a neural network with the same multi-layered architecture where images at every layer go through convolution process. A convolutional process is a process where there is a window of weights that moves along the image from left to right and runs top to bottom for the calculation operation that is performed at every layer. The window of weights is called a kernel. There are any one of 3 convolution operations possible in a CNN: a convolutional layer, a pooling layer, and an activation layer. A brief explanation of each of the layers is given below:

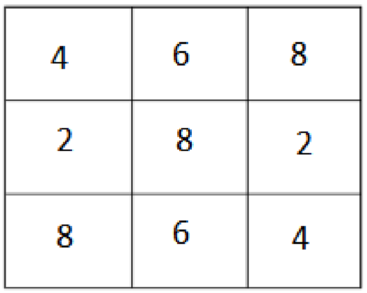
- Convolution layer: In a convolution layer, there is a grayscale image of size 8 x 8 and there is a kernel of 3 x 3 dimension. This window will move over the image starting from the very top left corner and will slide from left to right and then top to bottom and end the calculation window in the bottom right corner.

  The process of the window moving from left to right and top to bottom starting from the top-left corner is called convolution. Hence, a neural network is called a CNN when there is a convolution operation involved at every layer. A sample of this is as shown in the figure below:



24	10	12	26	35	47	49	37
8	0	2	14	45	59	61	51
22	6	4	16	43	57	63	53
30	20	18	28	33	41	55	39
34	46	48	36	25	11	13	27
44	58	60	50	9	1	3	15
42	56	62	52	23	7	5	17
32	40	54	38	31	21	19	29

8 x 8 matrix of grayscale image matrix



4	6	8
2	8	2
8	6	4

3 x 3 window of kernal

Figure 2.3: Figure (on left) shows a grayscale image of size 8 x 8 dimension This is represented in a matrix format. The window (on right) is the kernal of 3 x 3 size with will be used to convolve over the image.

- Pooling Layer: A pooling layer also works on convoluting principle but here in the kernel, there are no weights. From the kernel, there are values chosen or calculated based on the type of pooling layer. There are 3 types of pooling layers: Max-pooling, Min-pooling and the average pooling. In max-pooling, the highest of all the values within the kernel window is the output. In the min-pooling, the minimum of all the value within the kernel window is the output. And in the average pooling, the average or the mean of all the values within the kernel window is the output. Max-pooling and average pooling highly used pooling layers that are used in a CNN implementation. Figure 2.4 shown below demonstrates how the pooling layer works:

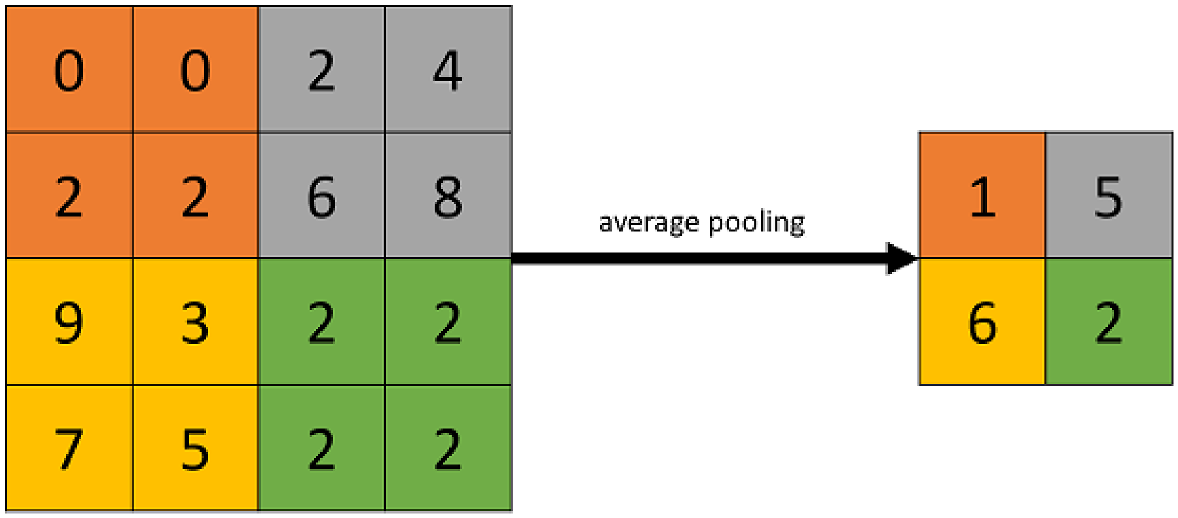


Figure 2.4: Average pooling represented in a color representation for better understanding

[3]

The goal of pooling layer is to reduce the input size of the data for the next layer. As we see in the above figure, the original size of the image was 8 x 8 which is reduced to 2 x 2 after the pooling layer. Here average pooling is implemented.

- Activation layer: Same as the above 2 layers, the activation layer also works on convoluting principle. Here, each value in the window is passed through a function for calculating the output value. Some of the functions are the binary step, ArcTan function or the Tanh function, Rectified Linear Unit (ReLU) function and many more.

[3]The image in the pictorial representation of the average-pooling is used from https://www.kaggle.com/questions-and-answers/59502 . The image on the page was edited for representing application related topic.

ReLU and Tanh are the most commonly used activation functions used in the industrial application. Some of the functions and formula are as listed below:

$$\text{Binary Function} \qquad f(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x >= 0 \end{cases} \tag{2.1}$$

$$\text{Tanh Function} \qquad f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.2}$$

$$\text{Rectified Linear Unit (ReLU)} \qquad f(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x >= 0 \end{cases} \tag{2.3}$$

$$\text{Paramertic ReLU} \qquad f(x) = \begin{cases} \alpha x, & \text{if } x < 0 \\ x, & \text{if } x >= 0 \end{cases} \tag{2.4}$$

$$\text{SoftPlus} \qquad f(x) = \log_e(1 + e^x) \tag{2.5}$$

Using any of the above-mentioned function, the pixel value can be squashed down anywhere between -1 and +1. And this value can be passed to the next layer for further processing.

A Convolutional Neural Network (CNN) works quite well based on the application as the pixels and the data of the image are operated, calculated and manipulated in a pattern. This keeps the relationship between the neighboring pixels as the kernel (window) is sliding and considering neighboring pixels during the layer calculation.

2.3 TensorFlow

Tensorflow is an open source developed by Google and Open Source primarily for deep learning application development, where one can write computer programs which is a state-of-the-art in the artificial intelligence and machine learning field. Tensorflow is mostly used for data flow programming models. The nodes that are present in the CNNs can be programmed using TensorFlow library and all the calculations and math operations that are needed to be performed in a CNN at every layer can be programmed in TensorFlow. Each data coming to the node that is passed to different layers are in a data array format. These data arrays are called as tensors. These data arrays are multidimensional Not only limited for CNN application, TensorFlow can be used for traditional machine learning models and algorithms. Tensorflow can handle calculation and any numerical operations on a large scale. The execution and working of a TensorFlow model are in the form of a graph as there are nodes connected in multi-layered architecture which has edges and nodes interconnected [4]. Never intended to be used for developing CNNs, TensorFlow was developed for internal use at Google Labs. Looking at the ability to work on data nodes and data-flow models, researchers saw a large potential in this library and hence it was publicly made available. The early version of TensorFlow was not so handy to write and develop a CNN or even a complex machine learning application but with improvements in the library, TensorFlow was made available with changes and syntax those similar to Python programming language. TensorFlow was released with C++ and Python front-end for developing CNN models. The popularity increased and demand for using TensorFlow increased over several fields and currently it is available on more than 10 different programming languages for developing CNN models. Even light version for low capacity memory devices was also released recently. TensorFlow helps in building our CNN model that would take input as chest x-ray images which will be multidimensional arrays. Image data will be converted into tensors and these tensors will be used for any mathematical operations that will be performed on the image at several layers of the model. First, the image will be converted into arrays and then passed into the model. The visual of the above-mentioned procedure is as shown in figure 2.5.

[4]

[4]Part of the pictorial representation in figure 2.5 is used from website: http://www.astroml.org/book_figures/appendix/fig_neural_network.html

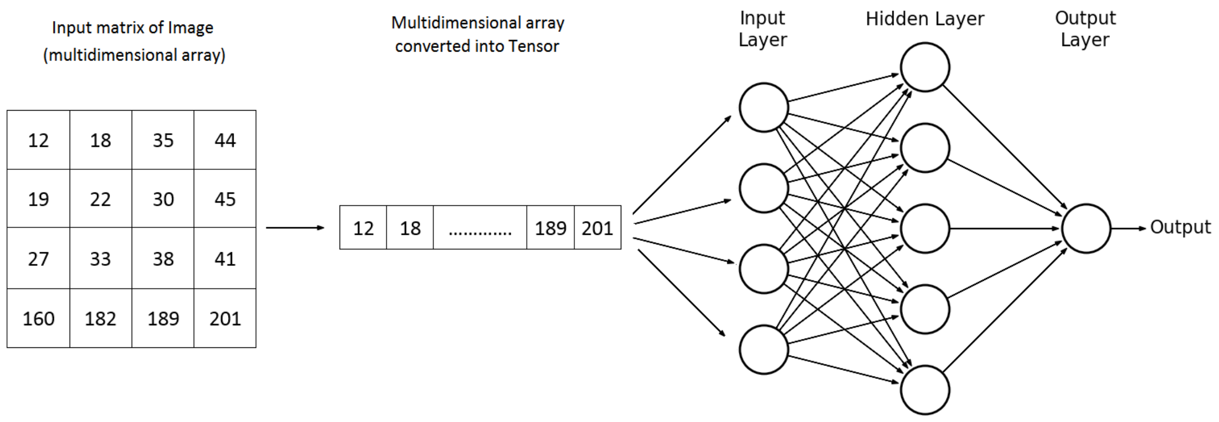


Figure 2.5: Demonstration of an image of multidimensional array converted into a tensor and then passed to the input layer of the CNN. Calculations are every node is forwarded to the following layer. At the output layer we get the output of CNN.

