Weakly Supervised Classification and Localization of Thorax Diseases on X-Ray images

by

Alinstein Jose

B.Tech, Mahatma Gandhi University, 2013 - 2017

A Project Report Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Engineering

in the
Department of Electrical and Computer Engineering
University of Victoria

UNIVERSITY OF VICTORIA

Spring 2021

Approval

Name:	**Alinstein Jose**
Degree:	**Master of Engineering (Electrical Engineering)**
Title:	**Weakly Supervised Classification and Localization of Thorax Diseases on X-Ray images**
Examining Committee:	Dr. Wu-Sheng Lu, Supervisor Department of Electrical and Computer Engineering University of Victoria Dr. Hong-Chuan Yang, Member Department of Electrical and Computer Engineering University of Victoria

Abstract

Deep learning has added a vast improvement to the already rapidly developing field of computer vision. The ability to solve many computer vision problems like image classification, object detection, localization, and tracking has grown significantly in terms of performance and efficiency in recent years when the field is equipped with state-of-the-art deep learning techniques. In this project, we focus on classification and localization for medical imaging. Specifically, in the first part of the project, we develop a deep neural network that predicts disease in Chest *X*-ray images. Recent advancements in transfer learning suggest using the pre-trained model and fine-tuning since it is shown to produce state-of-the-art results. Therefore, in this project, we use both the learning of a model from scratch and transfer learning to classify chest *X*-ray images. In the second part of the project, we tackle the unavailability of dense annotation of region-level bounding boxes of diseases in *X*-ray images and propose a method to locate disease regions in *X*-ray images by constructing a weakly supervised localization method.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Wu-Sheng Lu, whose patience, valuable guidance, and suggestions have immensely helped me throughout this incredible journey of my MEng studies and related research, which culminated in the completion of this project report. I am also grateful to Dr. Hong-Chuan Yang for his time serving as the supervisory committee member and for his insightful comments and encouragement.

I would also like to thank MakerMax and Matrox Electronics for providing me with an opportunity to work as a Machine Learning Engineer intern, where I got practical experience in the field.

Also, I thank my friends Derrell D'Souza, Bharath Madela, and Amy Sun at the University of Victoria for being an integral part of this journey. I am also grateful to Anju Rama Krishnan for all her constant support and for inspiring me every day.

Finally, I would like to thank my family: my parents and my brothers for supporting me emotionally throughout writing this report and my life in general.

Chapter 1 Introduction

Practically everyone has taken an *X*-ray at some point in their lives. *X*-ray has been one of the most effective non-invasive methods to help doctors detect and diagnose diseases, and chest *X*-ray is one of the standard radiological examinations for diagnosing and screening lung-related diseases in medicine. Chest *X*-ray imaging uses a small amount of radiation to produce pictures of the chest and identify abnormalities or diseases in airways, blood vessels, bone, heart, and lungs [49].

According to an article published in 2016, cardiothoracic and pulmonary abnormalities constitute the leading causes of illness, mortality, and health service use all over the world [46]. In the United States, according to the American Lung Association, the most number of fatalities among various types of cancers is caused by lung cancer, in both men and women, with more than 33 million Americans suffering from chronic lung diseases [47]. Due to its effectiveness in characterizing and detecting abnormalities in the cardiothoracic and pulmonary cavity, chest *X*-ray remains the most commonly requested and conducted radiological examination. It is also widely used in lung cancer prevention and screening. On the other hand, chest radiography requires timely reporting of potential findings and diagnosis of disease in the *X*-ray images. Unfortunately, timely reporting of every X-ray image is not always possible due to heavy workload in many large healthcare centers or lack of experienced radiologists in less developed regions. In summary, automated, fast, and reliable disease detection based on chest *X*-rays has been a critical step in radiology workflow.

The recent breakthrough in deep learning-based computer-vision algorithms has brought about new hope for efficient automated disease identification from *X*-ray images. In effect, many digital signal processing challenges, like image colorization, classification, segmentation and detection, can now be addressed with a deep learning framework. More specifically, a class of deep learning architectures known as convolutional neural networks (CNNs) has shown the ability to considerably improving prediction and classification performance as long as a sufficient amount of reliable data and powerful computing resources are available. Problems that were considered untractable are now being solved with high accuracy. Deep learning has also gained popularity in

medical imaging for disease detection and segmentation. There is a significant amount of research works on machine learning techniques that are aimed at medical applications such as detection and classification of the pulmonary nodule in CT images [50], automated pancreas segmentation [51], and cell image segmentation and tracking [52], to name a few.

In this project, we develop a deep convolutional neural network (DCNN) that predicts eight common thoracic diseases found in chest *X*-ray images and spatially locates the diseases in the chest *X*-ray image. The model is built within a so-called weakly supervised multi-label classification and disease localization framework that is trained and validated using a large-scale chest *X*-ray dataset, ChestX-ray8 [1], that contains 112,120 *X*-ray images with disease labels. A challenge we face with the dataset is that it is loosely labelled or contains noise in the classification labels because labels for classifications are created from *X*-ray reports using natural language processing. Besides, disease regions in the X-ray images are unavailable in the dataset. We understand that region-level bounding boxes labeling of diseases is not practical because of the dataset's huge size and, also, accurate labelling of disease region in X-ray image requires skilled and experienced radiologists, which is not feasible for the time being and will be very expensive if done so. Moreover, this is the primary reason motivating us to develop a weakly supervised disease location method to detect disease regions that require only image labels for classification.

1.1 Contributions of This Project

The contributions of this report are as follows:

- We have proposed a DCNN that can classify one or more diseases in an *X*-ray image and trained several DCNNs like ResNet and DenseNet on the Chest X-ray8 dataset [1]. The performance of the deep learning models loaded with and without pre-trained is compared and examined separately, and varying input image resolution effects are studied. An important observation made from our numerical simulations was the substantial performance gains when pre-trained weights and higher-resolution input images are employed.
- We have developed a binary DCNN classifier to identify any abnormality (*X*-rays containing any disease considered abnormal) in Chest *X*-ray images.

- We have built a weakly supervised localization framework to locate the region of diseases in the X-ray image. Specifically, weakly supervised localization methods such as Class Activation Map, GradCAM, and GradCAM++ are implemented and compared.

1.2 Organization of the Report

This report is organized as follows:

Chapter 1 provides a brief introduction to the topics to be covered and gives an overview of the project and Chest X-ray8 dataset [1] for *X*-ray classification and disease localization.

Chapter 2 provides a brief history and introduction to deep learning. We also discuss DCNNs that will be used in the rest of the report.

Chapter 3 presents various weakly supervised localization methods that can locate diseases in *X*-ray images.

Chapter 4 presents a new method for *X*-ray image classification. We discuss issues in training and evaluating our model on the Chest X-ray8 dataset and report numerical findings. We also describe a binary classifier to identify any abnormalities in *X*-ray images.

Chapter 5 describes the experiments conducted using various weakly supervised localization methods. We also report the performance scores for the methods examined on the test data set.

Chapter 6 concludes the report with several remarks.

1.3.1 Dataset

The dataset used in this project is Chest X-ray8 [1]. The dataset comprises 112,120 frontal-view X-ray images of 32,717 unique patients with eight thoracic disease image labels. Each X-ray image can have multiple labels or multiple diseases. The eight thoracic disease image labels in the dataset are Atelectasis, Cardiomegaly, Effusion, Infiltration, Mass, Nodule, Pneumonia, and Pneumothorax. *X*-ray images are labelled by text mining the *X*-ray radiology report corresponding to the *X*-ray images. A sample *X*-ray image is shown in Figure 1, and the associated labels are Cardiomegaly, Infiltration, Mass, and Nodule.

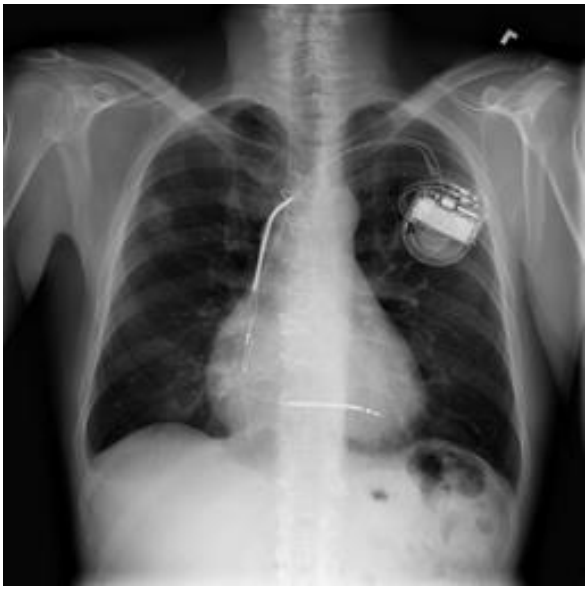

Figure 1: Chest- *X*-ray of a patient observed with Cardiomegaly, Infiltration, Mass, Nodule.

The pathologies (disease) keywords from the radiology report (corresponding to the X-ray) are extracted with various natural language processing (NLP) techniques that detect the pathology keywords. Radiological reports corresponding to the *X*-ray image will be either linked with one or more diseases or marked as ' No label' (when no disease is found in the *X*-ray report) as the reference category. Figure 2 illustrates the correlation between the diseases and reveals some connections between different pathologies. Since annotating a large-scale *X*-ray dataset by a radiologist expert is expensive and time-consuming, the disease labels are extracted from the radiology report using medical NLP tools like DNorm [2] and MetaMap [3]. The final labels are selected by merging the results from DNorm and MetaMap to maximize the recall. The NLP model's accuracy is evaluated by testing the model in comparison with the existing X-Ray dataset – Openl API, which is manually annotated by the radiologist. The evaluated model archives mean precision of 0.90, mean recall of 0.91, and mean F1-score of 0.90.

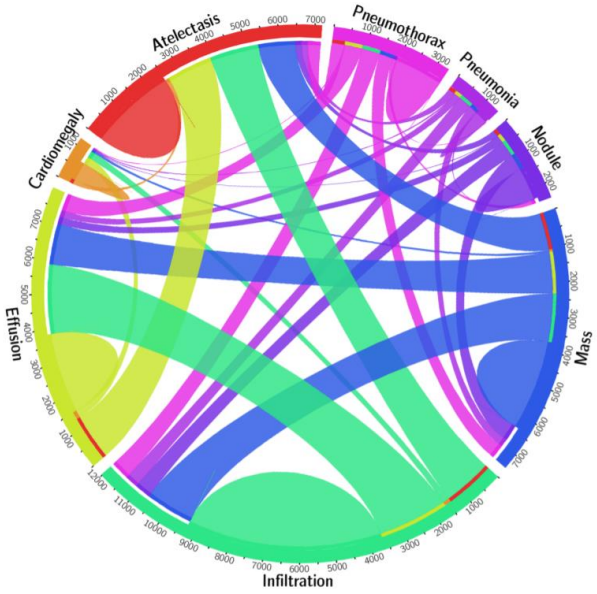

Figure 2: Correlation between 8 pathological diseases. Source [2]

The total number of appearances of each disease in the entire dataset is shown in Table 1, where "No label" represents the *X*-ray images that are not detected with any diseases. Also, a bar chart in Figure 3 visualizes the total number of appearances of each disease. The Table and Figure indicate that the dataset has a considerable imbalance between the classes, with almost half of the images labelled as "No label." Obviously, the class-imbalance issue, as well as label noise caused by natural language processing (NLP), need to be addressed while designing the model and evaluating its performance.

Disease	**Count**
Atelectasis	10585
Cardiomegaly	2559
Effusion	12295
Infiltration	18139
Mass	5327
Nodule	5754
Pneumonia	1317
Pneumothorax	5020
No label	58678

Table 1: Total occurrence of diseases in the entire dataset.

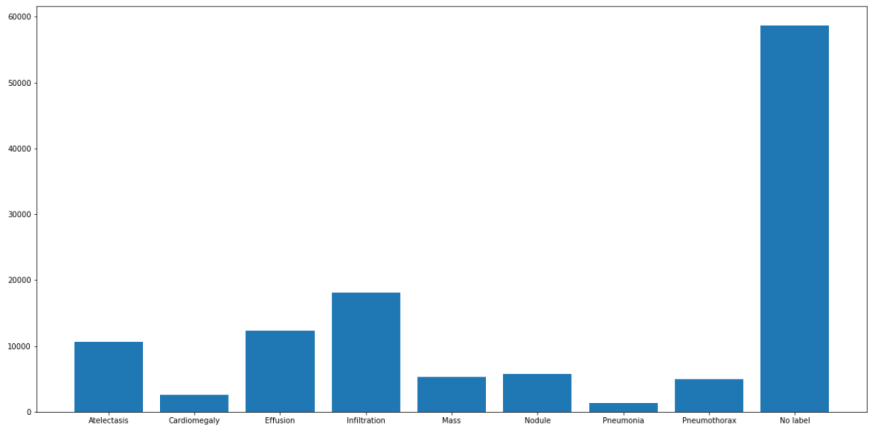

Figure 3: Bar chart of the total number of occurrences of each disease in the entire dataset.

The Chest-*X*ray-8 dataset can be downloaded from "https://nihcc.app.box.com/v/ChestXray-NIHCC". The entire dataset is divided into training, validation, and test sets for training and evaluation purposes. The distribution of labels for training, validation and test sets are given in Table 2 below.

Data	**Number of samples**
Train Set	75714
Validation Set	10810
Test Set	25596

Table 2: Label distribution in training, validation and test sets.

Table 3 illustrates a portion of the test.csv file containing the titles of the *X*-ray images and associated labels, where label '1' means the associated disease is present while label '0' means the disease is not present. For example, *X*-ray "004.png" contains Effusion, Infiltration, and Pneumothorax.

Name	Atelectasis	Cardiomegaly	Effusion	Infiltration	Mass	Nodule	Pneumonia	Pneumothorax
004.png	0	0	1	1	0	0	0	1
005.png	0	0	0	1	0	0	0	1
006.png	0	0	1	1	0	0	0	0
007.png	0	0	0	1	0	0	0	0

Table 3: Samples of the name of X-ray and labels corresponding to X-ray.

1.3.2 Bounding Box

In dataset ChestX-ray8, a small number of images with pathology are provided with hand labelled bounding boxes (B-boxes), which can be used as the ground truth to evaluate disease localization performance. Two hundred instances for each pathology are labelled with B-Boxes (1,600 instances total), consisting of 983 images. The B-Boxes in the images are labelled a board-certified radiologist identified with the image's corresponding disease instance. The B-Box is saved as a CSV file containing the image file name, disease keyword, and B-Box coordinates. B-Box coordinates include (*x*, *y*) and (*w*, *h*), where *x* and *y* represent the top-left coordinates of B-Box and *w* and *h* represents the width and height of B-Box, respectively. If an image contains multiple pathologies instances, then each pathologies instance is labelled separately and stored.

Chapter 2 Review of Deep Learning

In recent years, deep neural networks have made numerous advancements in pattern recognition and machine learning. Currently, deep neural networks are being used in a wide range of applications like recognizing handwritten digits [15], machine translation [16], language generation [21], playing board and video games like Atari [19], generating realistic images [17], generating fake videos [20], facial recognition [18], object tracking [17], object detection [26], and many others. Although most of these applications were previously attempted to solve with other statistical methods like support vector machines [23], decision trees [58], and domain-specific methods, in many cases, deep learning has improved accuracy, and its powerful generalization abilities have been removed the dependency on domain-specific knowledge.

Deep learning, especially in the context of computer vision, typically refers to the process of training deep neural networks to minimize an objective function corresponding to the given training (input images, target label) data. An adequately trained neural network is expected to work well on a test set that is not used during the training. This is accomplished by modifying the neural network parameters based on the gradients of the objective function with respect to these parameters. This procedure is called backpropagation.

2.1 Artificial Neural Network

An Artificial neural network operates a series of algorithms that endeavour to recognize underlying relationships in a data set by imitating how the human brain operates. A neural network mimics a way similar to the neural system inside the human brain. A “neuron” in a neural network represents a mathematical function that collects and classifies information according to a specific architecture (Figure 4). The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis. A fully connected deep neural network consists of multiple layers of neurons, as shown in Figure 5. Layers are made of interconnected neurons (Figure 4) that contain activation functions like sigmoid or ReLu [8]. The first layer of a network accepts an input pattern called the input layer, and the final layer, which generates the expected output, is called the output layer. All intermediate layers are called the hidden layers.

