2021

Classification of X-rays images using Deep Convolutional Neural Network: COVID-19

Bore, Laban Kipchirchir
Strathmore Institute of Mathematical Sciences
Strathmore University

Classification of X-rays Images Using Deep Convolutional Neural Network: COVID-19

Strathmore
UNIVERSITY

Laban Kipchirchir Bore

The thesis presented in fulfillment of the academic requirement for the degree of Masters of Statistical Science (Statistics) of Strathmore University

September 2021

Declaration and Approval

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Laban Kipchirchir Bore

Signature .....

Date 27/09/2021

Approval

The thesis of Laban Kipchirchir Bore was reviewed and approved by the following:

Dr. Collins Odhiambo,
Senior Lecturer, Institute of Mathematical Sciences,
Strathmore University

Dr Godfrey Achono Madigu,
Dean, Institute of Mathematical Sciences,
Strathmore University

Dr. Bernard Shibwabo,
Director of Graduate Studies,
Strathmore University

Dedication

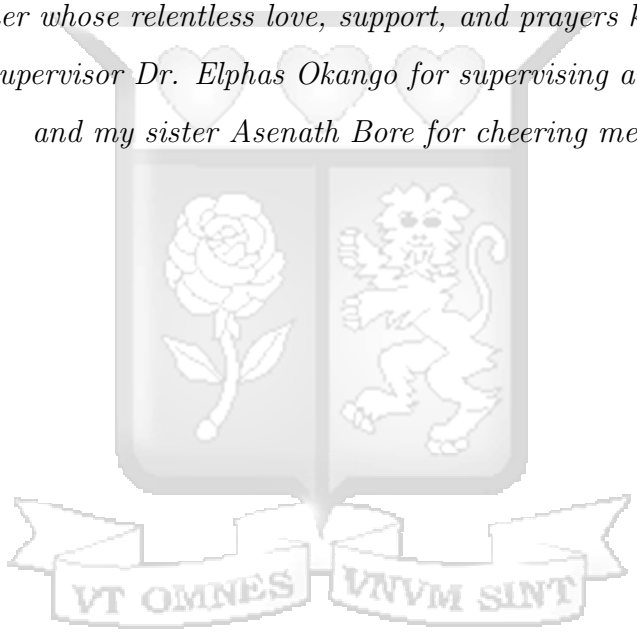
This thesis is dedicated to the memory of my father, Symon Kangogo Chebore, who played a critical role in my education; offered support and encouragement during my early education.

Special thanks to...

...my wife through challenging and glorious moments, your kindness, love, and support have always been present during these critical times of my life.

...my mother whose relentless love, support, and prayers kept me going.

...finally, to my supervisor Dr. Elphas Okango for supervising and guiding the thesis and my sister Asenath Bore for cheering me.



Acknowledgment

First, I wish to extend my special thanks to my supervisor Dr. Elphas Okango who guided and supervised me in this project. I would also like to thank Dr. Collins Odhiambo for his advice on this project and supervision.

I wish to acknowledge Strathmore University, Institute of Mathematical Sciences (SIMS) for the the world-class education, practical and technical knowledge.

Abstract

The increased amount of labeled X-ray image archives has triggered increased research work in the application of statistics, machine learning, deep learning, and computer vision across the different domains. The fresh studies on the application of deep transfer learning (60) CNN to detect and classify few COVID-19 datasets have had major success. COVID-19 dataset has been collected since the outbreak of the COVID-19 viruses in quarter four of 2019. COVID-19 virus confused the diagnosis, treatment, and care of patients because there is no cure and the virus mutates into different fatal variants. This has led to thousands of people dying, increased admission into hospital beds, ICU, and other health facilities. Hundreds of thousands of new infection cases are reported daily across the world. The overburdening of the health system by the COVID-19 virus has caused access to other health services difficult in the under-served world (89). Traditionally, medical doctors carry several tests such as full blood count tests to ascertain if the body is fighting certain pathogens, sputum tests, and chest X-rays. Doctors will examine patients' medical history, carry physical exams such as listening to the lungs with astethoscope for abnormal crackling sounds. The success of this traditional diagnosis process is dependants on the doctors' experience, skills. quality of X-ray images and the availability of patient's historical records. This is almost unattainable and unsustainable in the under-served countries in Africa. The motivation of this paper is to complement the traditional diagnosis and analysis of chest X-ray images by introducing machine classification approaches and state-of-the-art deep residual network ResNet18 (14, 35). According to WHO (58), diagnosis is a process and requires classification steps to inform research, health policies, and care of the patients. An alternative definition is a "pre-existing set of categories agreed upon by the medical profession to designate a specific condition" (43).