Recently, there are libraries that were developed that can be used on top of the TensorFlow, which makes writing programs even easier for a CNN model. One such library is called Keras. We use Keras for the implementation of all the layers of our CNN model.

2.4 Keras

Like TensorFlow, Keras too is an open-source library, developed in python programming language, that works on top of TensorFlow for developing and programming CNNs. Keras is an Application Programming Interface (API) that wraps multiple frameworks and is used by the researchers and developers, for developing a neural network where there is a need for faster calculations and experimentations. One of the most important features of Keras is that it runs with no lagging on Graphical Processing Units (GPUs) as well as CPUs. Keras gained popularity in the research community, especially for deep learning as it minimizes the number of actions that are required in the data-flow programming. Keras reduces the efforts of writing, debugging and building custom modules and layers inside CNN that include lots of calculation. Modeling a CNN in Keras doesn't require to manually input the calculation procedure. Applications, where Keras is highly used, include Computer Vision, Speech Recognition, Image Classification and many more. The procedure of building a CNN is as explained here: First, you define a network (a CNN) with what layers would constitute to complete a network model. Then, we can compile the network and check for the data flow within layers and maintain consistency of the network. Then we fir the network with the type of input data that we expect for the network to receive. Once we know the data that would fit into the network, we can evaluate the network for error ratio, loss of data and accuracy. Then we can use this network for the actual data and use it to make predictions.

Now, once the CNN is modeled, in the very first layer, there has to be input provided to the model. input data needs to be in a format that can be passed into the network. In the next chapter, we see how we create our dataset.

Chapter 3

Dataset

In this chapter, we will focus on the details of the dataset used for the project. The dataset of x-ray images, ‘ChestX-ray8’, is used throughout the project, and discussed further in Chapter 3.1. Images of this dataset are modified as per our requirements to pass the images into the neural network. Firstly, the visual gaze data is plotted onto the images. Detailed information of this gaze data is discussed in Chapter 3.2 and 3.3, where we talk about the eye-tracker sensor used in this project and how the data collected is mapped on the image, respectively. In Chapter 3.4 we discuss the generation the final stage of image processing that we performed for this project. Ultimately, these are the images that we pass to the neural network.

3.1 ChestX-ray8

Before continuing further, we will first establish what the ‘ChestX-ray8’ data set is. ‘ChestX-ray8’ is a dataset which contains 108,948 images, are only the frontal views captured from 32,717 different patients with advanced chest diseases. In September 2017, the National Institute of Health (NIH) Clinical Center, released this dataset to the public in an attempt to further research into this growing field. All the images in this dataset are .png file format. The x-rays were captured on patients with advanced chest diseases. X-rays in the dataset are diagnosed with at least one of the following eight diseases: atelectasis, cardiomegaly, effusion, infiltration, mass, nodule, pneumonia, and pneumothorax. This information is extracted by using a number of Natural Language Processing (NLP) techniques for detecting the keywords that are assigned to each of the images when diagnosed by an expert radiologist.

Figure 3.1 shown below is a set of chest x-ray images from the 'ChestX-ray8' dataset:

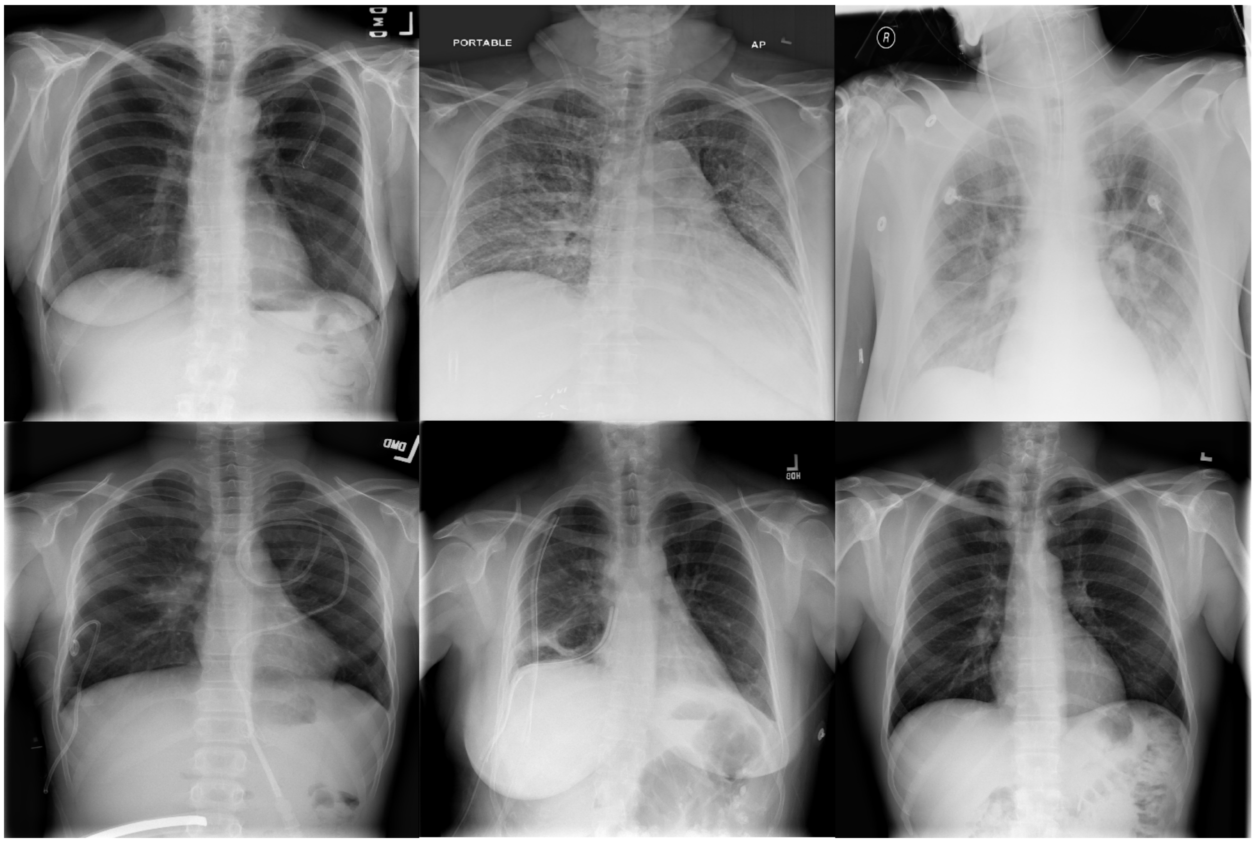


Figure 3.1: X-ray Images of diagnosed patients from ChestX-ray8 Dataset.
(Dataset source [5])

As mentioned earlier, there are almost 109,000 images of chest x-rays in the dataset. There are some images of x-rays with inadequate exposure, some with improper positioning of patients, some chest x-rays with implantable cardiac pacemakers and implantable cardioverter defibrillators (ICDs). All such x-ray images with faults are discarded and we choose 500 good quality x-ray images from the entire dataset. The figure below shows some of the discarded x-rays images with inappropriate quality.

Going from left to right and top to bottom: we see x-ray with the cardiac pacemaker, irregular mass in the body, improper positioning of the patient, inadequate information with the exposure problem, amount of exposure problem and the last one has improper positioning, too many cables of electrical instruments with inadequate exposure, all together. All such images are discarded.