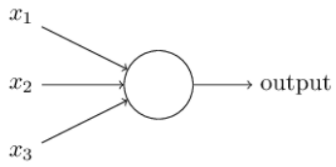

Figure 4: A neuron, source [33].

As illustrated in Figure 4, each neuron takes multiple inputs and outputs the sum of the product of inputs and their corresponding weights. These weights are the neural network parameters, which are trained by minimizing an objective function (e.g., mean square error) using a training data set.

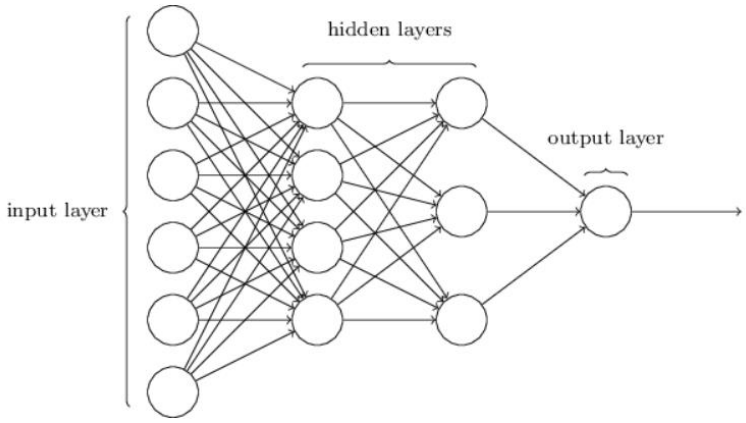

Figure 5: A fully connected neural network with two hidden layers, source [33].

The weight that connects the *k*th neuron in (*L*–1)th layer to *j*th neuron in the *L*th layer of the neural network is denoted as w_{jk}^L. Similarly, the bias of the *j*th neuron in the *L*th layer is denoted by b_j^L, while the activation of the *j*th neuron in the *L*th is denoted as a_j^L. Let σ the activation function like sigmoid or ReLU [8]. Then, the output at a_j^L is given by

$$a_j^L = \sigma\left(\sum_k w_{jk}^L a_k^{L-1} + b_j^L\right) \quad (2.1)$$

The input to the network is presented as $\boldsymbol{a}^0$ which is passed to the first layer of the neural network to generate the output $\boldsymbol{a}^1$ using Eq. (2.1). For a fully connected neural network of *k* layers, there are *k* + 1 pairs of parameter vectors which are denoted by $\{\boldsymbol{w}^L, \boldsymbol{b}^L\}$ for Layer *L* = 1, 2,…., *k* + 1. Equation (2.1) is recursively evaluated till the last layer is reached. The output of each layer is passed as input to the next layer. The final output $\boldsymbol{a}^L$ is often expressed as $\hat{\boldsymbol{y}}$. If the input

vector $\boldsymbol{a}^0$ has input dimenstion of N and output vector $\boldsymbol{a}^1$ has a dimension of M, then the weight matrix $\boldsymbol{w}^1$ has dimenstion of [NxM] and bias $\boldsymbol{b}^1$ has dimension of M. The network's parameters $\{\boldsymbol{w}^L, \boldsymbol{b}^L\}$ for layer L = 1, 2, …., k + 1 are updated to reduce the cost or error function, typically done by gradient descent.

2.2 Convolutional Neural Network

Convolutional neural network (CNN, also known as ConvNet) is one of the first successful architectures of deep learning, which finds applications in the classification of images, video, texts, and speech. Proposed by LeCun et al., the first modern framework of CNNs, known as LeNet-5 [29], was developed to classify handwritten digits. LeNet-5 has multiple neural network layers and was trained with the backpropagation algorithm.

CNN uses a unique architecture that is particularly well-adapted to classify images. Today, deep convolutional networks and several close variants are used in many neural networks for image recognition. While fully connected layers can solve many easy problems like MNIST image classification, they fail to work for larger datasets as they cannot capture spatial information and hence break when the images are slightly enlarged or rotated. Moreover, fully connected neural networks are costly to train and test compared to convolutional neural networks because the convolutional neural network needs fewer parameters than the fully connected neural network. Convolutional layers help us overcome these problems using learnable filters, slid across the entire input example to capture spatial relations or identify a particular feature.

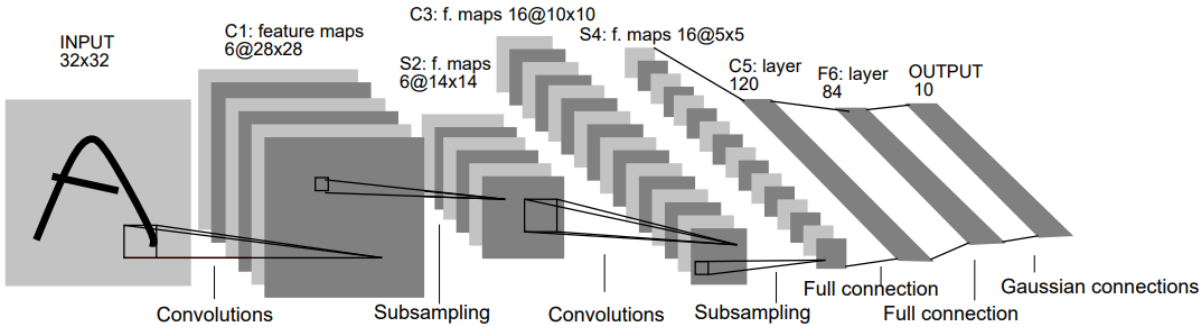

Figure 6: Architecture of LeNet-5, a convolutional neural network for digit recognition. Source [29]

The convolutional neural network's essential components are the convolutional layer, activation layer, pooling layer, and fully-connected layer. The convolutional layer tries to learn feature representations of the input. As illustrated in Figure 6, the convolution layer comprises several convolution kernels used to compute different feature maps; for example, in Figure 6 (LeNet-5), the first convolution layer contains six kernels of size 5 x 5. Each kernel takes a rectangular section of the preceding layer as input, computes a weighted sum (filtering, where the weights form a kernel matrix), and, with a particular stride (the number of pixels by which we slide the filter window over the preceding window) the kernel slides over same previous layer till it covers entire preceding units. Consequently, a convolutional layer is merely the result of 2-D FIR (finite-impulse-response) filtering [31]. Convolutional operation of a two-dimensional input image '*I*' with a two-dimensional kernel *K* can be represented as

$$Z_{(i,j)} = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.2}$$

In convolutional network terminology, the first argument ($\boldsymbol{X}$ in (2.3) below) of the convolution is often referred to as *input,* and the second argument ($\boldsymbol{W}$ in (2.3) below) is referred to as *kernel*. The output is sometimes referred to as *feature map* (z in (2.3) below). Mathematically, the feature value at the location (*i*, *j*) in the k^{th} feature map of the l^{th} layer $z^l_{(i,j,k)}$, is calculated as

$$z^l_{(i,j,k)} = \boldsymbol{W}^{l\,T}_k . \boldsymbol{X}^l_{(i,j)} + b^l_k \tag{2.3}$$

Where the $\boldsymbol{W}^l_k$ and b^l_k are weight matrix and bias term of the k^{th} filter in the lth layer, respectively, and $\boldsymbol{X}^l_{(i,j)}$ is the input matrix centred at the location (*i*, *j*) of the lth layer.

Commonly used activation functions like ReLU, Tanh, Sigmoid, leaky ReLU, Maxout, and ELU are illustrated in Figure 7. The activation function introduces nonlinearity to ConvNets, which is desirable for multi-layer networks to detect nonlinear features. Let $\sigma(.)$ represent the activation function, then the activation value $a^l_{i,j,k}$ is computed as

$$a^l_{i,j,k} = \sigma(z^l_{(i,j,k)}) \tag{2.4}$$

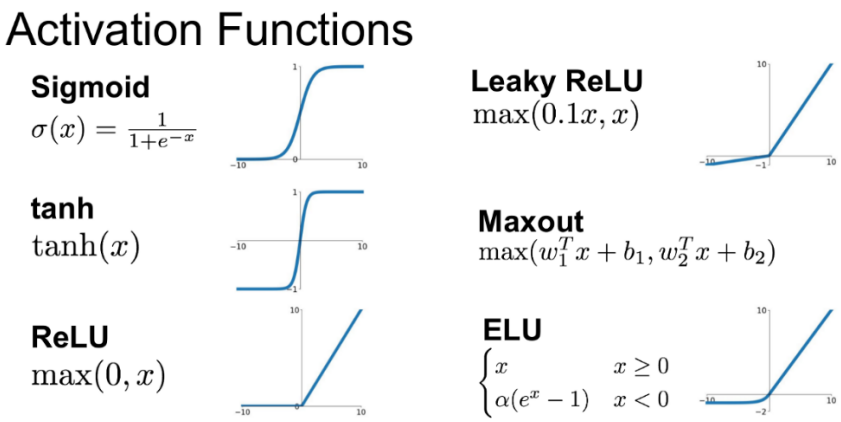

Figure 7: Commonly used activation functions in neural networks.

After the nonlinear activation function, the pooling layer is used to modify the layer's output further in order to reduce the spatial resolution of the feature map. The pooling layers are generally placed between two convolutional layers. Two commonly used pooling operations are Max pooling and Average pooling. Figure 8 illustrates the operation of Max and Average pooling, where each feature map of a pooling layer is connected to its corresponding feature map of the preceding convolutional layer.

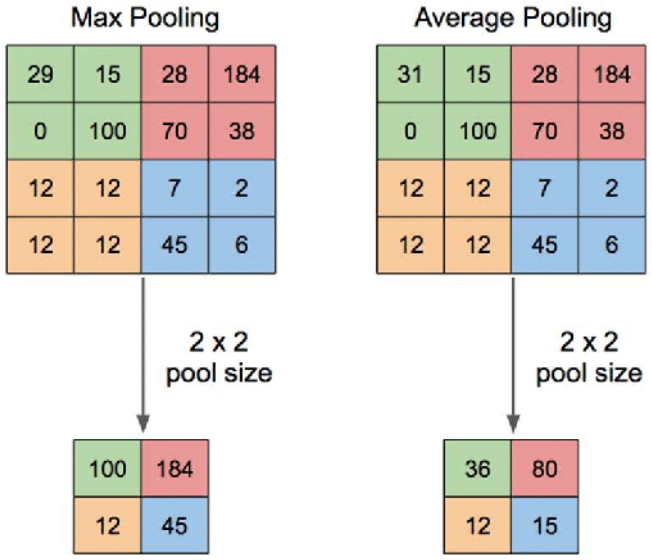

Figure 8: Example of Max and Average pooling.

The operation of pooling helps make the representation approximately invariant to small translations of the input. Here the term “invariance to translation” means that if one translates

an input by a small amount, the values of most of the pooled outputs do not change significantly [32].

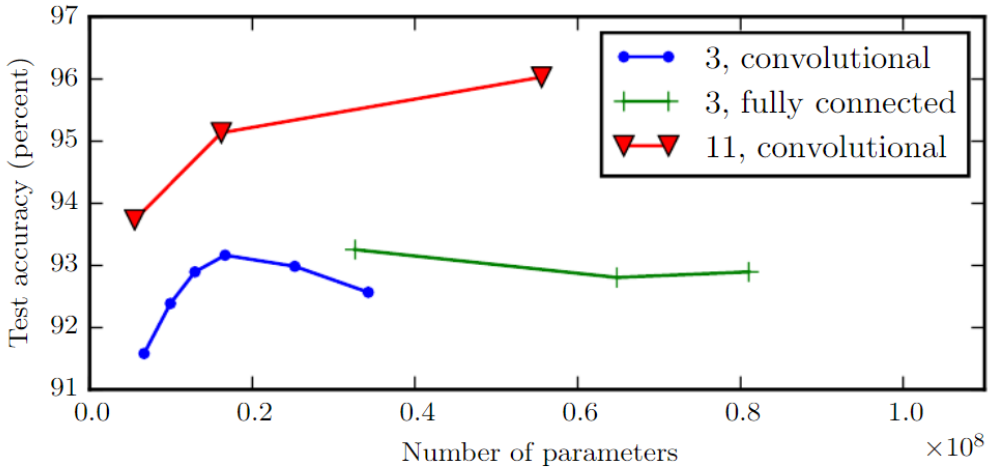

Figure 9: Effect of the number of parameters. Source [32].

Figure 9 compares the performance of a deep CNN (with 11 layers), a shallow CNN, and a fully connected neural network when classifying the test set from the MNIST database. From the figure, it is observed that deeper CNN tends to perform better. The experiment from [32] shows that increasing the number of parameters in layers of ConvNet without increasing their depth (increasing the number of layers) is not nearly as effective for performance improvement. Moreover, the shallow CNN overfits at around 20 million parameters, while deep CNN is not overfitting when employing 60 million parameters. This suggests that deep CNN has a better ability to learn more complex functions and patterns.

2.3 ResNet

Deep convolutional networks (DCNN) have recently achieved great success in image classification and object detection tasks. With the introduction of AlexNet (Krizhevsky et al., 2012) [9], the trend towards a deeper convolutional neural network with the increasing number of layers is found in, for example, GoogleNet (by Christian Szegedy] [10] and VGG (by Karen Simonyan) [11]. As convolutional neural networks become increasingly deep, new issues emerge: as information about input or gradient passes through many neural network layers, information starts to vanish or "wash out" before reaching the end (or beginning when performing backpropagation) of the

network [14]. In addition, experiments have found that in some cases, deeper networks may perform worse than shallow networks in terms of accuracy [12], as shown in the figure below.

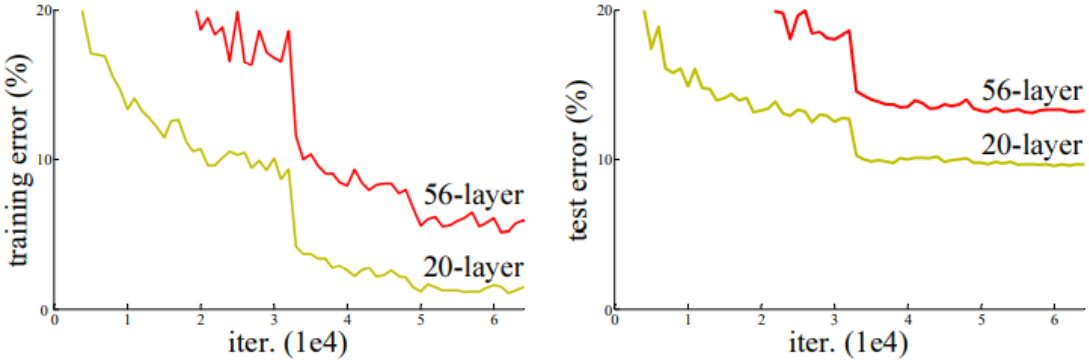

Figure 10: Performance degradation of deeper networks. Source [12].

ResNet [12] model came up with the idea of using *residual* layers to prevent deep CNNs from performance degradation, where an extra layer, called a residual layer, provides a *shortcut connection* between layers. Unlike conventional deep CNNs, which often find it difficult to approximate identity mapping over multiple nonlinear layers, the shortcut connections introduced in ResNet can readily approximate mapping without extra parameters or computational complexity. For illustration, a residual block is shown in Figure 11, where the network directly passes a copy of the input to the next output layer and later summed elementwise with the next layer's output.

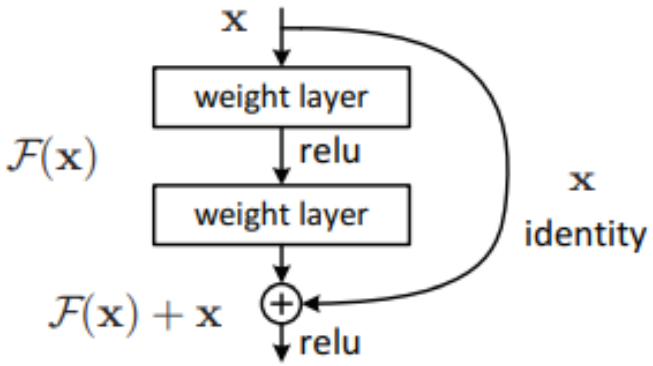

Figure 11: A residual layer in Resnet. Source [12]

The ResNet architecture has been shown to help reduce performance degradation in deeper networks [12]. A residual layer can be defined as

$$\boldsymbol{y} = F(\boldsymbol{x}, \{\boldsymbol{W}^i\}) + \boldsymbol{x} \tag{2.5}$$

Here y and x are the output and input of the residual layer, respectively, and function $F(\boldsymbol{x}, \{\boldsymbol{W}^i\})$ represents the intermediate convolutional layer between the input and output.

layer name	output size	18-layer
conv1	112×112	7×7, 64, stride 2
conv2_x	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$
	1×1	Average Pool

Figure 12: Layers in Resnet18. Source [12].