We applied statistical learning model to separate and classify all the X-Rays images with patchy areas into one distinct class for further research, examination, analysis, and care of the patients. The observed white patchy areas in our X-Rays images was our statistical variables of interest in classifying Chest X-Rays images into COVID-19 and

non-COVID-19, fig 3.2. In addition, the final model can be replicated in other non-covid datasets and extended to other related classification tasks. Deep CNN classification model(ResNet18) as a subfield of non-parametric statistics was used for classifying and predicting COVID-19 positive images. The datasets used were COVID-19 positive (184 cases) and the COVID-19 negative cases (5000) were aggregated from different sources. The COVID-19 negative cases was from 10 disease categories (Pneumonia, Pneumothorax, Lung opacity, Fracture, Atelectasis, Edema, pleural, etc). The finetuned deep CNN model (ResNet18) performed significantly with precision (87.5%), sensitivity (75%) and specificity (99.8%). Rerunning the model using larger datasets by adding noise through data augmentation demonstrated sensitivity (90%) and specificity (100%). Hence, when more dataset is fed into the neural model, the classification performance such as precision, AUC and recall improves significantly. This classification model can be used to aid radiologists or medical practitioners in chest X-ray image diagnosis and treatment (59) by categorization, diagnosis, detection, and prediction. Further extension of this research work will focus on using larger COVID-19 or non-COVID-19 datasets with more focus on systematic review around data acquisition, data certification, model development and pitfalls, and explanation construction (39).

Contents

Abbreviations

ROC	Receiver Operating Characteristic
ResNet18	Residual Network
AUC	Area Under the Curve
CI	Confidence Intervals
CNN	Convolution Neural Network
DL	Deep Learning
ReLu	Rectified Linear Unit (ReLU)
ANN	Artificial Neural Networks
Res	Residue
CXR	Chest X-Rays
CT	Computed Tomography
MRI	Magnetic Resonance Imaging
SVM	Support Vector Machine
PCANet	Principal Component Analysis Network
KHIS	Kenya Health Information System
KNBS	Kenya National Bureau of Statistics
WHO	World Health Organization
SOR	Society of Radiography in Kenya
NIH	National Institutes of Health
Kag	Kaggle
Img	Image
Bir	Bird
etc	et cetera

Chapter 1

Introduction

This chapter introduces the background of the study, problem statement, research objectives, and significance of the research.

1.1 Background to the study

In 1895 professor Wilhelm Rontgen discovered the concept of x-rays (32) which is based on passing ionized radiation through the patient's body and projecting its images on a photosensitive plate. The body tissues are detected on the plate which will display the presence of abnormalities. Nuclear medicine where patients were infused with radionucleotides in combination with pharmaceutical was introduced in 1950 (32). This concept records images and detects medical pathogens using a gamma camera. At around 1970 CT scan and MRI were developed which was build on magnetic nuclear technology. Strong magnetic forces are directed on the body which can display the alignment of protons in cells. This is further investigated and the problem in body tissue is differentiated by a physician. To date, there has been improvement in medical imaging and accuracy in diagnosing a medical condition. As a result of these technological successes, medical practitioners are performing less exploratory surgery and analysis.

With recent improved high precision technology, availability of large datasets, fast computing power, and the need to quickly make accurate diagnosis and decision, radiologists are under immense decision fatigue to provide instant reliable examination reports. The challenges of detecting medical problems from high-resolution images and examination with the human natural eye may be partially met by a highly experienced photographer and radiologist. The physician looks at the images with their physical eye for abnormalities. This is not always the case when it comes to diagnosing emerging challenging diseases or new variant of disease evidence in the image. The practitioners

with their level of skills may fail to diagnose or may misinterpret the anomalies displayed in the image hence produce incorrect results.
A recent study on artificial neural network(ANN)-machine learning suggests that the technology performs better in visual and auditory recognition tasks than the human eye (48).

The technology has witnessed a successful landmark in its application to learning rules instead of being programmed with the rules and find the underlying statistical structure in performing specific tasks. Some of the applications are in the classification of images, associating photos to specific tagging tasks, and computer vision. Image classification problem cannot be solved by machine learning models because it is complex and the intensity of its information is represented by thousands of millions of pixels. The image classification task cannot be performed by classical statistical methods such as Bayesian analysis.

This research paper will focus on the application of non-parametric statistical classification and analysis of images in the medical field using Deep Convolution Neural Network as compared to a shallow Convolution Neural network which is built by one or two layers of neural network.