Given below is a set of chest x-ray images that are discarded from our experimentations:

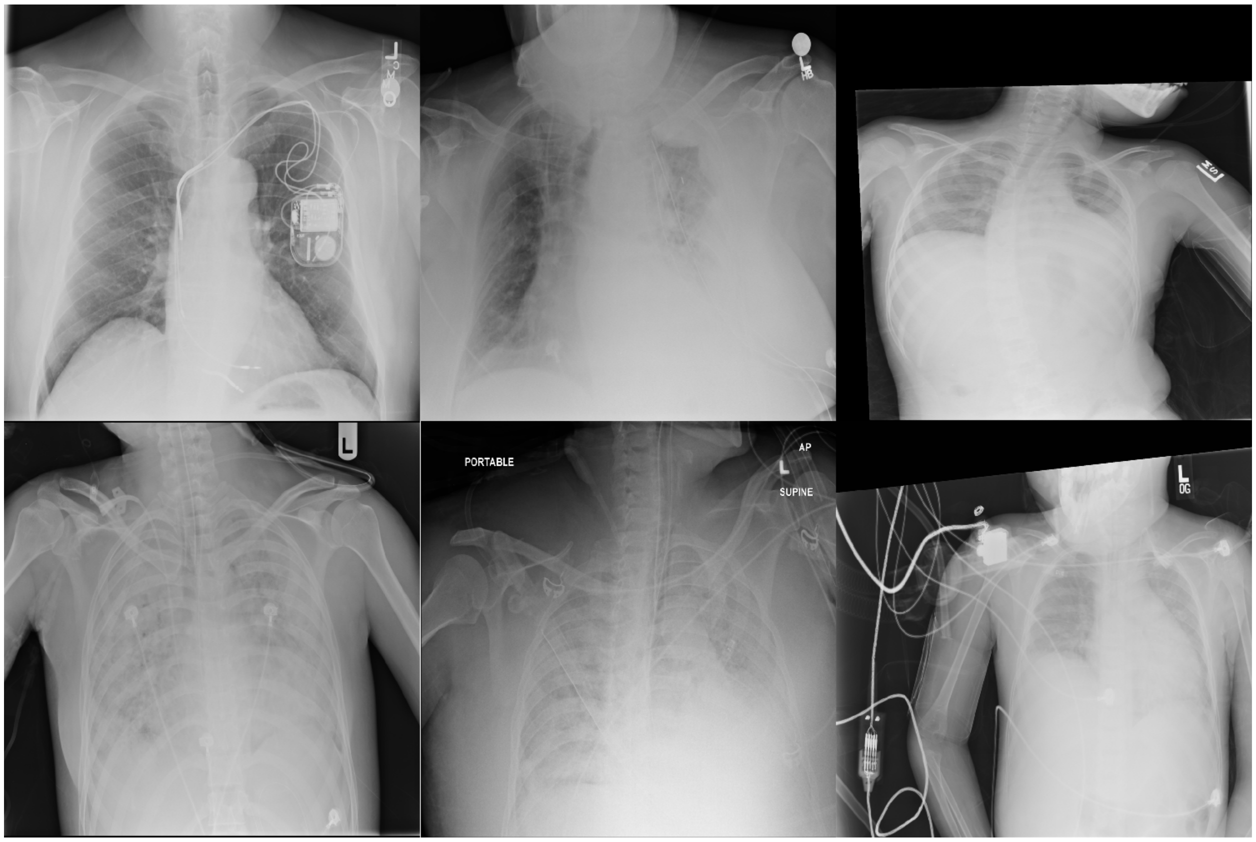


Figure 3.2: X-ray Images that are discarded because of inappropriate quality.
(Dataset source [5])

In the above image, (left to right, top to bottom) we see x-rays with a cardiac pacemaker, an irregular mass in the body, improper positioning of the patient, inadequate information with the exposure issues, amount of exposure complications, and lastly improper positioning with too many cables of electrical instruments. All such images are discarded.

3.2 Tobii Eye Tracker

Eyes are the most important sense in a human body. By understanding what is being viewed and perceived by the human eye, we can further discern human behavior and decision making.

For the success of this project, the need for a sensor that can monitor and track the movements of the eyes is crucial. Being a pertinent part of this thesis, the use of such a sensor is to capture movements of the eyes as radiologists while he or she is interpreting the chest x-ray. Tobii Pro X2-30 is the eye-tracker device that we use for the project, displayed in figure 2.2 as shown below :

Figure 3.3: Tobii Pro X2-30 eye-tracking device used for collecting the gaze point data for experimentation.

[1]

The device is approximately 18.5 cm in length, weighs about 250 grams and is connected to the computer via a USB 2.0 connector, which captures data at 30 Hz. It has a dual-camera system, which has 2 cameras that capture accurate and precise data every time. One of the most important features of this device is that the device takes no time to recover from any blinks by responding immediately after an eye-blink. On top of these advantages, the Tobii device is compatible with other portable devices such as laptop computers, tablets, and cell phones.

As mentioned earlier, the purpose of using this eye-tracker device is to capture eye movements of the radiologist when he or she is examining a chest x-ray. The Tobii sensor captures information such as the timestamp, pupil diameter, gaze point, and much more. For our project, only the gaze point of both eyes, captured by the sensor, is used in further stages.

[1]This picture is originally published by tobii pro on their website: https://www.tobiipro.com/product-listing/tobii-pro-x2-30/

The device captures the data even if there is a head movement or is there is in ambient light. The subject, a radiologist in our case, has to be between 30cm and 50cm from the sensor. It is recommended to install this device at the bottom of the viewing screen. Figure 3.4 shown below represents the arrangement of the screen and the device with the distance between the eyes and the sensor.

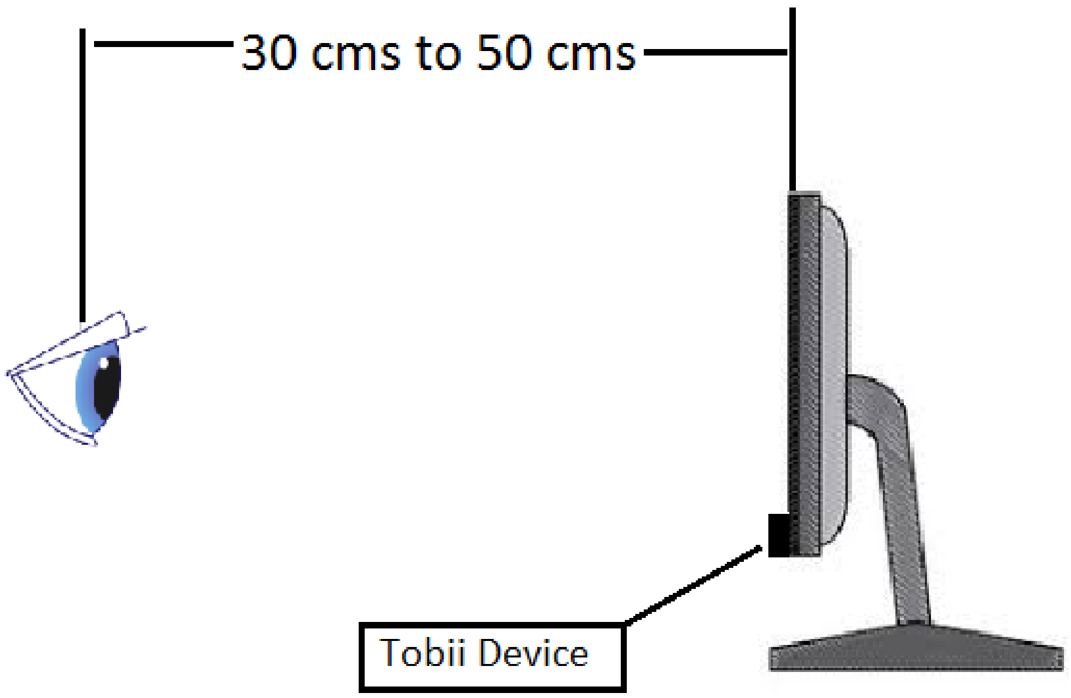


Figure 3.4: Subject, a radiologist, looking at the screen with distance of 30 cms to 50 cms from the screen. The Tobii device is mounted on the monitor.

For this project and experimentation, we use a similar set-up where the display screen is a 17” monitor with the sensor installed at the bottom of the monitor. Now, the gaze point data that is captured by the sensor is of the Active Display Coordinate System (ADCS). All the data is then mapped into the 2D Cartesian coordinate system. These points are then plotted onto the display screen. For plotting, we only consider the display area and not the frame of the display device for calculation. This display region is called the Active Display Area (ADA). Since we use the coordinate system here, there is a point of origin on the ADA, the monitor. For this project, the upper left corner of the monitor is the origin of the ADCS. This point will represent as (0,0). The lower right corner is the end-point of the ADCS and hence it is denoted by (1,1). So, the gaze point data that will be captured by the device will be a point between the origin and the end-point, i.e., between (0,0) and (1,1) with floating point values.

Since the gaze point will be captured of both the left eye and the right eye, there will be two separate gaze points that will be returned from the sensor. Each of these points is then multiplied with the resolution of the screen in their respective direction. For instance, the sensor recorded at gaze point of the left eye as (0.6511 , 0.3251) and (0.5942 , 0.2893) for the right eye. The screen that we are using is of 1920x1200 resolution, which means 1920 pixels on the X-axis and 1200 pixels on the Y-axis. Now to find the gaze point on the screen for the left eye; the x-axis value of the left eye is multiplied with 1920: Left eye (x-axis) = 0.6511 x 1920 = 1250.1, the y-axis value of the left eye is multiplied with 1200: Left eye (y-axis) = 0.3251 x 1200 = 390.1. Similarly, for the right eye; Right eye (x-axis) = 0.5942 x 1920 = 1137, Right eye (y-axis) = 0.2893 x 1200 = 347.16. We then round-off the number to the nearest whole number. Hence, the gaze point on the screen is (1250,390) for the left eye and (1137,347) for the right eye.