In this project, the CNN architecture we have used to classify X-Ray images is that of ResNet 18. The structure of ResNet 18 is shown in Figure 12, which accepts images of dimension 224 x 224. The network starts with a 7 x 7 2-D convolution with a stride of 2, followed by 3 x 3 max-pooling with a stride of 2 that is followed by four convolutional blocks with each convolutional block containing two 3 x 3 2-D convolutional layers. *A residual* layer is applied after each convolutional block. Finally, a global average pooling is applied on the feature map. In addition, batch normalization (BN) [13] is applied right after each convolution operation and before the activation function.

2.4 DenseNet

Dense Convolutional Network (DenseNet) [14] is composed of multiple dense convolutional blocks, as shown in Figure 14. And each convolutional block contains multiple layers densely connected, and each layer accepts inputs from all previous layer output features, as illustrated in Figure 13,. Unlike ResNet with a single connection to the previous layer containing only one skip connection, the DenseNet connects each layer in the block to every other subsequent layer, including multiple skip connections. In other words, instead of having *L* connections in a traditional *L*-layer CNN, an *L*-layer DenseNet contains *L*(*L*+1)/2 direct connections, where typically each layer is composed of batch normalization [13] followed by ReLU [8] activation and followed by 3x3 2-D convolution. The author had used batch normalization and ReLU before the convolution because it was found more efficient than the usual post-activation mode [14].

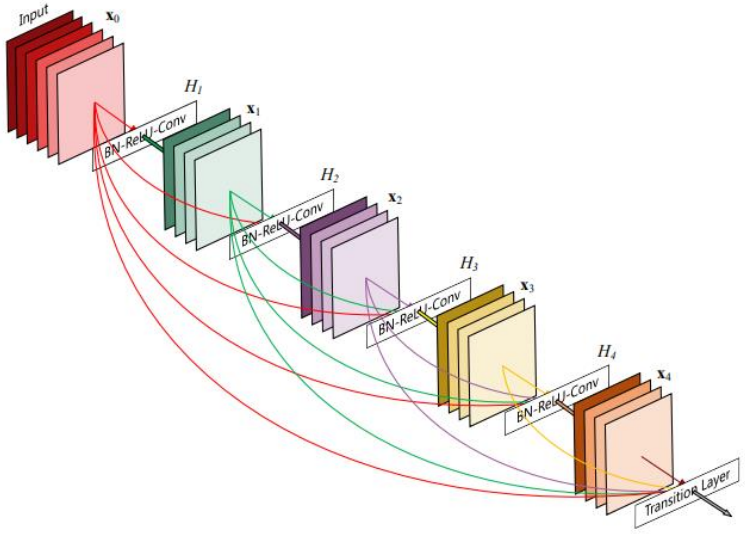

Figure 13: Convolutional Block in DenseNet with 5-layers. Each layer takes all preceding feature-maps as input—source [14].

As illustrated in Figure 14, each Dense block is connected to each other with a transition layer made of a batch normalization layer, a 1 x 1 convolution layer (1 x 1 kernel convolution generally requires only fewer parameters than 3 x 3 kernel, so they are computationally cheaper), and an average pooling layer. The average pooling layer in the transition layer down-samples the feature map from the previous dense block, or in other words, it reduces the spatial resolution of the

feature map by a factor of 2 because the average pooling has a stride of 2. Besides, the 1x1 convolutional layer in the transition layer regulates and reduces the number of channels in the feature map from the convolutional block. The channel numbers are regulated because higher channel outputs require more parameters in the next convolutional block to process input from the previous layer; hence the model becomes bigger. By reducing the number of channels, we could maintain the model's size.

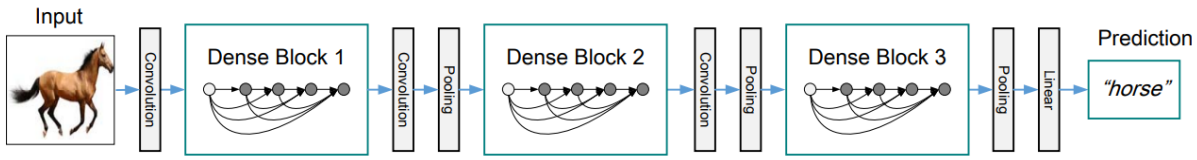

Figure 14: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers, which change feature-map sizes via convolution and pooling—source [14].

DenseNets have significant advantages over other networks as they alleviate the vanishing gradient problem as error signals can be easily propagated to earlier layers more directly and significantly reduced the number of parameters [14].

Layers	Output Size	DenseNet-121
Convolution	112 × 112	7 × 7 conv, stride 2
Pooling	56 × 56	3 × 3 max pool, stride 2
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	
	28 × 28	
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	
	14 × 14	
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Transition Layer (3)	14 × 14	
	7 × 7	
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$
Classification Layer	1 × 1	7x7 Global Average Pool

Figure 15: DenseNet121 architecture. Source [14].

In this project, one of the DCNN used to classify *X*-Ray images is DenseNet121. The structure of DenseNet121 is shown in Figure 15. The network accepts images of resolution 224 X 224. The network starts with a 7x7 2-D convolution with a stride of 2, followed by a 3 x 3 max-pooling with a stride of 2 and followed by four dense and transition blocks, one after the other. Finally, the feature map is passed to the global average pooling layer.

Chapter 3 Weakly-Supervised Localization

Reference [5], entitled “Learning deep features for discriminative localization”, demonstrates that convolutional neural networks can localize the discriminative image regions on various tasks despite not being trained for them. Also, a CNN learns to localize the objects in an image without explicitly training the locations of objects in the image. This sheds light on how it explicitly enables CNN to have remarkable localization ability despite being trained on image-level labels. This principle is taken seriously in this project to localize diseases in *X*-ray images, where a CNN model was trained to classify the *X*-ray to disease identification.

It is important to stress that this ability may get lost when fully-connected layers are used for classification [5]. The issue can be addressed using global average pooling, which also acts as a structural regularizer, to prevent overfitting during training. This technical trick allows quick identification of discriminative regions in images for diverse datasets, including those where the network was trained initially for classifying images or not trained with explicit localization labels. Here the term “localization labels” refers to the coordinates of bounding boxes for the objects in an image used in an object detection model like SSD [41]. Figure 16 shows that a CNN trained to classify images can locate discriminative regions in images. In this chapter, we discuss several weak supervised localization methods such as class activation maps [5], GradCAM [6], and GradCAM++ [7].

3.1 CAM

Class activation map (CAM) [5] is the method to identify discriminative regions in an image used by CNN to classify the image into a particular category or class. This method generates a heatmap, as shown in Figure 16, where the highlighted parts are regions being examined by a CNN or regions used by a CNN to discriminate the image and classify it to a particular class. The neural architecture used to generate a *class activation map* is illustrated in Figure 17. To generate a Class activation map, the CNN architecture needs to be slightly modified. A global average pooling is performed right before the fully connected layers or output layer and softmax. The global average pooling is applied on the feature map generated by the CNN model, which computes the spatial average of the feature map, separately for each channel, generating a single value

corresponding to each channel. A weighted sum of these values is used to make the final prediction or classification. Similarly, we compute a weighted sum of the last convolutional layer's feature maps, i.e. the feature maps, before taking the global average pooling to obtain class activation maps.

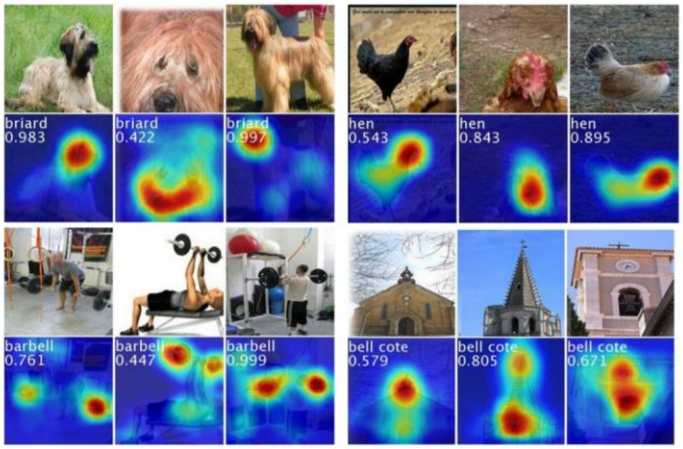

Figure 16: Heatmaps highlight the discriminative image regions used for image classification. Source Bolei Zhou [5].

As shown in Figure 17, the CNN can be used to extract features map $f^k(x, y)$ where *k* represents the channel number and (*x*, *y*) defines a spatial location. The dimension of the feature map is *C* x *M* x *N*, where *C* represents the number of output channels, *M* and *N* define the spatial resolution of the feature map. For channel *k*, performing the global average pooling generates F_k as

$$F_k = \sum_{x,y} f^k(x, y) \tag{3.1}$$

For class *c*, the input to softmax is given by

$$S_c = \sum_k w_c^k F_k \tag{3.2}$$

where w_k^c is the weight of the output layer corresponding to class *c* and channel *k*. Essentially, w_k^c represents the importance of F_k for class *c*. And finally, the output of softmax is P_c for class *c* by ignoring the bias, as bias has less impact on the classification performance, thus it is given by

$$P_c = \frac{exp(S_c)}{\sum_c exp(S_c)} \tag{3.3}$$

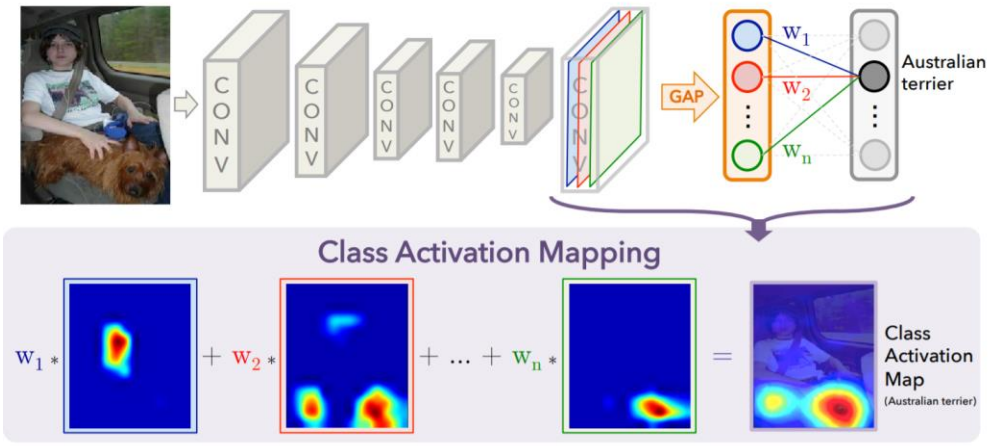

Figure 17: Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate class activation maps (CAMs). The CAM highlights class-specific discriminative regions. Source [5].

Substituting (3.1) into (3.2), we obtain

$$S_C = \sum_k w_c^k \sum_{x,y} f^k(x,y)$$

$$S_C = \sum_{x,y} \sum_k w_c^k f^k(x,y) \tag{3.4}$$

The class activation map for class *c* is represented as M_c, a weighted sum of the feature maps, namely

$$M_c^{\text{CAM}}(x,y) = \sum_k w_c^k f^k(x,y) \tag{3.5}$$

In this way, we produce a weighted spatial activation map for each class (with the size of *D* X *M* x *N*) by multiplying the feature map from CNN (*C* X *M* X *N*) and weights of the output layer (with dimension *C* X *D*). The $M_c^{\text{CAM}}(x,y)$ is the class activation map for class c, which has high values near where the object is present and low values where the object is absent. Finally, a threshold value is used to identify the exact spot (crop the region of interest) of the objects in the images.

3.2 GradCAM

GradCAM [6] is an alternative version of the class activation map, which can be used for broader CNN architectures without modifying exiting network architectures. In the CAM method, the feature map needs to be directly behind the softmax layer or output layer, so it only works with a particular type of CNN architecture that performs global average pooling over the convolutional feature map before the softmax layer (i.e., an information flow from convolutional feature maps to global average pooling, and then to softmax layer). Such CNN architectures may lead to inferior accuracy relative to general CNN networks on some tasks such as image classification.

The convolutional layers retain spatial information of an input image, which, however, will be lost in subsequent fully-connected layers. Therefore, to get the best spatial information as well as semantic information, the last convolutional layer is selected. Here the 'last convolutional layer' refers to the convolutional layer before any operations like global pooling or fully-connected layers. The neurons in these layers look for semantic class-specific patterns in the image (say, object parts in an image) [6]. These neurons get activated when particular patterns appear in the image. Grad-CAM uses the *gradient information flowing* into the last convolutional layer of the CNN to assign importance values to each neuron for a particular decision of interest. Moreover, it can be used to explain activations in any layer of a deep network [6].

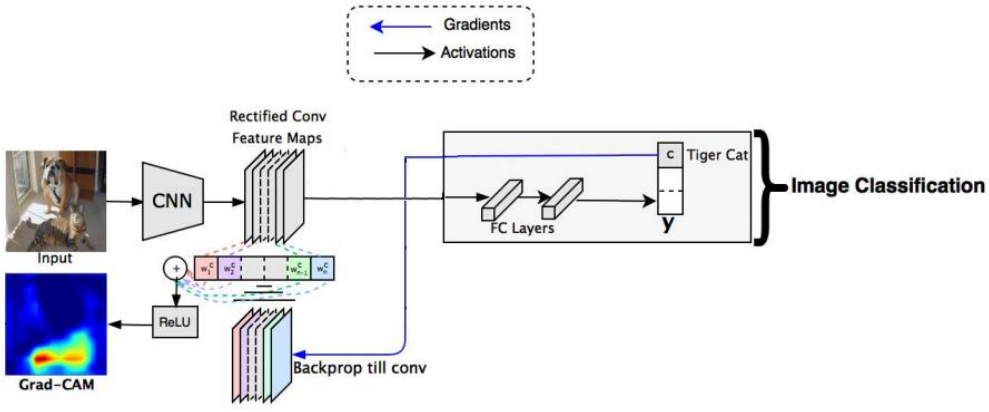

Figure 18: Pipeline of Grad-CAM-based image localization. Source [6].

The class discriminative localization map generated by GradCAM is given by $M_c^{\text{GradCAM}}(x, y)$ of size *M* x *N* where *M* and *N* define the spatial resolution of the localization map (called salient

map). Outline of the class discriminative localization map ($M_c^{\text{GradCAM}}(x, y)$) generated by GradCAM is illustrated in Figure 18.

In CAM, the importance of *k*th feature map f^k for a class *c* is represented by the weight of the output layer w_k^c. In the case of GradCAM, the importance of feature map *k* for class *c* is computed by estimating the gradient of probability score for class *c* (y^c), with respect to feature maps f^k of the convolutional layer, namely, $\frac{\partial y^c}{\partial f^k}$. Finally, the mean of gradients is computed over each feature map channel as

$$w_k^c = \frac{1}{z} \sum_{x,y} \frac{\partial y^c}{\partial f_{x,y}^k} \tag{3.6}$$

$$M_c^{GradCAM}(x, y) = ReLU(\sum_k w_c^k f^k) \tag{3.7}$$

where *z* is the total number of pixels in the activation map, and the sum of the product of weights w_c^k and feature map f^k is computed to generate the discriminative localization map. A ReLU activation function (where ReLU is the rectified linear unit activation function) is applied over the resulting output because ReLU passes only features with positive values, i.e., pixels whose intensity should be increased in order to increase y^c. Negative pixels are likely to belong to other categories in the image, which will be removed by the ReLU activation function [6].

3.3 GradCAM++

GradCAM++ [7] is a new approach that addresses the shortcomings of GradCAM. Although GradCAM can detect objects of different types, when an object of the same type occurs multiple times in an image, GradCAM fails to localize all occurrences of the object. This can be a severe issue as multiple occurrences of the same type of object in an image are prevalent in real-world scenarios. Furthermore, in some cases, localization may focus only on some parts of the object, such as the most discriminative region of the image [7]. This problem is illustrated in figure 19, and the figure compares the results generated by the GradCAM and GradCAM++. Objects in the figure are marked with green bounding boxes, and the image regions are visualized where higher values in the heatmap are observed. From the figure, we can show that GradCAM cannot detect multiple objects of the same type in the image, and in some cases, only some parts of the object are not focused like some parts of the van is not detected.