For our experimentation, we record the gaze data for 60 seconds. This means that the data is recorded by the Tobii device for 60 seconds when the radiologist is examining the chest x-ray. As mentioned earlier the data is recorded at 30 Hz, we collect about 1700 to 1800 instances of data. The number is not exactly 1800 (as 60 sec times 30 Hz is 1800 instances) due to some eye-blinks. The Tobii device is very robust at collecting data after an eye-blink, but there is a loss of data during the time the eye was closed for the blink. Hence, we collect about 1700 to 1800 instances of data for each x-ray examination. The above-mentioned example of the gaze point was just one such instance during the examination.

3.3 Gaze Data Mapping

Gaze data mapping is one of the pivotal element of this project. This mapped data will be used as labels in the Convolutional Neural Network (CNN). This section involves more math than the one in the previous chapter.

We studied, in the previous section, how a gaze point is calculated from the Active Display Coordinate System (ADCS) values on a display with the resolution 1920 x 1200. Our x-ray images, from the ChestX-ray8 dataset, are of size 1024 x 1024. When an x-ray image is loaded on the full-screen mode on the monitor, the image will not reach to the boundaries as the display resolution is larger than the size of the image. Hence the image is loaded on the center of the screen with black borders. Any data that will be captured by the eye-tracker device cannot by directly plotted onto the image. The visual presentation of this case is presented in the figure below:

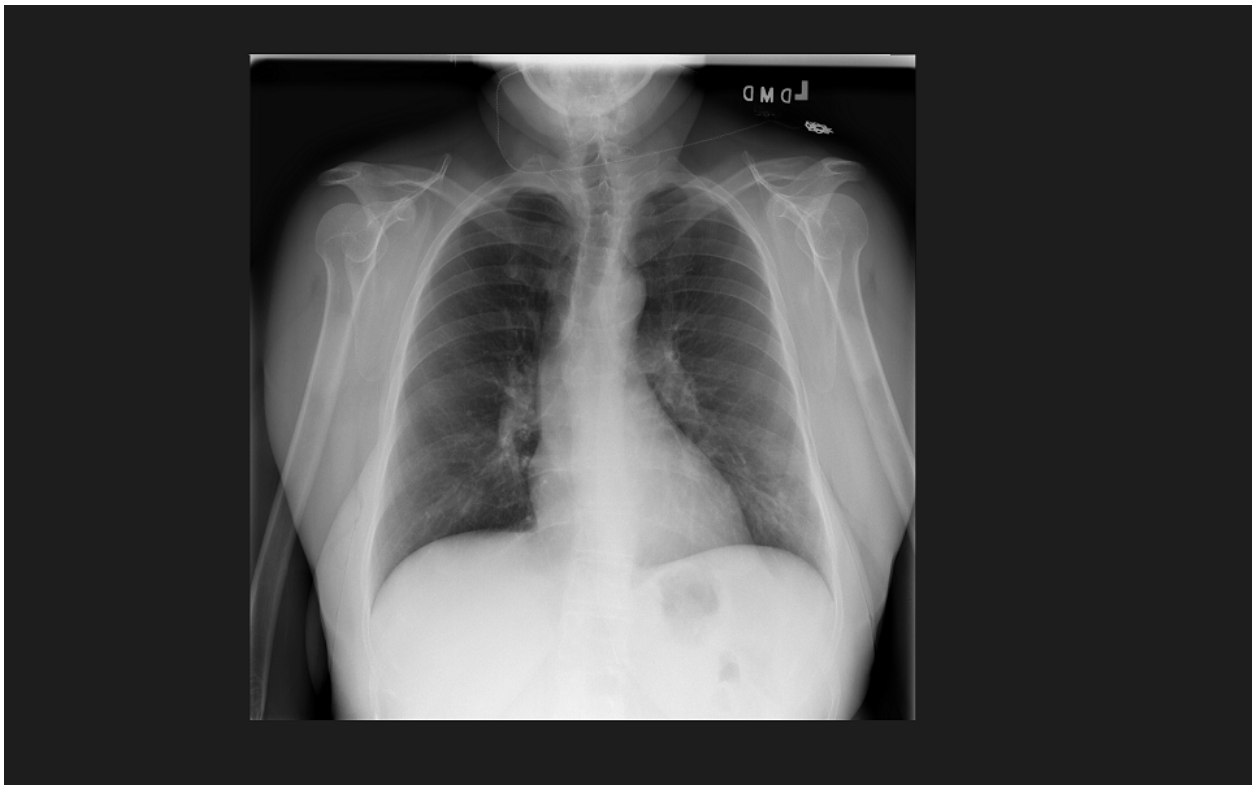


Figure 3.5: Chest x-ray image when displayed on full screen mode with gray border as the resolution of the image is less than the resolution of the screen. (x-ray image source [5])

Hence, we cannot just map the plotting data on the image for the gaze points. Meaning, the gaze point of the pixel on the images will not be the same as the gaze point of the pixel on the screen. Therefore, we create a black image of size same as the x-ray image, meaning 0 intensity in the grayscale value of the image, we plot this 0 onto an array of 1024x1024, which gives us the black image. This will act as a canvas on which we will plot the gaze point.

Now, we match the origin (the top left corner) of the black image to the origin of the screen. Therefore, any gaze point that is captured is re-calculated to overlap onto the chest x-ray image. We move the x-axis value of the gaze point in the left direction and the y-axis value of the gaze point in the upper direction which will plot the gaze point onto the black image. This will overlap the gaze point data to the chest x-ray image. As the full screen displaying the x-ray will be in the center, the screen of 1920 pixels resolution will be divided into 1024 pixels of chest x-ray and 896 pixels of the border. As the border is in both the direction, there are 448 pixels on the left and 448 pixels on the right. Like we said earlier, we only need to move in the left direction, so only 448 value will be subtracted from the x-axis value of the gaze point. And similarly, for the y-axis value of the gaze point needs to move in the up direction. The x-ray displaying screen will have 1200 pixels resolution on the y-axis which will be divided into 1024 pixels of the chest x-ray image and 176 pixels of the border. And since the border is in the upper and the lower direction of the x-ray image, there are 88 pixels of the border on the top and 88 pixels on the bottom of the screen. Like we did for the x-axis value, we will subtract 88 from the y-axis value of the gaze point. Let the gaze point be called (Gx,Gy) which cal be represented by the following formula:

$$Gx = |(ADCS\ of\ x\ axis \times 1920) - 448| \qquad Gy = |(ADCS\ of\ y\ axis \times 1200) - 88|$$

Table 3.1: Plotting the gaze point onto the image. The column 'Gaze point in ADCS' is the original value that is recorded by the Tobii device, 'Gaze point on screen' is the value we get after multiplying with the screen resolution(our case 1920x1200), 'Gaze point on x-ray image' is the overlapping gaze point on the x-ray image.

	Gaze point in ADCS	**Gaze point on screen** (1920 x 1200)	**Gaze point on x-ray image** (1024 x 1024)
Left Eye	(0.5816 , 0.4192)	(1116 , 504)	(668 , 413)
Right Eye	(0.5911 , 0.4164)	(1134 , 500)	(686 , 412)

The table below is another example of the gaze point mapped onto the chest x-ray image. Here, the gaze point recorded by the Tobii device in the ADCS format for the left eye is (0.5816,0.4192) and one of the right eye is (0.5911,0.4164). The 'Gaze point on screen' column is the multiplication of the screen resolution 1920x1200 in their respective direction with the original gaze point value in ADCS, which we calculate to be (1116 , 504) and (1134 , 500) of the left eye and the right eye respectively. The 'Gaze point on x-ray image' column in the table is the gaze point on the x-ray that we get after subtracting 448 and 88 in x-axis value and y-axis value of the gaze-point respectively, which is calculated as (668 , 413) of the left eye and (686 , 412) of the right eye.

If we plot the gaze point that we calculated in the table onto the x-ray screen, it would look something like the image shown in figure 3.6:

The above gaze point, when plotted on the chest x-ray image size black image, looks like the image shown in figure 3.7:

This black image with the all the gaze points that is generated, we call it the Gaze Map and follow the same procedure for all 200 images that are selected for the experimentation. We use these gaze maps as labels for the Convolutional Neural Network (CNN).

Figure 3.6: Plotted gaze point on the screen calculated in the above table.

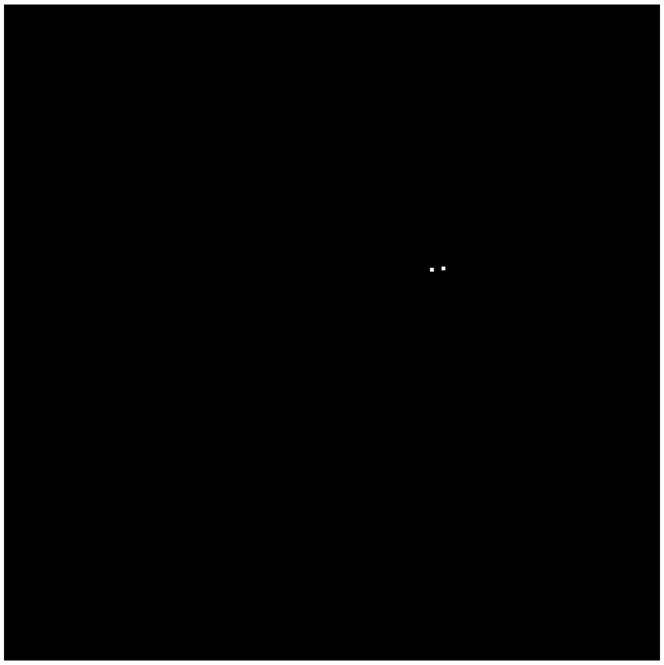


Figure 3.7: plotted gaze point on the black image calculated in the above table.

Chapter 4

Deep SaliencyNet

Plotting prominent information on an image is called saliency mapping. In other words, highlighting information on an image that has an importance, which may include humans or object, is saliency mapping [6]. This terminology has been in the computer vision field for over 25 years, but since 2012 when Borji and Itti presented their 'state-of-the-art in Visual Attention Modeling', [7] that compared all the existing models that were used for mapping attention models onto images. All the models in this paper [7] has successful derivation theoretically but no proof of how the models would work in the real world when hundreds and thousands of images will be passed for mapping attention. There are no practical application of these models and no test to verify if these methods can be even used in the field of computer vision. But Judd et al. [8] came up with a breakthrough in saliency prediction. Their method was proven to predict the fixation of human eye of where the eyes were gazing at any given image.

Many in the research community worked on the problem of predicting the fixation of a human eye and hence predict the saliency information on any image. Saliency mapping was first used in the biological field, where predicting a fungal growth was the ultimate task. project in the biological field where predicting a fungal growth was the ultimate task. In recent years, deep learning, especially with artificial neural networks, are used as the tool for mapping fixation and saliency information. The rich feature evaluation of the Convolutional Neural Networks (CNNs) has proven records in image segmentation and image classification problems. Using those rich features of the CNNs, Wang and Peng et al. [9] developed the chest x-ray classification of eight different types of chest diseases. The ultimate goal of this was to enable high precision Computer Aided Diagnostics (CAD). This method lacks the main element, human involvement in learning to classify and then predict the diseases. N. Liu et al. [10] modeled a CNN architecture called Multi-resolution CNN (MrCNN) that helps predict eye fixation. This network is developed to classify different regions of a single image. Here, an image is first transformed into multiple smaller resolutions of important parts of the same image based on what the eye would look at by instinct. These multiple images are fed into the CNN at the same time. MrCNN helps in predicting the fixation of eyes on multiple scales of resolution of a single image. At the same time, this model only focuses on a definite area of an image rather than considering a complete entity at one time. There is a loss of information even before the model begins its execution. Hence in this thesis project, there is human involvement as a supervision element and doesn't completely rely on the dataset images. The data that we recorded, using the eye-tracker device, acts as a label and the images from ChestX-ray8 dataset will be the input for the model. Also, this data (chest x-ray images) are fed into the CNN without manipulation of the dataset, while using the global context of the data for better prediction of the saliency map and eye fixation.

4.1 Architecture

Inspired by the small kernel used by the VGG Group of Oxford in their VGG-16 model [11], we use the same concept of using small kernel sizes at every layer of the network. Since we are using these kernels with the convolution principle, of sliding the kernel onto the image from left to right and top to bottom, our network architecture is called fully convolutional neural network. With the use of the small kernels, 3 x 3 and 5 x 5 kernels, every pixel of the image is worked upon, allowing for the classification of pixels as important in saliency of not. we are working on every pixel of the image for predicting is that particular pixel is classified as important in the saliency or not. Also, having more layers in the network helps to extract detailed features of the images /citegoog, which led us to create a network with 23 layers. and hence this network has 23 layers. All CNN layers such as the convolution layer, pooling layer, and the activation layer are used either individually or in a combination with one another. (Combinations of layers are further explained with inception block.) (meaning: some layers are a combination of the 3 CNN layers, explained detailed with inception block). Here is the sketch of the CNN architecture shown in figure 4.1 :

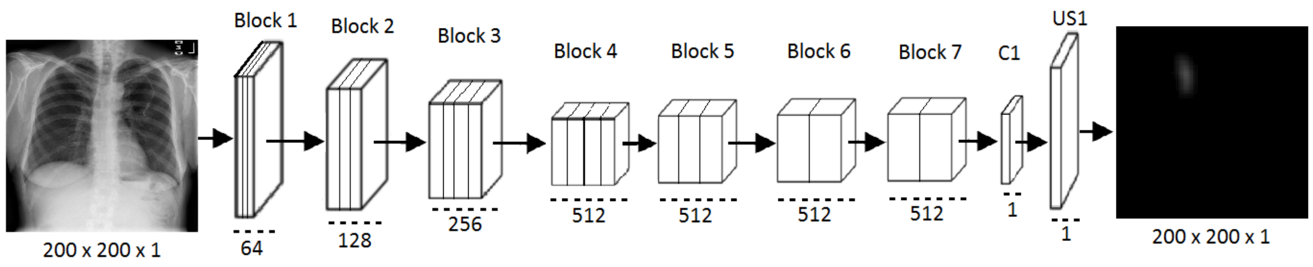


Figure 4.1: Architecture of the CNN where input is a chest x-ray image (200x200x1) is fed as input to the network and we derive a saliency map in the output as an image (200x200x1).

Like we discussed, we feed in the chest x-ray from the ChestX-Ray8 dataset into the CNN. All these images are of 200 x 200 grayscale images. As a limitation on computational resources, we transform the original dimensionality of the images to a resolution that would be faster for execution. During the execution, each image is taken 1-by-1 and this image goes through a series of convolution operations over every layer of the network in every block of the architecture. Considering the principle of multiple convolution layers in a row from VGG16 architecture [11] , our CNN architecture has a similar principle in every block (Block 1 to Block 7 in the above figure). All the filters used in the convolution layer inside the blocks has the kernel of size 3 x 3 or 5 x 5 and no high resolution.

Block 1 and Block 2 has the same series of layers, starting with two convolution layers and then a pooling layer. In the entire architecture, we use max-pooling to downscale the image. However, the number of filters or the so-called dimensionality of the output space changes in both the blocks. As represented in figure 4.1, block 1 has 64 filters which means that all the layers in Block 1 use 64 filters. Similarly, for Block 2 is 128 denoted in the architecture, which means that Block 2 used 128 filters. The layered structure of Block 1 and Block 2 is as presented in figure 4.2.

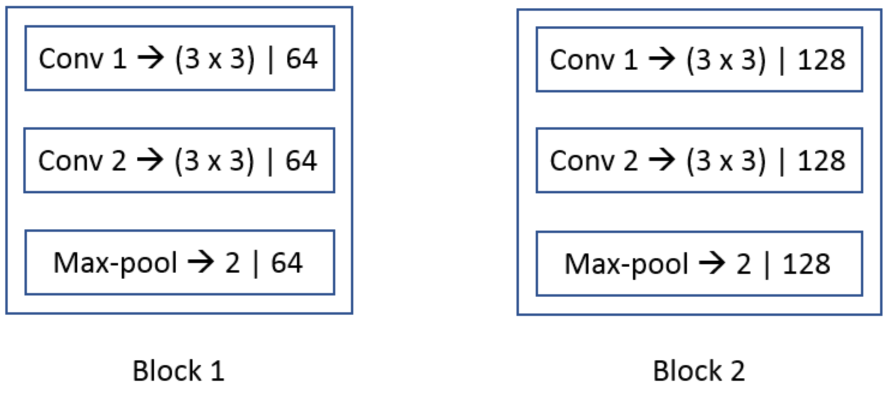


Figure 4.2: CNN layers inside Block 1 (left) and Block 2 (right).

Here, after Conv 1 layer is executed, the dimensionality of the output changes and the next layer receives the new dimensionality data as input. In our case, the image of 200 x 200 x 1 will be changes to 200 x 200 x 64 and repeatedly continues. Again for Block 2, the dimensionality will change as there will be 128 filters in every layer of Block 2. The max-pooling layer reduces the size of the image. Max-pooling of stride 2 is executed. Hence, the dimensionality of the image becomes exactly half the size of the height and the width.

In Block 3 and Block 4, we have not two but three convolution layers, with kernel 3 x 3. In Block 3, all the layers have 256 filters and that in Block 4 have 512 filters. Figure 4.3 shows the architecture of Block 3 and Block 4 with all the layers.

Notice here, the depth or the number of kernels increases in every block. Gradually increasing the depth helps to explore some rich features of the data, as it is one of the basic principles of the deep neural networks.