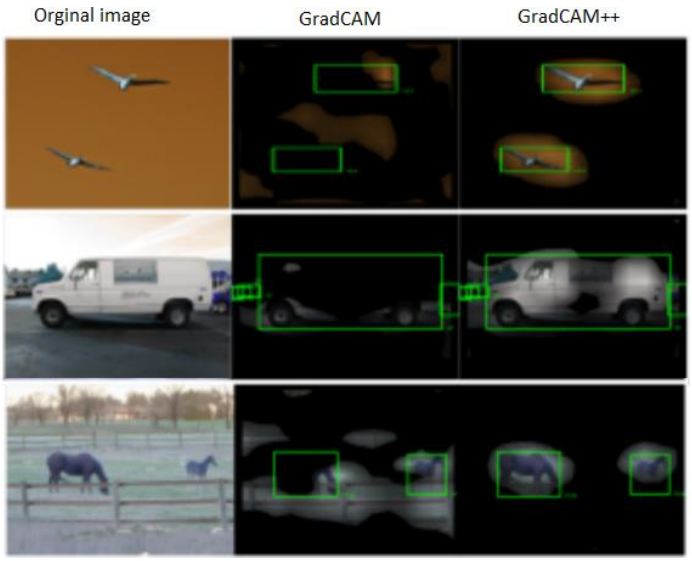

Figure 19: Comparison between GradCAM and GradCAM++, Source [7].

Hence, if there were multiple occurrences of an object with slightly different orientations or views (or, there are parts of an object that excite different feature maps), different feature maps may be activated with different spatial footprints, and the feature maps with lesser footprints fade away in the final saliency map [7]. For comparison, the weights $\boldsymbol{w}_k^c$ associated with class *c* and feature map *k* computed with CAM, GradCAM and GradCAM++ are illustrated in Figure 20.

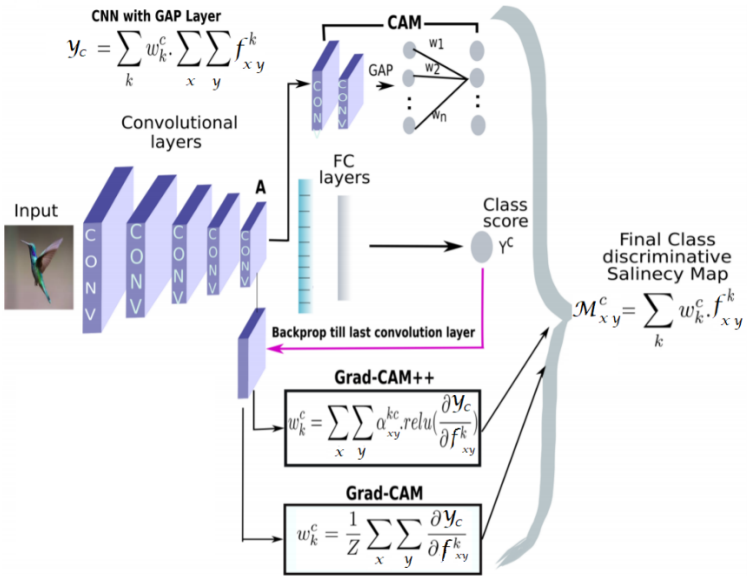

Figure 20: An overview of all the three methods – CAM, Grad-CAM, GradCAM++ – with their respective computation expressions. Source [7].

The above issue can be addressed by taking a *weighted average* of the pixel-wise gradients instead of directly taking the *average* computed in GradCAM. For GradCAM++, (3.6) is reformulated as

$$w_k^c = \sum_{x,y} \alpha_{x,y}^{k,c} . \text{relu}\left(\frac{\partial y^c}{\partial f_{x,y}^k}\right) \tag{3.8}$$

where ReLu [8] denotes rectified-linear-unit activation function and $\alpha_{x,y}^{k,c}$ are the pixel-wise weight coefficients of gradients for class *c* and feature map *k*. Here, ReLU is used to extract only positive gradients because the weighted combination of positive gradients with respect to each pixel in an activation map f^k strongly correlates with the importance of that activation map for a given class *c*. An empirical result for positive correlation can be found in [7]. Derivation to determine the gradient weight coefficient $\alpha_{x,y}^{k,c}$ for class c and feature map *k* can also be found in [7]. The gradient weight coefficient $\alpha_{x,y}^{k,c}$ is computed by (3.9) below

$$\alpha_{x,y}^{k,c} = \frac{\frac{\partial^2 y^c}{(\partial f_{x,y}^k)^2}}{2.\frac{\partial^2 y^c}{(\partial f_{x,y}^k)^2} + \sum_a \sum_b f_{a,b}^k \left\{\frac{\partial^3 y^c}{(\partial f_{x,y}^k)^3}\right\}} \tag{3.9}$$

By substituting $\alpha_{x,y}^{k,c}$ into (3.8), we obtain weights w_k^c as

$$w_k^c = \sum_{x,y} \left[\frac{\frac{\partial^2 y^c}{(\partial f_{x,y}^k)^2}}{2.\frac{\partial^2 y^c}{(\partial f_{x,y}^k)^2} + \sum_a \sum_b f_{a,b}^k \left\{\frac{\partial^3 y^c}{(\partial f_{x,y}^k)^3}\right\}}\right] . relu\left(\frac{\partial y^c}{\partial f_{x,y}^k}\right) \tag{3.10}$$

Finally, the salient map (highlighted regions) for class *c* is represented as $M_c^{\text{GradCAM++}}$ and is computed as the weighted sum of the feature maps, namely,

$$M_c^{GradCAM++}(x,y) = \sum_k w_k^c f^k \tag{3.11}$$

Chapter 4 Disease Classification

In Chapter 2, we discussed various deep convolutional architectures. This chapter will focus on experimental performance scores attained on our testing set by several trained architectures described in Chapter 2. We begin by introducing pre-processing procedures applied to the input image before passing it on to the network. We then present a detailed analysis of the deep learning model involved for multi-label classification on the Chest-Xray8 dataset. Besides, a performance comparison between pre-trained and randomly initialized models is provided. Finally, we introduce an abnormal classifier using DCNN, which is a classical binary classifier that classifies *X*-ray images as abnormal or normal. The *X*-ray image is considered abnormal if an *X*-ray image has one or more diseases associated with it, and the image is considered normal if no disease is detected.

4.1 Data Pre-Processing

The original size of the *X*-ray images in ChestX-ray8 is a single channel (black and white) 1024 × 1024 images. Since processing high-resolution images are computationally expensive, the images were either resized to a lower resolution of 512 x 512 or 224 x 224 without significantly losing the detailed contents.

All the experiments in this project are conducted on Nvidia GTX 1080Ti GPU, which has a memory size of 11 GB. Since the total amount of all *X*-ray images is large as 41.9 GB, it is not possible to fit all images in GPU memory. So the images are loaded as mini-batches containing only a few images from the entire dataset. The batch size or number of images in a single batch depends on the size (number of parameters) of the neural network as well as the GPU memory. These batches are dynamically loaded with the help of the PyTorch helper function Dataloader, which loads images and associated labels in batches. Apart from loading images, the Dataloader function performs augmentation, resizing, and standardizing images as well as converting images to tensor format. Parallel computing in GPU requires the matrix in the format of Tensor. Given below are some operational details while loading a batch of images:

1. Read a batch of images and their labels.

2. Since input images have only one channel (black and white image), each image is stacked in the channel axis by concatenating the image three times. A single-channel input image is defined as a 1 x *N* x *N* array, and the expected output image is a 3 x *N* x *N* array, where *N* defines the resolution of the image. This simple pre-processing step is necessary because some existing neural network models like RetinaNet only accept 3-D inputs.
3. Resize images to have expected resolution. For example, DenseNet neural network expects an input image of resolution 224 x 224.
4. Augmentations like colour jittering and random horizontal flipping are applied to the images. The Color jittering function randomly changes the brightness, contrast and saturation of the image, and the horizontally flip function flips an image randomly with a probability of 0.5.
5. Normalizes and standardizes input images.
6. Finally, convert input images and their labels to form a tensor array.

4.2 Unified DCNN Framework

In this project, a deep convolutional neural network (DCNN) is designed to identify one or more diseases present in *X*-ray images and later locate plausible regions of the diseases in the X-ray images. The problem is addressed by training a multi-label deep convolutional neural network, as illustrated in Figure 21. The network architecture is inspired by weakly-supervised object localization methods [34], where an *X*-ray image is passed through a pre-trained DCNN trained using the ImageNet dataset [35]. DCNN architectures like ResNet18 [12] and DenseNet121 [14] are employed for classification. In both ResNet18 and DenseNet121 (originally ResNet and DenseNet are designed to classify 1000 classes in ImageNet dataset), where typical final fully-connected layers and final classification layers are replaced with a transition layer, a global pooling layer, and a prediction layer. The heatmap of plausible disease location in *X*-ray is computed as sum of product of the feature map generated by the transition layer and weights of the prediction layer.

The global pooling and prediction layers in a DCNN are designed not only for classification but also for generating a likelihood map of the diseases, which is termed as heatmap or semantic

map. The top part of Figure 21 illustrates the process of producing a heatmap. The region with high values in the heatmap corresponds to the occurrence of a disease pattern with high probability. The pooling layer plays an essential role in choosing which information to be passed on to the next layer.

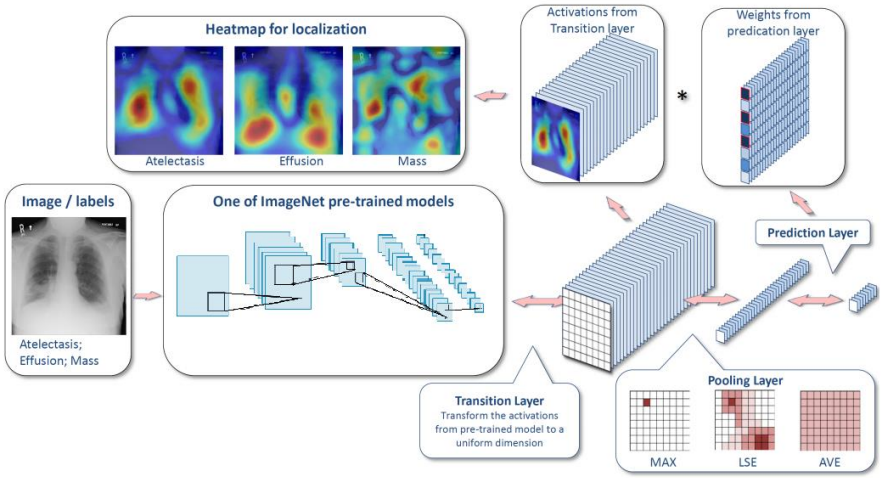

Figure 21: Pipeline of proposed unified DCNN framework and disease localization method

Besides the conventional pooling layers like max-pooling and avg-pooling layers, we have utilized the log-sum-exp (LSE) pooling proposed in [39]. The LSE pooling is performed over a region *S*, which is a square tile of size *N* x *N*, as

$$x_p = \frac{1}{r}\log\left[\frac{1}{s}\sum_{(i,j\in \boldsymbol{S})}\exp\left(r.x_{ij}\right)\right] \quad (4.1)$$

where x_{ij} represents the feature map value at location (*i*, *j*) in region ***S*** and *s* is the total number of pixels in the tile **S**. For example, when **S** is of size *N* x *N* tile then s is equal to N^2. By controlling hyper-parameter *r*, the pooling value can be changed. The pooling value reaches the maximum value in ***S*** when *r* approaches ∞, and the pooling value becomes the average value of ***S*** when *r* is reduced toward 0. Therefore, *r* serves as a tuning parameter between max pooling and average pooling. This ability of an LSE pooling layer helps improve the localization ability of DCNN. In all the experiments where LSE pooling is used, we have assigned the tuning parameter r = 10. For X-ray images, both classification and localization scores turn out to be highest when r = 10 [1]. It was experimentally found that when the r-value is close to 0, the LSE pooling acts as AVG pooling layer, and when the r-value is close to 20 or greater, the LSE pooling act as MAX pooling layer. .

Thus, when the value of r is 10, the LSE pooling layer acts as a pooling operation that has properties intermediate to max pooling and average pooling [1].

4.3 Loss Function

The images in Chest *X*-ray8 [1] dataset may contain more than one label; the task of classification is a multi-label classification. We consider a setting where each label is represented as an 8-dimensional vector $y = [y_1, y_2, \ldots, y_8]$ with $y_c \in \{0, 1\}$. A $y_c = 1$ signifies the presence of the c^{th} disease in the image, and the all-zero vector represents "normal," i.e. no-disease present in the X-ray image. The multiple-label problem at hand can be addressed as a regression task where, instead of using softmax, a sigmoid activation is used for each vector component (class), meaning that the loss for every component of the output vector is computed independently from the computation of other component values. We remark that loss functions such as Hingle loss, Euclidean loss, and normal cross-entropy loss do not work for this task because the image labels are highly sparse, meaning there are considerably more 0's than 1's in the labels because the dataset is highly imbalanced between normal *X*-ray images (as known as a negative class) and *X*-ray images with diseases (known as positive classes). Under these circumstances, we introduce *weighted cross-entropy loss* to balance positive and negative classes. If we denote the weights for balancing loss by β_P and β_N, a weighted cross-entropy loss is defined as

$$L_{WCEL} = -\beta_P \sum_{y_c = 1} y_c \log(\hat{y_c}) - \beta_N \sum_{y_c = 0} (1 - y_c) \log(1 - \hat{y_c}) \qquad (4.2)$$

where β_P and β_N are set to $\frac{P + N}{P}$ and $\frac{P + N}{N}$, and *P* and *N* represent the numbers of 1's and 0's present in a single image label batch, respectively.

4.4 Model Evaluation Methods for Classification

To validate the performance of a DCNN that classifies a given set of *X*-ray images, we decide to adopt multiple evaluation metrics, including AUC-ROC, which stands for the area under receiver operator characteristic curve, sensitivity (also known as recall), and specificity. Since ChestX-ray8 is a highly imbalanced dataset (from chapter 1), we will evaluate classifier performance for each disease other than the classifier's average performance on the entire dataset. This is to understand how the classifier performs on diseases with fewer samples in the dataset.

Sensitivity (also known as *recall* or *true positive rate* (TPR) measures how often the model correctly predicts a positive result for disease in input *X*-ray images, which actually have the disease that is being tested for. Consequently, a highly sensitive model for a disease will predict almost everyone as having the disease and not generate many false-negative results. For example, a model with 90% sensitivity to a particular disease shall correctly predict positive result for 90% of the *X*-ray images with that disease, and shall return negative result for 10% of the *X*-ray images with the disease that should have tested positive (called *false negative*) [42].

Thus the sensitivity measure can be defined as

$$\text{Sensitivity} = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \tag{4.3}$$

In a medical scenario, we often seek a model with low false negatives as it might appear to be life-threatening. A higher value of sensitivity would mean a higher value of true positive and a lower value of false-negative; and a lower value of sensitivity would mean a lower value of the true positive and a higher value of false negative. For the sake of healthcare, therefore, models with high sensitivity are desirable.

Specificity (also known as *true negative rate*) measures a model's ability to correctly predict a negative result for input images that *do not* have the health issue being tested for. A high-specificity test will correctly rule out almost everyone who *does not* have the disease and will not generate many false-positive results. For example, a model with 90% specificity shall correctly return a negative result for 90% of X-ray images that do not have the disease, and shall return a positive result for 10% of the *X*-ray images that do not have the disease and should have tested negative (called *false positive rate* (FPR)) [42].

The specificity is defined by

$$\text{Specificity} = \frac{\text{True Negative}}{\text{True Negative + False Positive}} \tag{4.4}$$

A higher value of specificity would mean a higher value of true negative and a lower false-positive rate; a lower value of specificity would mean a lower value of true negative and a higher false positive rate.

Receiver operator characteristic (ROC) curve is a metric to evaluate binary classification performance [44]. It is a curve that plots the ratio of TPR against FPR at various threshold values (add words here to define the term "threshold"). A threshold is a value to classify a point to either one of the classes. For example, at a threshold of 0.5, all values equal or greater than the threshold are mapped to one class, and all other values are mapped to another class. A metric we will adopt to check a model's performance is the area under the curve (AUC) of ROC that quantifies a classifier's ability to distinguish between classes and summarizes the ROC curve. The larger the AUC, the better the model's performance in distinguishing between the positive and negative classes.