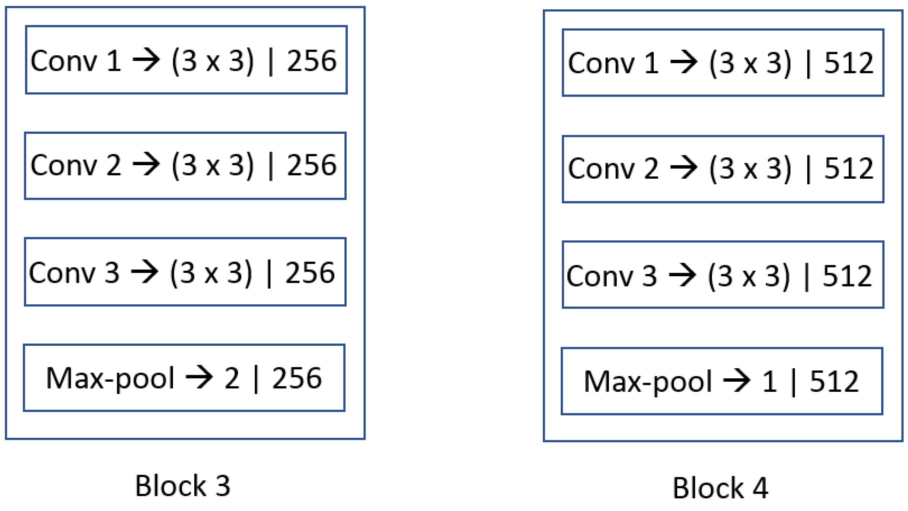


Figure 4.3: CNN layers inside Block 3 (left) and Block 4 (right).

In Block 5, all three layers are convolution layers of kernel size 3 x 3, same as all the convolution layers of Block 1 to Block 4. The number of kernels remains the same Block 5 as that in Block 4. The depth is 512 in every layer of this block. But, the kernel here is changes to a different shape. A 3 x 3 kernel is enlarged and there are holes introduced in the kernel. By introducing holes, the 3 x 3 kernel enlarges to become a 5 x 5 kernel and introduces spaces or what is called holes between the values in a kernel. To consider a wider field of view to capture more features with the same computational cost is the goal behind introducing holes in the kernel. This introduction of holes in the kernel is called the hole algorithm or the atrous algorithm'. Papandreou and Chen et al. [13] used a similar concept with their segmentation problem. The window or the kernel size used in Block 5 is as depicted in figure 4.4 as shown below :

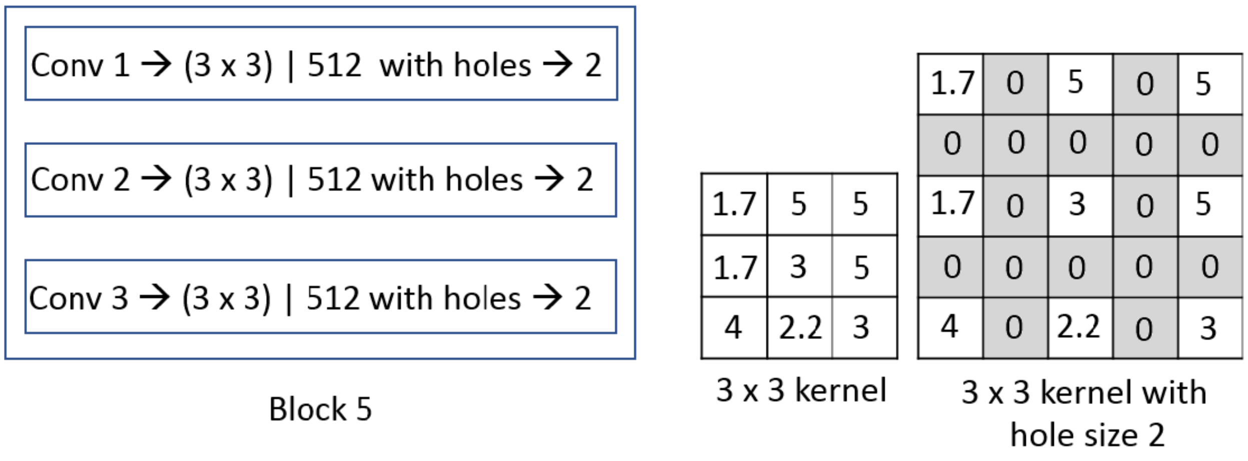

Conv 1 → (3 x 3) | 512 with holes → 2
Conv 2 → (3 x 3) | 512 with holes → 2
Conv 3 → (3 x 3) | 512 with holes → 2
Block 5
1.7 5 5
1.7 3 5
4 2.2 3
3 x 3 kernel
1.7 0 5 0 5
0 0 0 0 0
1.7 0 3 0 5
0 0 0 0 0
4 0 2.2 0 3
3 x 3 kernel with hole size 2

Figure 4.4: CNN layers inside Block 5 (left), a 3 x 3 kernel (middle), a 3 x 3 kernel with hole size 2 becoming a 5 x 5 kernel (right)

While executing this network, the pixel intensity is used for calculation with the kernel weight. The weight of these windows or the kernels in the convolution layers across all the blocks is loaded from the weights of the well-trained VGG-16 model. The weights of the kernel of this VGG-16 model are derived after training more than 1.2 million images [11].

Following this, Block 6 in the architecture, is the inception block. The concept of an inception block was first introduced by Szegedy, Liu et al. [12]. Convolution operation with different kernel size helps to extract different types of features. Using all of them together is the main and using them all together is the main ideology behind the use of inception layer [14]. So, in the inception block, the data is fed in from the previous layers into four separate channels, with each channel having different sizes of dimensionality and kernels as well. Figure 4.5 shows the inception layer. As we can see, all the channels have a varied number of filters, each responsible to extract unique feature than another. Once we have all the outputs from the individual channels, the next step is to combine or concatenate all the data, which will later be used in the next layer. later in the network, we can use this concatenated result to pass it to the next layer. In Block 6, there are two such inception layers and they both operate with the same policy. The output of Block 6 has the depth of 512.

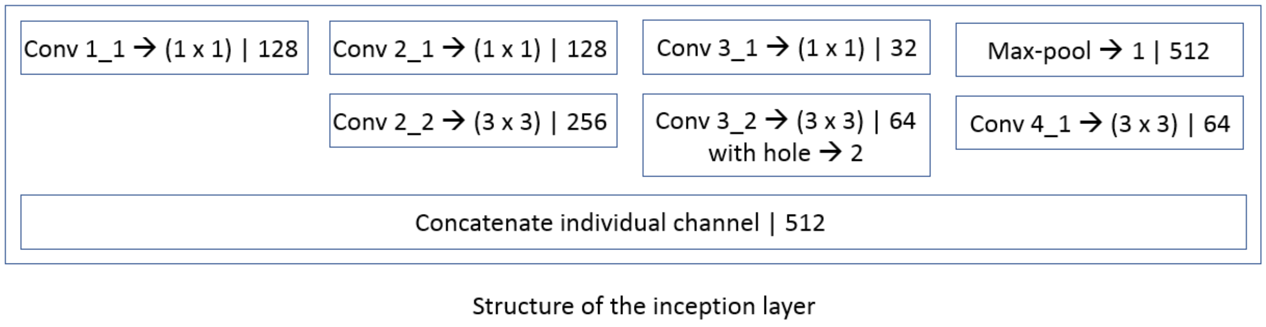


Figure 4.5: Architecture of the inception layer inside Block 6 with 4 different channels responsible for extracting features with multiple dimensionality of the kernel.

The output of Block 6 goes into Block 7, which has three convolution layers. and inside this block, there are 3 convolution layers. All three layers have a kernel size of 5 x 5 and we introduce holes of size 6 in the kernel. Now the actual size of the kernel that will convolve becomes 25 x 25. Now, since the beginning of the execution of the network, there has been max-pooling layers used multiple times. And as we know the length and the breadth of the reduced, by the time the image data reaches block 6, its length and breadth transform to 25 x 25. Hence the convolution operation basically occurs just 1 time as the kernel size would also become 25 x 25 with holes. Significant improvement was noticed in the result when such layers are introduced into the network. It is important to capture features at a global context in the saliency mapping problem, this type of layer helps the same principle. There are 2 such layers in Block 7, each with a depth of 512. This block is illustrated in figure 4.6.

Next in the CNN architecture is C1 (refer to figure 4.1 for architecture). Here there is just one layer of convolution operation, again using the weights of VGG-16. Here there is only 1 filter used, which means the dimensionality of the data changes from 512 in the previous layer to just 1 in the layer. Since our goal is to generate a 2D image at the output, the depth of this needs to be 1, as it would be a grayscale image. So we only use 1 filter in the convolution layer C1 to generate a 2D data matrix.