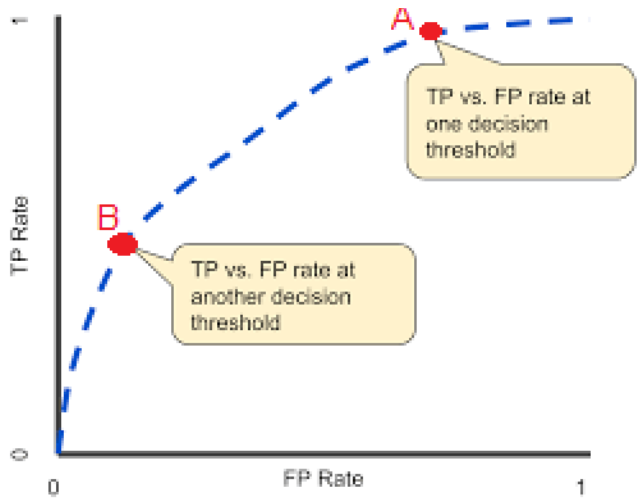

Figure 22: A sample diagram of the receiver operator characteristic (ROC) curve

For illustration, consider the sample diagram of the ROC curve shown in Figure 22, where point A represents a case with higher false-positive (than true negatives) as well as higher true positives (than false negatives), while point B represents a case with lower false positives (than true negatives) as well as lower true positive (false negatives). We see that a good choice of the threshold is one that achieves a balance between false positives and false negatives.

In contrast to binary classification and multi-class classification, the threshold selection is relatively trickier to determine in multi-label classification. In multi-classification, the classes are mutually exclusive, whereas in multi-label classification, each label represents a different classification task, but the tasks are somehow related. In the case of binary classification, a threshold of 0.5 is typically used [53]. In the case of multi-class classification (with a softmax layer

as output), argmax of softmax output (i.e., output with highest class score or probability) is often used to predict the most probable class. Multi-label classification differs from traditional single-label classification in that the model needs to predict multiple labels for each instance.

Since our task is a multi-label classification, we have one or more than one disease for each X-ray image, so we cannot use the softmax layer as the final layer as it only works for a single label output. Therefore, We will use the sigmoid function as the final output, so the output probabilities are independent of each other classes. The output produced for the experiment is eight probability corresponding to eight different diseases. Moreover, each probability is independent and does not depend on each other also. Also, the sum of all the eight probabilities is not equal to one. Thus, each class needs a different threshold value. In other words, it is not possible to identify a class by thresholding class score above 0.5 (as seen in binary classification) or computing argmax of output (as seen in the multi-class classification using softmax output). Therefore we need to select a threshold separately for each disease.

In this project, the threshold is selected based on Youden's index, which has been popular in the medical field to evaluate a test's performance on a validation set. Formally Youden's index can be defined by

$$\mathrm{J} = max_t \{Se(\tau) + Sp(\tau) - 1\} \quad (4.5)$$

where Se and Sp represent the model's sensitivity and specificity, respectively, and τ represents threshold or cut-point. The index summarizes the ROC curve used in interpreting and evaluating a model. The threshold τ that achieves maximum J is referred to as the optimal threshold point τ^* because it is the threshold that optimizes the effectiveness of a model. Since the dataset we use involves eight diseases, we have calculated eight threshold values, one for each disease, by optimizing the sensitivity and specificity. As a result, a class score higher than Youden's index (threshold) is positive [40], [43].

We remark that although sensitivity and specificity are available to validate the model, AUC-ROC remains the primary metric for model validation. This is because the sensitivity and specificity of a model depend on the threshold value we have selected. In addition, the original publication of the ChestX-ray8 dataset and many subsequent publications that have used this dataset have validated the model with AUC-ROC to evaluate performance.

4.5 Classification with ResNet18

In this section, we examine training and evaluation details of the ResNet18 model without pre-trained weights. The smallest ResNet architecture, Resnet18 (see Figure 12), is used as the backbone of the DCNN for a classification task. In the initial experiments, DCNN was loaded with random weights. In the next section, we will compare this setting with the model obtained using pre-trained weights. The architecture of interest contains 18 convolutional layers, and the network is trained with three different global pooling layers – max pooling, average pooling, and LSE pooling. The linear prediction layer's output has eight outputs, that are passed through the sigmoid function to get an individual probability for each disease. The model is trained with stochastic optimizer Adam [36] with a learning rate of 0.0001 and weight decay of 0.001. For the ResNet18 model with an initial learning rate less than 0.0001, the model seems not to learn effectively as the loss function stops dropping. In the training phase, the batch size is selected based on GPU's memory size and model size; in this experiment, 96 images were used in a single batch, and the training of the DCNN was performed with two Nvidia GTX 1080Ti.

	Train AUC	Validation AUC	Test AUC	Test Recall	Test Specificity
AVG	0.866	0.778	0.720	0.662	0.661
MAX	0.823	0.774	0.736	0.675	0.673
LSE	0.818	0.779	0.735	0.675	0.674

Table 4: Training and testing performance for ResNet18.

The performance of the ResNet18 model achieved in 20 epochs without pre-trained weights is reported in Table 4, where AUC under ROC has been used as the primary performance measure. In the table, “Train AUC” represents the AUC attained by the training set in the last epoch. While “Validation AUC” represents the best validation AUC attained with the validation dataset. The parameters of DCNN achieved at the best validation AUC are saved and later used for evaluating the testing dataset. The testing results are summarized as “Test AUC” in Table 4. The evaluation results with different pooling methods are also included in the table. We noticed that all global pooling layers achieve similar AUC, with the global MAX pooling layer leading in Test AUC with an AUC of 0.736.

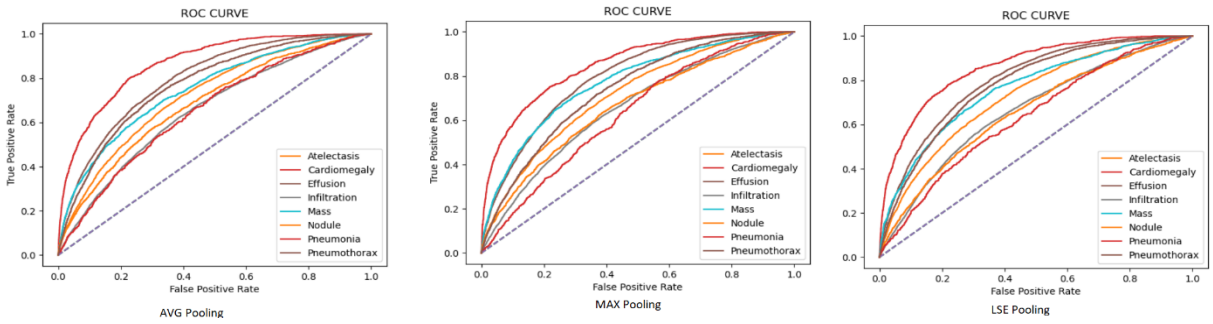

Figure 23: A comparison of ROC curve for ResNet18 with different pooling layers.

Each pathology's individual ROC curves for different pooling methods are shown in Figure 23, and the quantitative AUC-ROC for each pathology achieved with ResNet18 (without using pre-trained weights) is displayed in Table 5. With LSE global pooling layer, 'Cardiomegaly' (AUC_ROC = 0.859) and 'Pneumothorax' (AUC_ROC = 0.777) classes are consistently well recognized compared to other classes. Simultaneously, the detection ratios are relatively low for pathologies that have smaller discriminative regions that are hard to detect, which can be verified by the average size of the bounding box of these diseases, e.g., 'Infiltration' (AUC_ROC = 0.662) and 'Nodule' (AUC_ROC = 0.656). The pathology 'Pneumonia' (AUC_ROC = 0.634) has a low detection performance due to the unavailability of sufficient samples in that class (less than 1 percent).

	AVG Pooling	MAX Pooling	LSE Pooling
Atelectasis	0.7126	0. 7256	0.727
Cardiomegaly	0.8237	0. 8608	0. 859
Effusion	0.7933	0. 7924	0. 799
Infiltration	0.6577	0. 6751	0. 662
Mass	0.7445	0. 7576	0. 759
Nodule	0.6525	0. 6715	0. 656
Pneumonia	0.6308	0. 6550	0. 634
Pneumothorax	0.7419	0. 7491	0. 777

Table 5: Testing performance of ResNet18 for eight different diseases.

The DCNN was trained by running 20 epochs for all global pooling methods. The AUC and loss for each epoch with LSE global pooing are illustrated in Figure 24. The profile of "validation AUC" demonstrates that the validation AUC tends to saturate after 17 epochs, while "train AUC" keeps increasing. This indicates that the model is overfitting the training dataset after the 17th epoch.

Based on this we have only used model parameters that provide the best performance in the validation dataset for evaluating the testing dataset.

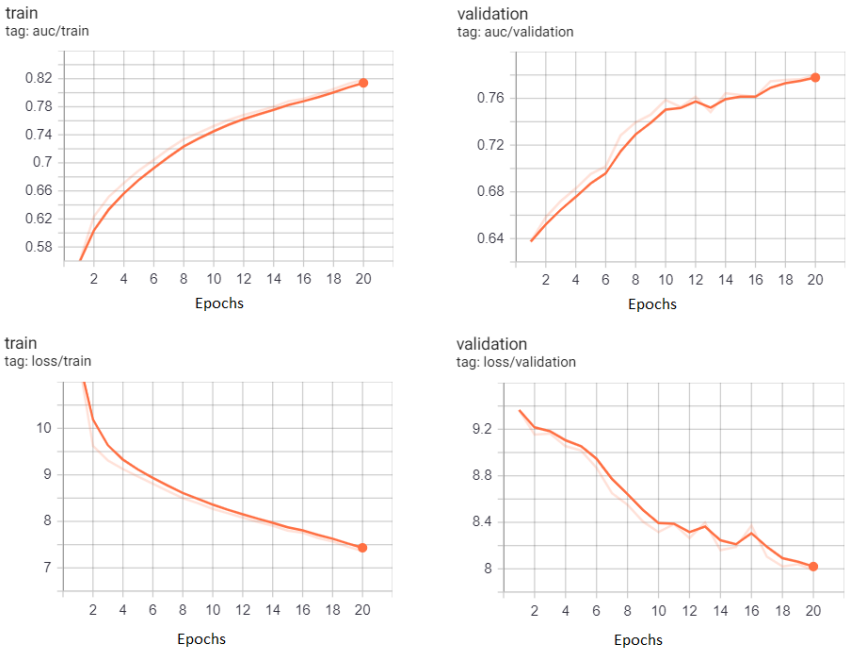

Figure 24: Plot of loss and AUC curve for ResNet18 with LSE pooling and without pretrained weights

4.6 Classification with ResNet18 with Pre-trained Weights

In this section, we examine the training and evaluation results of the ResNet18 model with pre-trained weights instead of initializing random weights as in the last section. To do so, we load the model with pre-trained weights of ResNet18 trained with the ImageNet dataset. The numerical results obtained with this model are reported in Table 6.

	Train AUC	Validation AUC	Test AUC	Test Recall	Test Specificity
AVG	0.9136	0.796	0.747	0.686	0.686
MAX	0.8701	0.801	0.763	0.697	0.697
LSE	0.8815	0.797	0.760	0.696	0.696

Table 6: Training and testing performance for ResNet18 with pre-trained weights.

Comparing Table 5 with Table 6, it is observed that the use of pre-trained weights has shown to improve Test AUC by approximately 3 percent. We have also seen improvements in terms of Validation AUC. These demonstrate the ability of pre-trained weights to improve performance

on small datasets like ChestXray8. Furthermore, with pre-trained weights, the model learns much faster; actually, the model reaches its best performance in 5 epochs. In contrast, models initialized with random weights require 20 epochs to finish. As it is evident from the loss plot in Figure 26, the model starts to overfit from the 5th epoch forward. The final model used to test is the model parameters with the highest performance on the validation dataset.

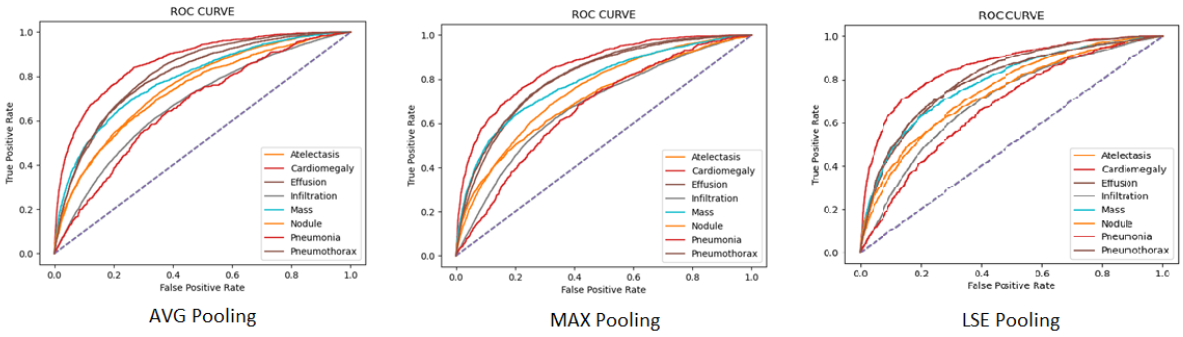

Figure 25: A comparison of ROC curve of ResNet18 with pretrained weights for different pooling layers.

Each pathology's individual ROC curves for different pooling methods are shown in Figure 25, and the AUC-ROC for each pathology achieved with ResNet18 with pre-trained weights for each global pooling layer is reported in Table 7. The simulation outcome indicates that AUC to identify each pathology has improved with the pre-trained weights. We have seen a significant improvement in performance on the test set for ResNet18 architecture loaded with pre-trained weights over the model loaded with random weights.

	AVG Pooling	MAX Pooling	LSE Pooling
Atelectasis	0.740	0.742	0.741
Cardiomegaly	0.858	0.837	0.859
Effusion	0.803	0.816	0.801
Infiltration	0.635	0.692	0.675
Mass	0.780	0.797	0.785
Nodule	0.707	0.723	0.722
Pneumonia	0.630	0.689	0.679
Pneumothorax	0.821	0.801	0.811

Table 7: Testing performance of ResNet18 with pre-trained weights on eight different diseases.

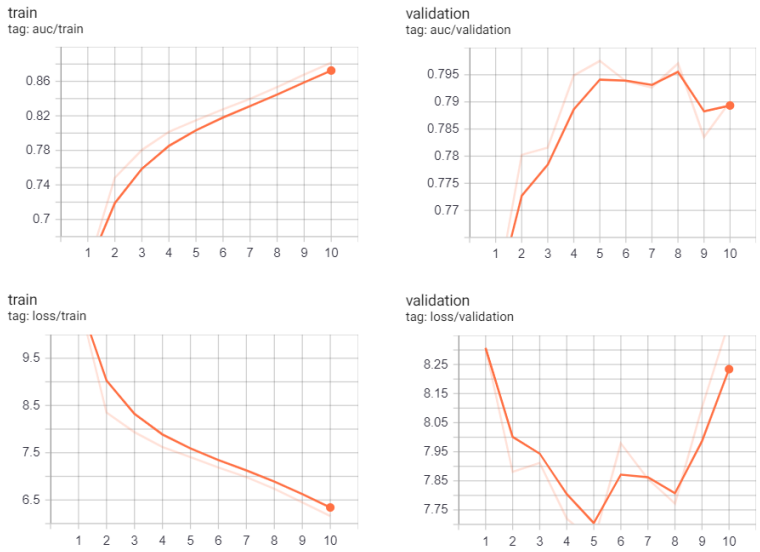

Figure 26: Plot of loss and AUC curve for ResNet18 with LSE pooling and pre-trained weights.

4.7 Classification with DenseNet121

In this section, the training and evaluation details of the DenseNet121 [14] model with pre-trained weights are discussed. The architecture of DenseNet121 is shown in Figure 15. The DenseNet121 model is trained on the ChestX-ray8 dataset. The input resolution of the *X*-ray images utilized for training the model was 224 x 224. We have previously mentioned the importance of using pre-trained weights, and we have decided to use pre-trained weights for the DenseNet model as well. Like ResNet18, we have trained the model with global average-pooling, global max-pooling, and global LSE-pooling. The training, validation, and test performance achieved on the DenseNet121 model are summarized in Table 8.

	Train AUC	Validation AUC	Test AUC	Test Recall	Test Specificity
AVG	0.940	0.806	0.747	0.684	0.684
MAX	0.874	0.813	0.775	0.705	0.712
LSE	0.888	0.813	0.771	0.705	0.706

Table 8: Training and testing performance for DenseNet121

It is observed that the DenseNet121 model achieved the highest Test AUC of 0.775 with global max pooling, a 1.2 percent improvement over the highest performance achieved by the ResNet18 model. We also observe improvements in Test Recall and Test Specificity.

	AVG Pooling	MAX Pooling	LSE Pooling
Atelectasis	0.723	0.764	0.750
Cardiomegaly	0.811	0.868	0.855
Effusion	0.801	0.813	0.819
Infiltration	0.656	0.688	0.684
Mass	0.776	0.794	0.794
Nodule	0.722	0.750	0.736
Pneumonia	0.672	0.678	0.697
Pneumothorax	0.810	0.839	0.831

Table 9 Testing performance of DenseNet121 for eight different diseases.

Each pathology's individual ROC curves for different pooling methods are shown in Figure 27, and the quantitative AUC-ROC for each pathology achieved with DenseNet121 with pre-trained weights for each pooling method is given in Table 9.

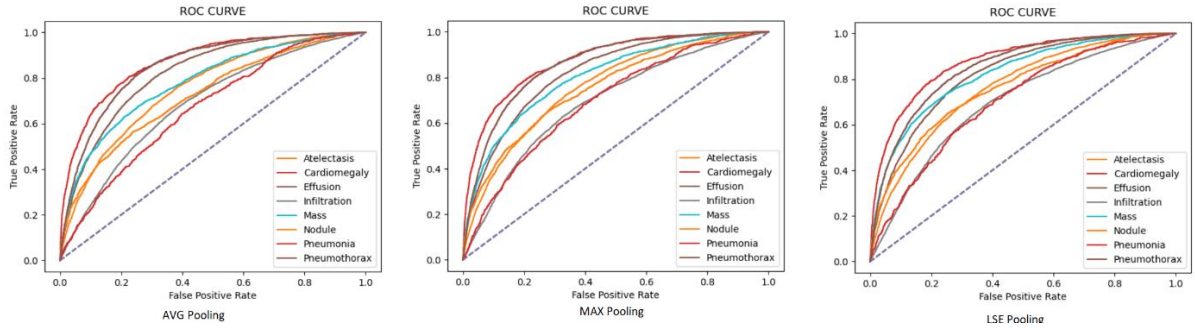

Figure 27: A comparison of ROC curve of DenseNet121 for different pooling layers.

DenseNet121 model is trained for 15 epochs. As DenseNet121 has more parameters to fit the model in GPU, we reduced the batch size to 12. By trial and error, the learning rate was set to 0.00001. The training and validation logs of AUC and loss are depicted in Figure 28. From figure 28, it is observed that the model starts to overfit after the eleventh epoch.

In our studies, the current model was trained with an input image dimension of 224x224; however, some authors have observed that the performance of DCNN also depends on the resolution of input images [37], suggesting that increasing input resolution may improve the

performance of the model. In the next section, we will present results achieved with higher input image resolution.

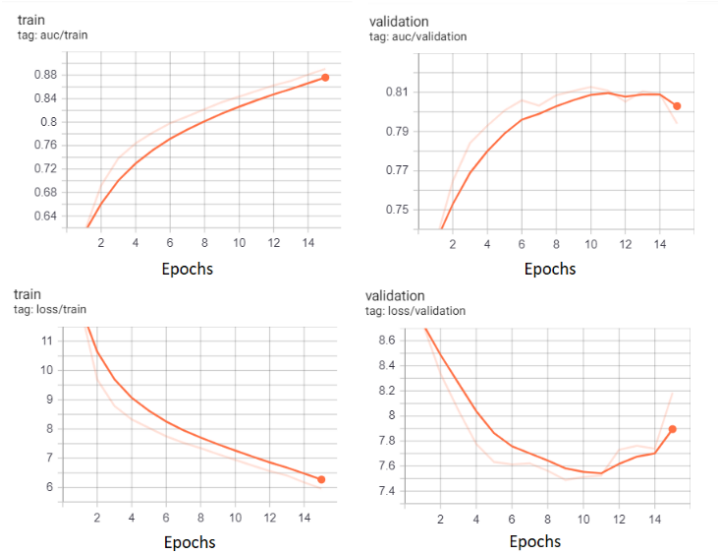

Figure 28: Plot of loss and AUC curve for DenseNet121 with LSE pooling and pre-trained weights

4.8 Classification with DenseNet with Higher Input Resolution

Originally DenseNet121 was designed to deal with input images of resolution 224x224. In this section, we will examine DenseNet121 with a higher input image resolution of 512x512. Since the existing architecture contains the global pooling layer before the prediction layer, we do not need to add extra CNN layers or pooling layers to reduce the feature map's spatial size caused by the higher input image resolution. The training, validation, and test performance achieved on the DenseNet121 model are summarized in Table 10.

	Train AUC	Validation AUC	Test AUC	Test Recall	Test Specificity
AVG	0.936	0.821	0.758	0.697	0.697
MAX	0.880	0.824	0.790	0.719	0.719
LSE	0.888	0.823	0.785	0.718	0.718

Table 10: Training and testing performance for DenseNet121 with higher input resolution.

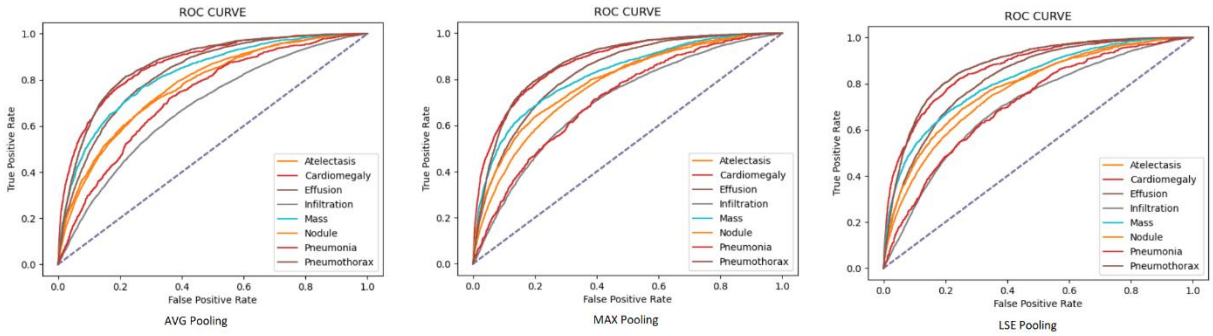

Figure 29: A comparison of the ROC curve of DenseNet121 with the high input resolution for different pooling layers.

Like the previous section, the model was trained for 15 epochs at a learning rate of 0.00001. The plots of AUC and loss versus epoch on the training and validation datasets are shown in Figure 30. Each pathology's individual ROC curves for different pooling methods are shown in Figure 29, and the AUC-ROC achieved by DenseNet121 using pre-trained weights for each pathology and each pooling method are given in Table 11.

	AVG Pooling	MAX Pooling	LSE Pooling
Atelectasis	0.746	0.768	0.761
Cardiomegaly	0.809	0.859	0.859
Effusion	0.804	0.826	0.824
Infiltration	0.668	0.696	0.696
Mass	0.782	0.803	0.810
Nodule	0.745	0.783	0.773
Pneumonia	0.657	0.707	0.697
Pneumothorax	0.850	0.862	0.860

Table 11: Testing performance of ResNet18 for eight different diseases.

We can see improvement in the Test AUC of the DenseNet121 model with higher input resolution. The Test AUC of DenseNet121 with max-pooling is increased from 0.775 to 0.790. This improvement is credited to higher input image resolution. In conclusion, increasing the input image’s resolution has a positive effect on deep learning models' performance.

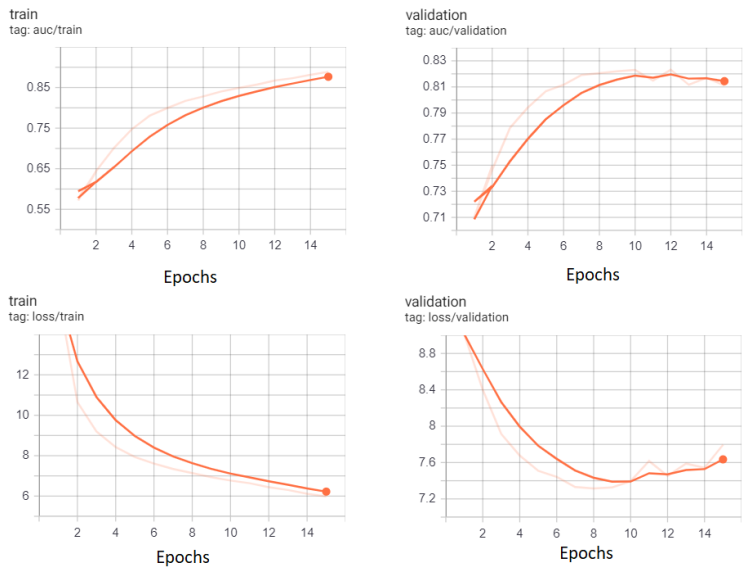

Figure 30: Plot of loss and AUC curve for DenseNet121 high-resolution input with LSE pooling and pretrained weights.

	Atelectasis	Cardiomegaly	Effusion	Infiltration	Mass	Nodule	Pneumonia	Pneumothorax	Mean
Our	**0.761**	**0.859**	**0.824**	**0.696**	**0.810**	**0.773**	**0.697**	**0.860**	**0.785**
Baseline [1]	0.707	0.814	0.732	0.612	0.560	0.716	0.633	0.789	0.695

Table 12: Classification performance between our result to baseline paper X. Wang [1].

In Table 12, the classification performance attained by our models is compared to the outputs recorded in baseline paper [1]. The results given table in 'our' is the best performance attained with DenseNet121 with an input image resolution of 512x512. The metric used to compare the performance is AUC-ROC. From the table, it is clear that our model performed better on average than the baseline model paper [1]. Also, the individual performance of each disease is higher in our model than in the baseline paper. The mean AUC of our model is improved by 9 percentage. This improvement could result from various factors like image augmentation, new deep convolution architecture like DenseNet121, which has a higher ability to classify images, and from pretrained weights.

4.9 Abnormality Classifier

In the previous sections, various deep convolutional neural networks with different configurations have been trained and evaluated. However, the performance of these algorithms

remains to be a lesser accurate relative to that of radiologists. The lack of superiority in performance might be due to the class-imbalance of the dataset and label noise caused by natural language processing as reported in chapter 1 (section 1.3) (the label of the X-ray images are extracted using the NLP tool from the *X*-ray report, which has only F1-score of 0.90, this means the label has noise in them). This section reports an experiment with DCNN that acts as a binary classifier to identify abnormal chest *X*-rays or *X*-ray images with any disease (including eight diseases in the ChestX-ray8 dataset). To verify the ability of DCNN to detect abnormalities within *X*-ray images, we divide ChestXray-8 into two categories, namely 'normal' and 'abnormal'. Major abnormal cardiac and pulmonary findings in this dataset include Cardiomegaly, lung opacity (including pneumonia, consolidation, and infiltrate), mass, nodule, pneumothorax, pulmonary atelectasis, edema, emphysema, fibrosis, hernia, pleural effusion, and thickening. *X*-ray images with these abnormalities are binned into the "abnormal" category, while other *X*-ray images are binned into the "normal" category. From the entire ChestXray8, a balanced dataset between abnormal and normal was selected for training and evaluation. A balanced dataset of 11596 *X*-ray images was selected, among which 10252 images were used for training, and the rest 1344 images were used for validation. The training and validation labels were assigned with the same automated NLP tool, while the test set was labelled in two different manners. Initially, the test set containing 1344 samples (Test1) was labelled with the NLP tool and later, an expert radiologist verified and corrected any erroneous labels. In test set two (Test2), the same test images were relabeled by taking three US-board-certified radiologists' consensus. These data sets and labels are available for download from [39].

	Train AUC	Val AUC	Test 1 AUC	Test 1 Recall	Test 1 Specificity	Test 2 AUC	Test 2 Recall	Test 2 Specificity
ResNet18	0.960	0.923	0.977	0.929	0.929	0.982	0.940	0.940
DenseNet121	0.948	0.928	0.976	0.926	0.932	0.981	0.932	0.930

Table 13: Training and testing performance for DenseNet121 and ResNet18.

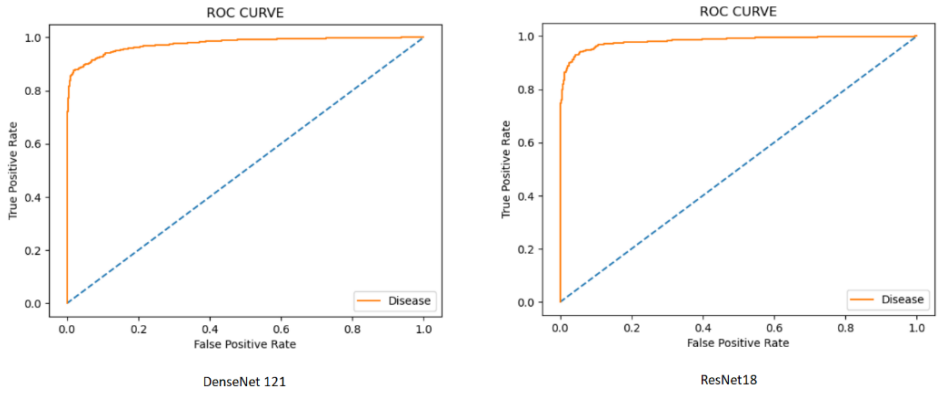

Figure 31 A comparison of ROC curve of DenseNet121 and ResNet18 as abnormality classifier in Test1.

As the problem at hand is a binary classifier, we modify the final prediction layer from eight outputs to a single output to train the model while the rest of the architecture remains the same. Both DenseNet121 and ResNet18 models were used for training and evaluation for abnormality detection. The model was trained for 15 epochs with an initial learning rate of 0.0001. We have used global max-pooling in both models because it achieved higher multi-label classification performance. The results achieved with both architectures are summarized in Table 13. The ResNet18 model achieves an AUC of 0.977 on Test1 and 0.982 on Test2 and achieves a recall of 0.929 and 0.940 on Test1 and Test2, respectively. The ROC curves of Test1 are shown in Figure 31.

Chapter 5 Disease Localization

This chapter presents several weakly supervised localization methods and discusses the experimental results obtained by our method compared to the baseline results published in [1]. We started by describing the procedures to generate the heatmap and bounding boxes in the heatmap. Also, we report performance analysis of weakly supervised localization methods, including CAM, GradCAM and GradCAM++.

5.1 Background and Localization Procedures

In dataset ChestX-ray8, a total of 983 images are provided, and bounding boxes labelled by radiologists are used to validate the weakly localization method's ability. For each *X*-ray image, a single disease label is attached with its corresponding bounding box location. In our experiments, these ground truth bounding boxes will be compared with the predicted bounding boxes. We have initially passed *X*-ray images as the input to a DCNN for all three localization methods and extracted the feature map from the final convolutional layer of the DCNN. We recall that the final convolutional layer in ResNet18 and DenseNet121 is the layer just before the global pooling layer.

For the ResNet18 model, we have selected the feature map after the batch normalization layer in the 4th convolutional layer. Using PyTorch, the feature map is stored as key-value pair, where the key is the layer's name and the value is the feature map or output generated by the corresponding layer. For the ResNet18 model, we have accessed the feature map with the key “layer4_bottleneck1_bn2”, representing the batch normalization layer in the 4th convolutional block. The extracted feature map has a dimension of 16 x 16 x 512 for an input image of 512 x 512. The spatial resolution of the feature map is 16 x 16, and the feature map contains 512 channels. We remark that as the input image resolution increases, the feature map's spatial resolution will also increase. In our experiment, we have used an input *X*-ray image of resolution 512 x 512 to localize diseases better. The reason we limit the input image resolution to 512 x 512 is because of the limitation on computational memory. In this project, we have only used the ResNet18 model to localize diseases because the localization ability of DenseNet121 was found to be unsatisfactory relative to ResNet18.