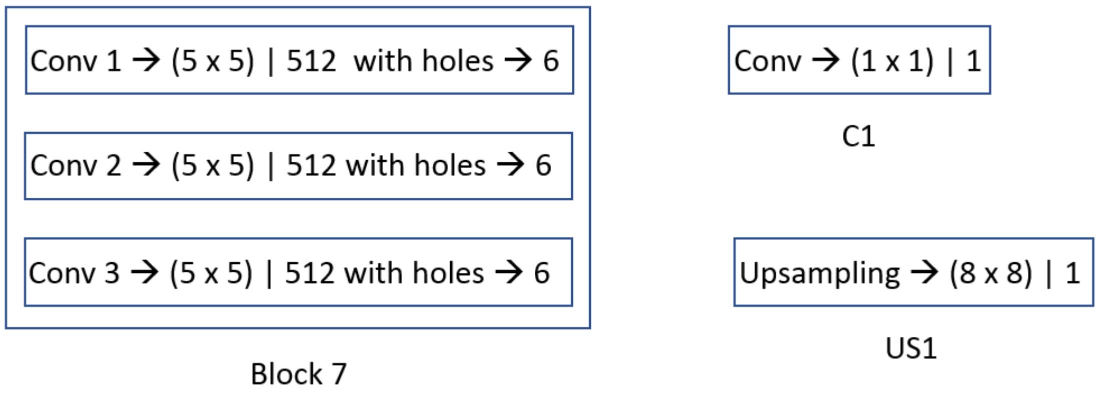

Conv 1 → (5 x 5) | 512 with holes → 6
Conv 2 → (5 x 5) | 512 with holes → 6
Conv 3 → (5 x 5) | 512 with holes → 6
Block 7
Conv → (1 x 1) | 1
C1
Upsampling → (8 x 8) | 1
US1

Figure 4.6: CNN layers inside Block 7 (left), C1 layer (top-right), US1 layer (bottom-right).

As mentioned before, there are multiple uses of the max-pooling layer in the network which results in the lowering of the length and breadth of the data. Since max-pool layer, individually with stride 2, are used trice in the network, the length and the breadth degraded 8 times the original input size. Since our input was 200 x 200, the output of the C1 layer reduces to 25 x 25. And since the expected result of the saliency map is the same as the original input size, we upsample the output of the C1 layer. In the US1 layer, the input of 25 x 25 x 1 is upsampled to 200 x 200 x 1. Notice, there is no change in the depth of the data in the upsampling process. There are different upsampling techniques that are used in the computer vision field. Bicubic interpolation is one of the best upsampling technique because it doesn't just copy the pixel data multiple times in the grid manner like in the linear interpolation. Bicubic interpolation uses 16 nearest neighbors of a pixel in the Lagrange polynomial [15]. This makes it computationally heavy, but produces a better upsampling than any other interpolation technique. Using the inbuilt bicubic interpolation in the upsampling process of Keras library, the height and the width are reacquired to match the original input size.

Classification not being the sole purpose of this model, there has to be a loss function and an optimizer defined for a successful execution of the model. Mean Squared Error (MSE) is the loss function used during the model compilation. Optimizer, being the essential argument for model compilation, has to be declared. Stochastic Gradient Descent (SGD) optimizer is used for this model and the momentum is set to be 0.9. Usually, the momentum ranges between 0.9 and 0.95. However, since we use SGD, 0.9 is our momentum value. Both, the loss function and the optimizer are required for the model compilation of any CNN, and both are available through the Keras library.

4.2 Results

This architecture was tested on the images from the ChestX-Ray8 dataset that was discussed in the previous chapter. The weights of the kernels used in the convolution layers are initialized from the weights of the VGG-16 model. The model is executed with 15 epochs; which means the model will execute from start to the end 15 different times. In each epoch, the accuracy of the results increases. The figure below shows the result of the experimentation with all the above-mentioned properties. In the figure, the leftmost column contains the images of the chest x-ray from the dataset, the middle column is the gaze map that we plot from the gaze-point data that is collected from the eye-tracker device and the rightmost column is the set of images that are resultant of the 15 epochs execution of the CNN.

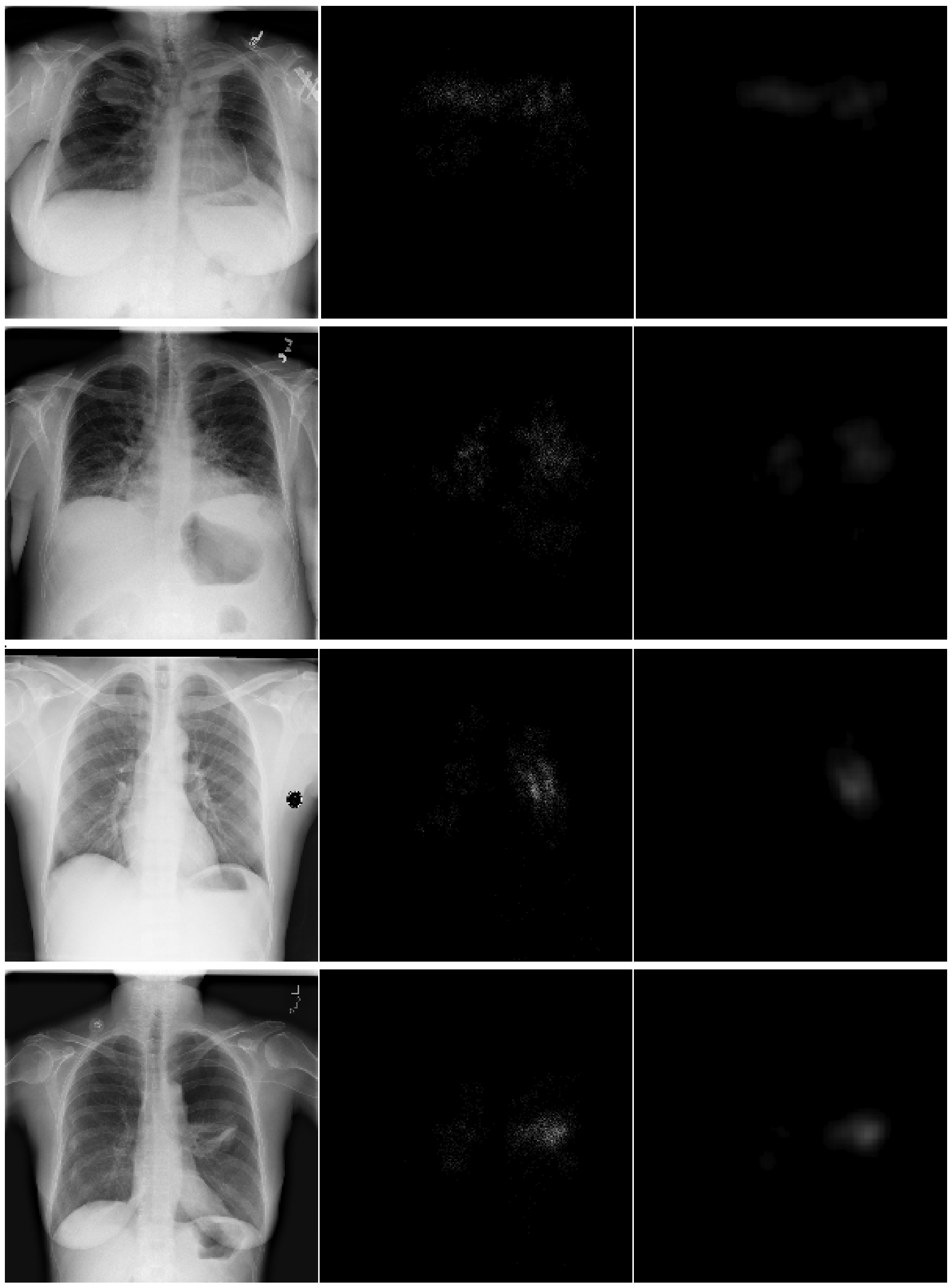


Figure 4.7: Sample 1 : Chest x-ray images (leftmost) from ChestX-ray8 dataset, Gaze points used as ground truth (middle), and the saliency map generated after the execution (rightmost)