The importance of *k*th feature map f^k for class *c* is represented by the weight vector w_k^c. This weight vector may be computed using different methods like CAM, GradCAM, and GradCAM++. The formulas to compute weight w_k^c with the three methods are detailed in Chapter 3 of this report and will be explained in the following sections. The sum of product between the weight vector of class *c* (w^c) and feature map (f_k) generates the heatmap we required for localization. For ResNet18, the weight vector of a class (w^c) has a dimension of 512, and the feature map has a dimension of 16 x 16 x 512; hence the resulting heatmap has a dimension of 16 x 16. The picture on the left-most side in Figure 32 illustrates a heatmap generated for cardiomegaly disease using GradCAM++. The highlighted region (yellowish-green region) in the heatmap is the most plausible region of that disease. The red bounding box in the figure is the original bounding box of the disease. Heatmaps are then normalized to within the range of [0, 1] and resized back to the original input image resolution (in our case, it is 512 x 512), where the resize is performed using function OpenCV.resize().

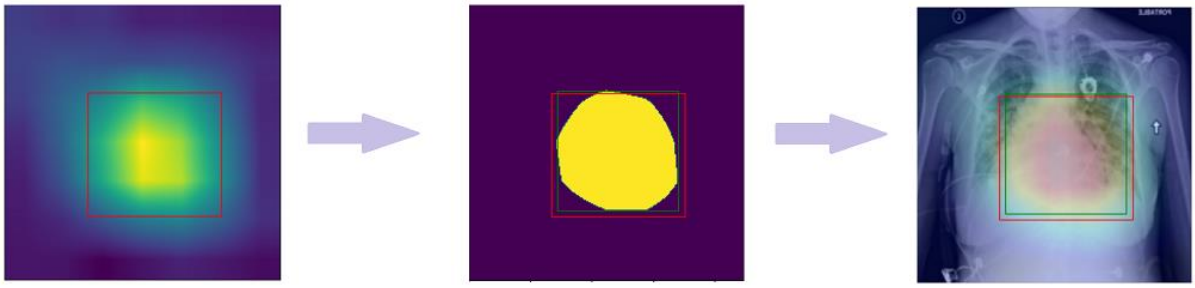

Figure 32: The process of thresholding the heatmap and generation of bounding boxes.

To generate a bounding box around the disease's plausible region, we need to threshold the heatmap. The threshold is selected based on the mean of the entire heatmap by using the formula below

$$\text{threshold} = \min\left(0.65, \sum \frac{\text{heatmap}}{n} * th\right) \quad (5.1)$$

where n represents the total pixels in the heatmap. The threshold is limited to a maximum of 0.65, so any value from the second term smaller than 0.65 will be truncated to 0.65. Varying variable $'th'$ had significant effects on the area of the region that is thresholded (in figure 32, the thresholded region is shown as yellow colour). As $'th'$ increases, the thresholded region as well

as the size of the bounding boxes will decrease. The value of $'th'$ is selected to increase the *intersection of union* between predicted bounding boxes and ground truth bounding boxes. In this project, the value of $'th'$ is selected from range [0.1, 0.25].

The threshold generated using (5.1) was found more effective than selecting a constant threshold for all heatmaps because the overall brightness is different for each heatmap. In Figure 32, the picture in the middle represents the thresholded heatmap.

In the next step, bounding boxes are generated around all nonnegative-value regions in the thresholded heatmap. Function "findContours()" from OpenCV was used to detect all nonnegative regions in the heatmap. We remark that this function also returns the coordinates of all founded bounding boxes. A detected bounding box is illustrated in Figure 32 as a green colour box around the thresholded region. As final outcome, both the input *X*-ray image and heatmap are summed together and displayed with a bounding box around the disease's plausible region. A sample illustrating a final image is displayed on the rightmost side of Figure 32.

5.2 Evaluation Methods for Localization

Computed bounding boxes are evaluated in comparison with the ground truth box that is hand-labelled by radiologists. Although the total number of bounding-box annotations is relatively low compared to other object detection datasets like COCO or Pascal VOC dataset, it may still be sufficient to get a reasonable estimate of how the proposed method performs a weakly-supervised localization task. To estimate the method's performance or the accuracy, we have used evaluation criteria such as "intersection over union" and "intersection over detected bounding-box". These evaluation metrics are detailed below.

Intersection over Union **(IoU)** is an evaluation metric used to measure an object detector's accuracy on a particular dataset [45]. The ground-truth bounding boxes are hand-labelled bounding boxes from the testing set that specify the object's locations in the images, and the predicted bounding boxes indicate the locations of the objects predicted by our model. IoU is the ratio between the area of overlap between a predicted bounding box and a ground-truth bounding box, and the area of union between the predicted bounding box and the ground-truth bounding box. The computation of IoU is illustrated in Figure 33.

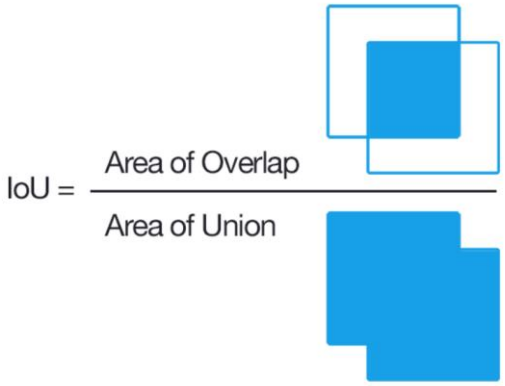

Figure 33: Computing the Intersection over Union is as simple as dividing the area of overlap between the bounding boxes by the area of union. Source [45]

Intersection over the detected Bounding-Box **(IoBB)** is the ratio of overlap area between a predicted bounding box and a ground-truth bounding box and the area of the detected bounding-box. In medical imaging, sometimes the detector identifies smaller parts of a large or scattered lesion with a large ground-truth bounding-box that motivates the use of IoBB, which is defined by

$$IoBB = \frac{Area\ of\ Overlap}{Area\ of\ Predicted\ B{-}Box} \tag{5.2}$$

Due to the relatively low spatial resolution of heatmaps (16 x 16 heatmap) compared to the original image dimension (512 x 512), a generated bounding box is often bigger than a ground truth bounding box. This suggests defining a proper localization by requiring either IoU > T(IoU) or IoBB > T(IoBB), where T(IoU) is the threshold value of IoU and T(IoBB) is the threshold value IoBB. The threshold selected for IoBB is ∈ [0.25] and threshold selected of IoU ∈ {0.1, 0.2, 0.3}. These thresholds are selected based on experimental results and to compare against base publication [1]. Also, we have computed the ratio of non-overlap (i.e. is IoU = 0). The ratio of non-overlap is the number of images that do not have an intersection between the predicted bounding box and ground truth bounding box divided by the total number of images. The non-overlap ratio describes when neither bounding boxes overlap with each other. This metric also gives the percentage of prediction that has an IoU equal to zero.

5.3 Localization Results of Class Activation Map

For the class activation map method (CAM), the weights w^c of a class is taken directly from the prediction layer (which is a linear layer). The weight vector of class *c* of the prediction layer is accessed using a Pytorch function: $'w^c =$ ResNet18.fc.parameters()[c]'. The feature map from the target layer is obtained while forward propagation through DCNN. This intermediate output (i.e. feature map from the target_layer) can be accessed by Pytorch function 'target_layer.register_forward_hook()'. This function saves a feature map generated by the target layer in each forward pass. Finally, a heatmap is created by summing the product of the feature map and the weights, as seen in (5.3) below.

$$M_c^{\mathrm{CAM}}(x,y) = \sum_k w_c^k f_k(x,y) \tag{5.3}$$

	ResNet18 with CAM					
	Mean IOU	Non-Overlap	IOU greater than 0.1	IOU greater than 0.2	IOU greater than 0.3	IOBB greater than 0.25
Atelectasis	0.134	0.311	0.478	0.3	0.139	0.578
Cardiomegaly	0.381	0	0.966	0.856	0.664	0.87
Effusion	0.214	0.248	0.634	0.503	0.314	0.614
Infiltration	0.245	0.187	0.715	0.553	0.333	0.667
Mass	0.158	0.235	0.506	0.329	0.224	0.394
Nodule	0.042	0.38	0.114	0.025	0	0.038
Pneumonia	0.154	0.192	0.467	0.308	0.192	0.608
Pneumothorax	0.087	0.5	0.214	0.173	0.133	0.357
Macro-average	0.1768	0.2566	0.5117	0.3808	0.2498	0.5157

Table 14: Pathology localization results with Class Activation Map for eight disease classes.

	ResNet18 with CAM
Micro-average-IOU	0.189
Micro-average-IOBB	0.462
Micro-average-Non-Overlap	0.256

Table 15: Micro-average performance metrics for localization with CAM.

The localization performance attained with class activation map (CAM) using ResNet18 on the ChestX-ray8 dataset is summarized in Table 14. The micro-average of IoU, IoBB and Non-Overlap is shown in Table 15. From Table 14, it is evident the CAM can detect the location of some diseases without any supervision of disease location (or explicit disease labels for training) in the X-ray images. Cardiomegaly has the highest localization performance compared to other

diseases. All predicted bounding boxes of Cardiomegaly have an intersection with the ground truth location and have a mean-IoU of 0.381. Diseases like Nodule and Pneumothorax have mean-IoU less than 0.1. The CAM model achieves a Micro-Average for Non-Overlap of 0.256, which means that 75 percent of predicted bounding boxes have some intersections with ground truth-bounding boxes. The CAM method achieves a Micro-Average IoU of 0.189.

5.4 Localization Results of GradCAM

In contrast to the class activation map (CAM), with GradCAM, the weights w^c of a class is computed as shown in (5.4) below. The gradient of class probability with respect to feature map ($\frac{\partial y^c}{\partial f_{x,y}^k}$) is computed by backpropagating gradients to the target layer from where we output the feature map. Similar to extracting intermediate output (i.e. feature map) from a particular layer, we can also get the gradient at a particular layer while perform backpropagating. The gradient at a target layer can be extracted using a PyTorch function 'targetlayer.register_backward_hook()', which stores the gradient at that particular layer. Finally, the heatmap can be computed as seen in (5.5) below

$$w_k^c = \frac{1}{z} \sum_{x,y} \frac{\partial y^c}{\partial f_{x,y}^k} \tag{5.4}$$

$$M_c^{GradCAM}(x,y) = ReLU(\sum_k w_c^k f^k) \tag{5.5}$$

	ResNet18 with GradCAM					
	Mean IOU	Non-Overlap	IOU greater than 0.1	IOU greater than 0.2	IOU greater than 0.3	IOBB greater than 0.25
Atelectasis	0.127	0.15	0.567	0.2	0.094	0.767
Cardiomegaly	0.462	0	0.986	0.952	0.863	0.973
Effusion	0.21	0.144	0.673	0.451	0.275	0.758
Infiltration	0.253	0.073	0.748	0.537	0.341	0.821
Mass	0.123	0.2	0.341	0.247	0.153	0.441
Nodule	0.026	0.291	0.025	0	0	0
Pneumonia	0.148	0.408	0.483	0.333	0.208	0.475
Pneumothorax	0.094	0.398	0.265	0.153	0.102	0.469
Macro-average	0.1803	0.208	0.511	0.3591	0.2545	0.588

Table 16: Pathology localization results with GradCAM for eight disease classes.

	ResNet18 with GradCAM
Micro-average-IOU	0.195
Micro-average-IOBB	0.578
Micro-average-Non-Overlap	0.208

Table 17: Micro-average performance metrics for localization with GradCAM.

The localization performance achieved with the GradCAM using ResNet18 on the ChestX-ray8 dataset is reported in Table 16. The micro-average of IoU, IoBB and Non-Overlap of GradCAM method is shown in Table 17. The macro-average IoU of GradCAM and CAM are comparable; however, the micro-average IoU of GradCAM is slightly higher than CAM because of the high imbalance in ChestX-ray8 dataset. The non-overlap of GradCAM is improved from 0.256 to 0.208 (the lower, the better) than the CAM method. From the experimental results, we could conclude that there is not much improvement in the localization performance for GradCAM compared to CAM.

5.5 Localization Results of GradCAM++

Using GradCAM++, weights are computed using (5.6) below. Feature map f^k and gradient of predication score of class with respect to the feature map $\left(\frac{\partial y^c}{\partial f_{i,j}^k}\right)$ is computed in a way similar to that of GradCAM, and (5.7) shows the computation of a heatmap.

$$w_k^c = \sum_{i,j}\left[\frac{\frac{\partial^2 y^c}{(\partial f_{i,j}^k)^2}}{2.\frac{\partial^2 y^c}{(\partial f_{i,j}^k)^2}+\sum_a\sum_b f_{a,b}^k\left\{\frac{\partial^3 y^c}{(\partial f_{i,j}^k)^3}\right\}}\right].relu\left(\frac{\partial y^c}{\partial f_{i,j}^k}\right) \quad (5.6)$$

$$M_c^{GradCAM++}(x,y) = \sum_k w_k^c f^k \quad (5.7)$$

	ResNet18 with GradCAM ++					
	Mean IOU	Non-Overlap	IOU greater than 0.1	IOU greater than 0.2	IOU greater than 0.3	IOBB greater than 0.25
Atelectasis	0.156	0.161	0.589	0.328	0.161	0.794
Cardiomegaly	0.46	0	1	0.966	0.87	0.993
Effusion	0.235	0.137	0.686	0.523	0.327	0.765
Infiltration	0.302	0.033	0.878	0.683	0.423	0.862
Mass	0.135	0.106	0.424	0.271	0.153	0.493
Nodule	0.033	0.266	0.051	0	0	0

Pneumonia	0.283	0.083	0.833	0.683	0.433	0.817
Pneumothorax	0.109	0.357	0.327	0.214	0.112	0.49
Macro-average	0.2141	0.1428	0.5985	0.4585	0.3098	0.6517

Table 18: Pathology localization results with GradCAM++ for eight disease classes.

	ResNet18 with GradCAM++
Micro-average-IOU	0.231
Micro-average-IOBB	0.629
Micro-average-Non-Overlap	0.142

Table 19: Micro-average performance metrics for localization with GradCAM++.

The localization performance achieved with the GradCAM++ using ResNet18 on the ChestX-ray8 dataset is reported in table 18. The micro-average of IoU, IoBB and Non-Overlap of GradCAM++ method is shown in table 19. With GradCAM++, the localization performance is increased for almost all the diseases; this can be understood from the IoU. For Cardiomegaly disease, 87 percent of the samples have IOU greater than 0.3. The IoU of Pneumonia has almost doubled compared to GradCAM and CAM. Both micro-average IoU and macro-average IoU of GradCAM++ are increased compared to GradCAM and CAM. The Micro-Average-Non-Overlap is improved to 0.142 (lower the better) from 0.208 achieved by GradCAM. Micro-Average-IOU of GradCAM++ is improved to 0.231 from 0.195 achieved by GradCAM. This shows the superiority of GradCAM++ over other weakly localization methods.

5.6 Discussion and Comparison

Figure 34 shows four different *X*-ray images with heatmaps and associated bounding boxes generated by CAM, GradCAM and GradCAM++, respectively. Figure 34(a) shows that the three weakly supervised localization methods were able to locate the disease in the images correctly, where CAM appears to provide better IoU than the other two methods due to the precise overlap of bounding boxes. However, the heatmaps generated by GradCAM and GradCAM++ are better than CAM because the highlighted regions from GradCAM and GradCAM++ are inside the ground-truth bounding boxes. This seems to suggest that bounding box generation is not very effective. In Figure 34(b), we observe that CAM and GradCAM are unable to detect every disease region, while GradCAM++ is able to locate most disease regions. Figure 34(c) indicates that all three localization methods generated a bounding box where the actual disease was not present,

and at the same time, all three models were able to detect the original location of the disease. From Figure 34(d), we observe that CAM fails to work as it generates multiple erroneous bounding boxes while GradCAM++ is able to highlight disease regions.

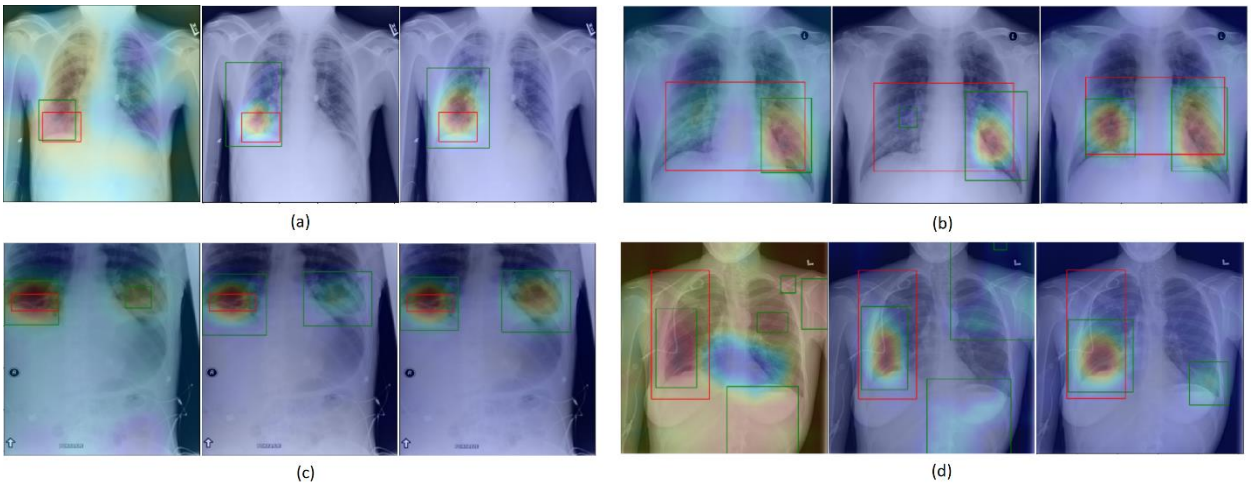

Figure 34: Sample examples of heatmaps and bounding boxes generated by CAM, GradCAM and GradCAM++. For each X-ray image, heatmaps generated with three methods are shown, the left-most heatmap is generated by CAM, the middle heatmap is generated by GradCAM, and the rightmost heatmap is generated by GradCAM++.

All three localization methods studied here have been most successful when it comes to localizing Cardiomegaly. Cardiomegaly is also the category that reaches the very best AUC-ROC for classification. Especially with GradCAM++ (table 18), it has 0 non-overlap ratio, which means that all the predicted bounding boxes of Cardiomegaly have at least some intersection with ground truth bounding boxes. Moreover, 87 percent of the cases of Cardiomegaly have IoU greater than 0.3. Cardiomegaly is an enlarged heart, and the bounding box of Cardiomegaly is over the heart, which means the location of Cardiomegaly always remains the same in all the X-ray images. Compared to other disease locations of Cardiomegaly is almost constant or always lies over the heart. This could be the reason the Cardiomegaly is easy to localize.

We note that the two classes, namely Nodule and pneumothorax, that perform the worst have tiny bounding boxes. Also, it seems that both these diseases can occur anywhere within the lungs. This could be the reason for the localization of Nodule and Pneumothorax are difficult. The disease with the worst classification AUC, Infiltrate, performs well in localization tasks with a mean IoU of 0.302, which is greater than the average mean IoU. This implies that the successful generation of good bounding boxes depends not only on the model's classification

strength but also on the bounding box's size, shape, and placement. Some good results obtained for the eight diseases using weakly supervised localization are exhibited in Figure 35.

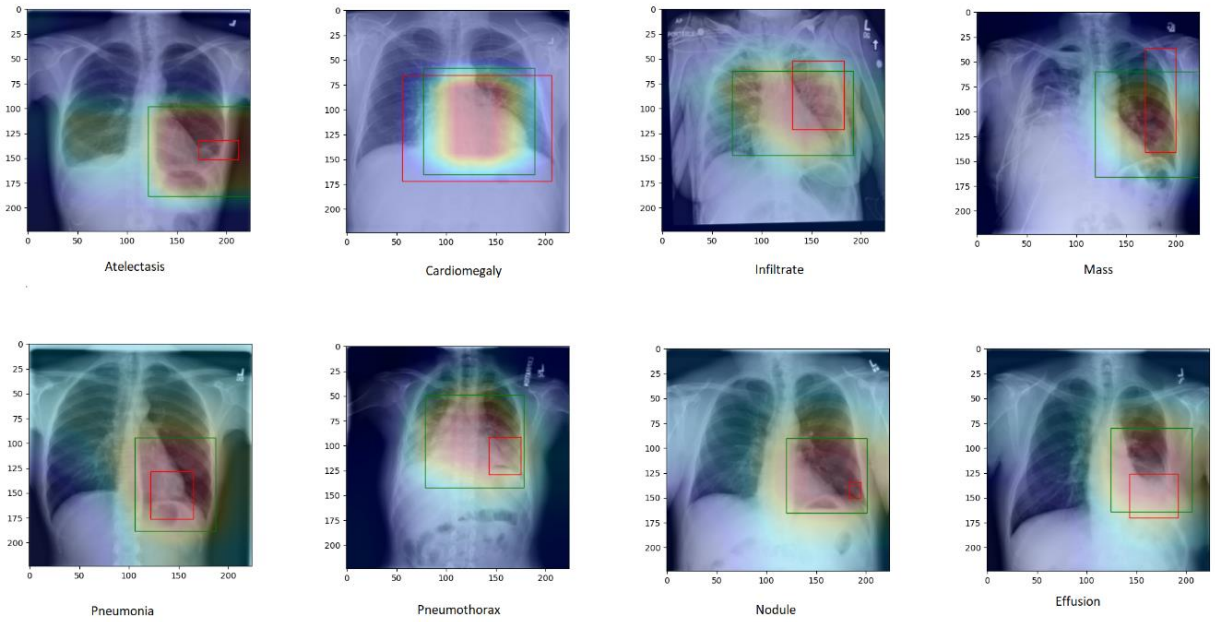

Figure 35: Weakly supervised location of eight diseases with GradCAM++.

	Atelectasis	Cardiomegaly	Effusion	Infiltration	Mass	Nodule	Pneumonia	Pneumothorax	Macro-average
T(IOU) > 0.1									
Proposed	0.589	**1**	**0.686**	**0.878**	**0.424**	0.051	**0.833**	0.327	**0.598**
Baseline [1]	**0.68**	0.938	0.66	0.707	0.4	**0.1392**	0.633	**0.377**	0.566
T(IOU) > 0.2									
Proposed	0.328	**0.966**	**0.523**	**0.683**	**0.271**	0	**0.683**	0.214	**0.458**
Baseline [1]	**0.472**	0.684	0.450	0.479	0.258	**0.050**	0.350	**0.230**	0.371
T(IOU) > 0.3									
Proposed	0.161	**0.870**	**0.327**	**0.423**	**0.153**	0	**0.433**	0.112	**0.309**
Baseline [1]	**0.244**	0.458	0.300	0.276	0.152	**0.037**	0.166	**0.132**	0.220

Table 20: Comparison of performance with baseline original paper [1].

Table 20 provides comparisons of localization performance of the original paper [1] to our results. The results shown in the table are the best results we obtained in the experiments on the localization of disease using the GradCAM++ method. It appears that the accuracy of the baseline paper [1] and our method are comparable when the IoU threshold is 0.1 (T(IoU) > 0.1). However,

our method's localization accuracy is significantly higher than the baseline paper [1] when the IoU thresholds were set to 0.2 and 0.3. When considering individual diseases, our method achieved higher accuracy for all the disease except for Atelectasis and Nodule. To conclude, we have shown that our method has a higher average localization performance relative to the baseline method [1] in all the IoU thresholds.

Chapter 6 Conclusion

In this report, we have focused on the weakly supervised classification and localization of chest-*X*-ray images. Various deep neural architectures for classification and various weakly supervised localization methods have been described and compared in detail in this report.

In the first part of the project, we have trained different deep neural architectures to predict diseases in chest *X*-ray images. We have used the publicly released ChestX-ray8 dataset [1], including the training set, validation set, and test set. Specifically, we have used DCNNs like ResNet18 and DenseNet121 to train chest *X*-ray images and examined different scenarios like neural networks loaded with pre-trained weights and different input *X*-ray image resolutions. Then we have evaluated the performance of the models and various techniques used in each experiment. The best results achieved for *X*-ray image classification obtained in the project have been compared to the baseline paper [1], see Table 12 where our model has demonstrated an improvement by nine percent in terms of the evaluation metric used.

In the second part of the project, we have developed a weakly supervised localization method to locate the diseases in the *X*-ray images only with the image-level labels and any information about the locations of the disease is not provided for training. Specifically, we have developed weakly supervised localization methods like Class activation map, GradCAM and GradCAM++ to locate the diseases in chest *X*-ray images. We have presented a method to generate bounding boxes from heatmap generated and evaluated our localization methods on ground truth bounding box labels of chest *X*-ray images. We have also compared and reported the localization performance of the three weakly supervised localization methods studied. Finally, our model with the best localization performance is compared to the baseline paper [1], and our method is found better performed than the original paper in locating diseases in *X*-ray images.

Bibliography

[1]. X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "ChestX-Ray8: Hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 3462-3471, doi: 10.1109/CVPR.2017.369. Available: https://arxiv.org/pdf/1705.02315.pdf

[2] A. R. Aronson and F.-M. Lang, "An overview of MetaMap: historical perspective and recent advances," *Journal of the American Medical Informatics Association*, 17(3):229–236, May 2010.

[3] R. Leaman, R. Khare, and Z. Lu, "Challenges in clinical natural language processing for automated disorder normalization," *Journal of Biomedical Informatics*, 57:28–37, 2015.

[4] J. Irvin, P Rajpurkar, M Ko, Yifan Yu, S Ciurea-Ilcus, C Chute, H Marklund, B Haghgoo, R Ball, K Shpanskaya, J Seekins, David A. Mong, S S. Halabi, Jesse K. S, Ricky J, David B. L, Curtis P. L, Bhavik N. P, Matthew P. L, and Andrew Y. Ng. "CheXpert: A Large Chest Radiograph Dataset With Uncertainty Labels and Expert Comparison". Proceedings of the AAAI Conference on Artificial Intelligence 33 (01):590-97, 2019. Available: https://arxiv.org/pdf/1901.07031.pdf

[5] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva and A. Torralba, "Learning Deep Features for Discriminative Localization," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 2921-2929, doi: 10.1109/CVPR.2016.319. Available:https://arxiv.org/pdf/1512.04150.pdf

[6] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, 2017, pp. 618-626, doi: 10.1109/ICCV.2017.74. Available:https://openaccess.thecvf.com/content_ICCV_2017/papers/Selvaraju_Grad-CAM_Visual_Explanations_ICCV_2017_paper.pdf

[7] A. Chattopadhay, A. Sarkar, P. Howlader and V. N. Balasubramanian, "Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks," *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Tahoe, NV, 2018, pp. 839-847, doi: 10.1 109/WACV.2018.00097. 1710.11063v3.pdf (arxiv.org)

[8] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," Available: https://www.cs.toronto.edu/~fritz/absps/reluICML.pdf

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, " ImageNet classification with deep convolutional neural networks," In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12). Curran Associates Inc., Red Hook, NY, USA, 1097–1105, 2012. Available:http://www.cs.toronto.edu/~kriz/imagenet_classification_with_deep_convolutional.pdf

[10] C. Szegedy *et al*., "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594. Available:https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43022.pdf

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556 (2014): 1409.1556.pdf (arxiv.org)

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90. 1512.03385.pdf (arxiv.org)

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," In *CoRR*, abs/1502.03167 (2015) 1502.03167.pdf (arxiv.org)

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243. Available:https://arxiv.org/pdf/1608.06993.pdf

[15] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, Wayne E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a backpropagation network," pp. 396–404, 1990. (nips.cc).

[16] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," In *3rd International Conference on Learning Representations*, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[17] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," *ECCV*, abs/1604.01802, 2016.

[18] M. Wang and W. Deng, "Deep face recognition: A survey," arxiv, abs/1804.06655, 2018. Available:https://arxiv.org/pdf/1804.06655v9.pdf

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," CoRR, abs/1312.5602, 2013.

[20] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. 2018. Deep video portraits. ACM Trans. Graph. 37, 4, Article 163 (August 2018), 14 pages. DOI:https://doi.org/10.1145/3197517.3201283

[21] S. Rajeswar, S. Subramanian, F. Dutil, C. J. Pal, and A. C. Courville, "Adversarial generation of natural language," CoRR, abs/1705.10929, 2017.

[22] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res*., 2:45–66, mars 2002.

[23] J. R. Quinlan : Induction of decision trees. Mach. Learn., 1(1):81–106, mars 1986.

[24] B. Neyshabur, S. Bhojanapalli, D. Mcallester, and N. Srebro, "Exploring generalization in deep learning," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds. vol. 30, pp. 5947–5956, 2017.

[25] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with Deep Learning: A Review," arXiv:1807.05511v2,1807.05511.pdf (arxiv.org)

[26] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169. Available:https://arxiv.org/pdf/1311.2524.pdf

[27] J. R. Quinlan. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (March 1986), 81–106. DOI:https://doi.org/10.1023/A:1022643204877,https://dl.acm.org/doi/10.1023/A%3A1022643204877

[28] R. Hecht-Nielsen, Theory of the backpropagation neural network, Neural Networks 1 (Supplement-1) (1988) 445–448.

[29] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, pp. 2278–2324, Nov. 1998.

[30] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, “Recent advances in convolutional neural networks”, Pattern Recognition, vol. 77, pp. 354-377. Available:https://doi.org/10.1016/j.patcog.2017.10.013

[31] W.-S. Lu, Optimization for Machine Learning, Department of Electrical and Computer Engineering, University of Victoria.

[32] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT, 2016. Available:http://www.deeplearningbook.org

[33] M. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.

[34] M. Oquab, L. Bottou, I. Laptev and J. Sivic, "Is object localization for free? - Weakly-supervised learning with convolutional neural networks," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 685-694, doi: 10.1109/CVPR.2015.7298668

[35] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848

[36] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” In 3rd International Conference for Learning Representations, San Diego, 2015.

[37] M. Tan and Q. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” Proc. 36th International Conference on Machine Learning, Available from http://proceedings.mlr.press/v97/tan19a.html

[38] Y. X. Tang, Y. B. Tang, Y. Peng, et al. “Automated abnormality classification of chest radiographs using deep convolutional neural networks,” npj Digit. Med. 3, 70 (2020). https://doi.org/10.1038/s41746-020-0273-z

[39] P. H. O. Pinheiro and R. Collobert. “From image-level to pixel-level labeling with Convolutional Networks,” 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): 1713-1721.

[40] Youden's J statistic Available:https://en.wikipedia.org/wiki/Youden%27s_J_statistic#:~:text=Youden's%20J%20statistic%20(also%20called,probability%20of%20an%20informed%20decision.

[41] . Liu, Wei, Anguelov, Dragomir, Erhan, Dumitru, Szegedy, Christian, Reed, Scott E., Fu, Cheng-Yang and Berg, Alexander C.. "SSD: Single Shot MultiBox Detector.." Paper presented at the meeting of the ECCV (1), 2016.

[42] Marcus D. Ruopp, Neil J. Perkins, Brian W. Whitcomb, and Enrique F. Schisterman*, Youden Index and Optimal Cut-Point Estimated from Observations Affected by a Lower Limit of Detection

Available:https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2515362/

[43] Rajul Parikh, MS, Annie Mathai, MS, Shefali Parikh, MD, G Chandra Sekhar, MD, and Ravi Thomas, MD ,Understanding and using sensitivity, specificity and predictive values

[44] Classification: ROC Curve and AUC
Available:https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

[45] Intersection over Union (IoU) for object detection
Available: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[46] Wang, H. et al. Global, regional, and national life expectancy, all-cause mortality, and cause-specific mortality for 249 causes of death, 1980–2015: a systematic analysis for the globalburden of disease study 2015. Lancet 388, 1459–1544 (2016).

Available::https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(16)31012-1/fulltext

[47] American Lung Association. Trends in lung cancer morbidity and mortality. Available:https://www.lung.org/assets/documents/research/lc-trend-report.pdf (2014).

[48] Dunnmon, J. A. et al. Assessment of convolutional neural networks for automated classification of chest radiographs. Radiology 290, 537–544 (2019).

[49] Chest X-rays Overview, Available:https://www.mayoclinic.org/tests-procedures/chest-x-rays/about/pac-20393494

[50] A. Setio, F. Ciompi, G. Litjens, P. Gerke, C. Jacobs, S. van Riel, M. Wille, M. Naqibullah, C. Snchez, and B. van Ginneken. Pulmonary nodule detection in ct images: False positive reduction using multi-view convolutional networks. IEEE Trans. Medical Imaging, 35(5):1160–1169, 2016.

[51] H. Roth, L. Lu, A. Farag, H.-C. Shin, J. Liu, E. B. Turkbey, and R. M. Summers. Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation. In MICCAI, pages 556–564. Springer, 2015.

[52] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3):211–252, 2015

[53] A Gentle Introduction to Threshold-Moving for Imbalanced Classification
Available: https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/