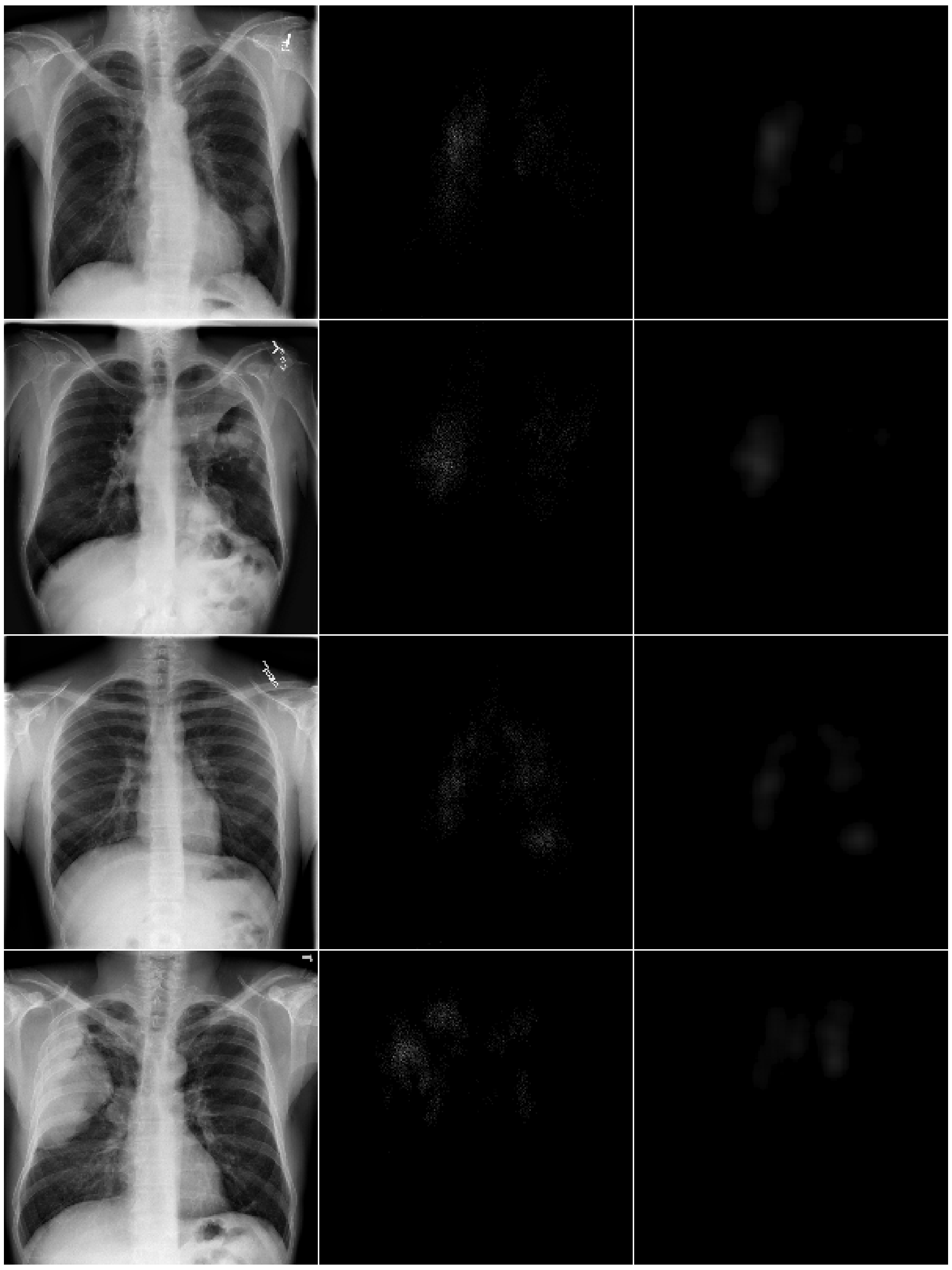


Figure 4.8: Sample 2 : Chest x-ray images (leftmost) from ChestX-ray8 dataset, Gaze points used as ground truth (middle), and the saliency map generated after the execution (rightmost)

4.3 Discussion

Considering multiple resolutions of images and addressing them into the model was the ultimate goal of MrCNN [10]. Our approach addresses the multiple resolutions of the chest x-ray image in a different way. In the Inception block, the kernel of different sizes (resolutions) is used in the individual channel which explores the features in a multi-resolution context. Hence we compare our approach to the MrCNN model. Some of the comparison elements include the following:

- Earth Moving Distance (EMD)

  EMD is the evaluated distance between two multi-dimensional matrices. For the image processing field, EMD is useful to calculate how far off is the second matrix from the first one. Comparison of the probability distribution results in the EMD evaluation. We compare the saliency map generated by the model to the ground truth generated by the eye-tracker device.

- Correlation Coefficient

  Linear Correlation Coefficient between the Saliency map (S) and the Groundtruth (G) is known as the Correlation Coefficient (CC). The value of CCis between -1 and +1. The resultant value of +1 is the best match between the images. The mathematical formula for calculating CC is as given below:

$$Correlation\,Coefficient(S,G) = \frac{Covariance(S,G)}{Standard\ Deviation\ of\ S \times Standard\ Deviation\ of\ G}$$

The table below shows the comparison of these elements:

Table 4.1: Comparing Earth Mover's Distance (EMD) and the Correlation Coefficient (CC) of Multi-resolution Convolutional Neural Network (MrCNN) to our proposed model.

	Earth Mover's Distance (EMD)	**Correlation Coefficient (CC)**
MrCNN	4.31	0.57
Our Model	2.08	0.72

In the above table, we see the difference in the evaluation results. For Earth Moving Distance (EMD), the minimum the distance, the better are the results. Hence our model outlasts the existing Multiresolution CNN model by a huge margin. This proves that our model is closer to the ground truth. Also, consider the Correlation Coefficient (CC), the resultant of the calculation is 1 for an exact match between 2 images that the CC is calculated. Our model wins by some margin and concludes that it is better at plotting saliency. Considering how critical it is, when it comes to the medical application of such project, our model proves to be better at a product level.

Chapter 5

Future work and Conclusion

In this thesis project, we propose a Convolutional Neural Network (CNN) that will be able to plot out saliency in bio-medical images, especially for chest x-rays. For our project, we use chest x-ray images from the Chest-Xray8 image dataset, provided by the National Institute of Health (NIH) Clinical Center, that contains more than 108,000 frontal views of the chest x-ray. For our model compilation, we select the best of quality images for experimentation. For the label, we plot the gaze points of where a radiologist is focusing on the chest x-ray while examining. We record this data using the eye-tracker device. The gaze points that are captured by the device are plotted on a black image and thus used as the label for the model. The idea of inception layer, from the famous GoogleNet, is used for capturing features at multiple resolutions of the images. We use this inception layer concept and architect an innovative deep CNN. After this model executes successfully, saliency mapped images are generated by this model. We place this saliency map output beside the ground truth to see the quality of the result.

Since we have good results for our project, a future extension to this project can be the prediction of the saliency map for other types of bio-medical images such as Magnetic Resonance Imaging (MRI) and Computed Tomography Scanned Medical images (CT Scan) and try predicting the diseases based on the saliency map. Since there is active research in making the bio-medical field for automating a variety of tasks, such projects will take the bio-medical field to the next level.

Another possible extension of this thesis project is to integrate such CNN models and software into the hardware. Reducing the number of equipment and hardware in the medical facilities is one goal that can be achieved with such a project.

Bibliography

[1] American Cancer Society, AACR Cancer Progress Report. 'Medicine in Development for Cancer', 2015 Report. Website: http://phrma-docs.phrma.org/sites/default/files/pdf/oncology-report-2015.pdf

[2] Louke Delrue, Robert Gosselin, Bart Ilsen, An Van Landeghem, Johan de Mey, Philippe Duyck. 'Difficulties in the Interpretation of Chest Radiography', September 2018.

[3] Ana M. Franco Watkins, Joseph G. Johnson. 'Running head: Eye-Tracking Methods for Decision Research', Judgm. Decis. Mak. 6, 740749, Est: 2011.

[4] Karlijn Willems. 'TensorFlow Tutorial For Beginners'. Website: https://www.datacamp.com/community/tutorials/tensorflow-tutorial

[5] 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. IEEE CVPR 2017

[6] Rui Zhao, Wanli Ouwang, Hongsheng Li, Xiaogang Wang. 'Saliency Detection by Multi-Context Deep Learning', in CVPR 2015.

[7] Ali Borji (Member of IEEE), Laurent Itti (Member of IEEE). 'State-of-the-Art in Visual Attention Modeling', IEEE Transactions on pattern analysis and machine learning intelligence, VOL. 35, NO. 1, January 2013.

[8] Tilke Judd, Krista Ehinger, Fredo Durand, Antonio Torralba. 'Learning to Predict Where Humans Look', MIT Computer Science Artificial Intelligence Laboratory (CSAIL) and MIT Brain and Cognitive Sciences.

[9] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, Ronald M. Summers. 'ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases', 14 Dec 2017.

[10] Nian Liu, Junwei Han, Tianming Liu, Xuelong Li. 'Learning to Predict Eye Fixations via Multiresolution Convolutional Neural Networks', IEEE transactions on Neural Networks and Learning Systems, 2016.

[11] Karen Simonyan, Andrew Zisserman. 'Very Deep Convolutional Networks for Large-scale Image Recognition', ICLR 2015.

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. 'Going Deeper with Convolutions', CVPR 2015.

[13] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan L. Yuille. 'Semantic Image Segmentation with Deep Convolutional Nets and Fully Connestedd CRFs', ICLR 2015.

[14] Dorin Ionescu, Julien Despois on 'How does the Inception module work in GoogLeNet deep architecture?'. Website: https://www.quora.com/How-does-the-Inception-module-work-in-GoogLeNet-deep-architecture.

[15] Prachi R Rajarapollu, Vijay R Mankar. 'Bicubic Interpolation Algorithm Implementation for Image Appearance Enhancement', ISSN : 2229-4333 (Print), IJCST Vol. 8, Issue 2, April - June 2017.