1.2 Problem Statement

Experienced and skilled medical practitioners or radiologists in the developing world find challenges in accurately diagnosing, classifying, and interpreting the patients X-rays image datasets. This is due to poor quality image and situations complicated by the development and mutation of the existing or rare disease. The underlying disease changes and damages in the human lungs can complicate the diagnosis process which can lead to misinterpretation or prolong delay in disease diagnostics. In 2019 WHO projected a shortfall of 18 million health workers towards accelerating universal health coverage in lower and middle-income countries by 2030 in line with achieving health Sustainable Development Goals. The shortfall in the health system is worsened by the lack of quality health and diagnosis equipment and limited experienced and unskilled

recruits.

According to the Society of Radiography in Kenya, there are 1070 registered radiographers are serving a country's medical system with a population of over 50 million people who are exposed to various pathogens and need body imaging and diagnosis attention. The radiographers and radiologists have to go through clinical training to keep abreast with the advanced imaging technology and also to provide accurate diagnosis, quality imaging services, and build competencies.

I believe the application of deep CNN in the analysis of medical big data can be useful in extracting important information, performing automatic abnormality detection, producing varied diagnoses, and preparing introductory radiology reports. The deep CNN-based analytical framework in the practice of radiology will complement the irreplaceable and remarkable physician skills. The radiologist's work will improve accurate diagnosis and analysis, decision making, and interpretation of underlying body-tissues conditions in chest X-ray images. Dr. Watson is one of the IBM radiology applications which factors in some of the functions discussed in this paper (30).

To aid the practitioners' in accurate diagnosis, making decisions, and interpretation of body images, we will build and deploy a classification model to detect, discriminate, and predict chest X-ray images into a different distinct class.

1.3 Research Objectives

The purpose of this study is to apply the deep ResNet-18 CNN model to classify 5000+ publicly labeled chest X-ray images into their distinct classes for ease of differentiation, diagnosis, and interpretation by the radiologist and medical professions using the transfer learning version of non-parametric deep convolution neural network. The learned knowledge from the pre-trained ResNet model will be applied to the target X-ray images to solve the classification problem which would have been time-consuming in constructing the deep ResNet model from the scratch due to insufficient certified COVID-19 samples.

1.4 Significance of research

This research work will aid the practitioners to interpret and classify the target medical images into different categories in a clinical setting for further disease diagnosis and analysis, examination, and treatment of patients. The statistical model will be packages as a software and used in hospital to classify X-rays images into different groups of pathogens. The classification model will achieve the following:

- It will improve accuracy in image diagnosis, analysis and interpretation in hospitals.
- The diagnostic results will support decision making process by informing patient care, policies, research & development in the field of health and healthcare.
- Detect, discriminate and predict chest X-Rays images into distinct group.

Chapter 2

Literature review

2.1 Introduction

Diagnostic analysis and process are one of the important steps in the examination of patients in clinics and hospitals. The diagnosis results will inform patient care, policies, research, and development. According to Jutel (43) diagnosis is a "pre-existing set of categories agreed upon by the medical profession to designate a specific condition". This can be viewed as a process and classification task (56). The clinical decision-making process depends on the outcome of the diagnosis. The diagnosis should be accurate and timely to provide correct doses and treat the patient's disease. Diagnosis is a process that involves patients data collection, clinical analysis to conclude on patient's health conditions. In most cases, the process requires cognitive skills, collaboration, and concentration around the patient.

Typically the diagnosis process is as follows: the patients seek help from a clinic after experiencing certain symptoms. The physician in the health system collects patients data, aggregate, interpret and determine the most probable workable diagnosis. The understanding of the patients' health problems involves gathering data by interviewing, conducting a physical examination, diagnostic tests, and consultation with relevant experts in the health system (17). The tasks are carried out in a feedback loop mechanism; the health information technology and other tech tools are used in the diagnostic process. All components involved in the diagnosis process interact (23). During this process depending on the outcome of the history and patients interviews; the doctor might direct the patient to the radiology labs for body scanning. Pictures of the targeted body parts are taken and return to the doctor for a further physical exam.

Image testing techniques such as CXR and CT scan are crucial in respiratory disease diagnoses such as COVID-19 and pneumonia. Significant steps have been made in convolution neural network to classify the medical image as a result of large annotated chest

X-rays datasets. CNN method provides representative learning for quality annotated images.

The fast research & development of digitized medical image and storage infrastructural technologies, medical image diagnosis, and understanding by doctors and computers is a practiced topic in statistical/machine learning discipline and application to specific problems (6). Previous research publication and work(83) has been used to solve the image classification problem of which we will group into traditional and deep methods. The traditional methods include low features such as texture, edge, color, and SVM. The application of the deep learning approach to classify images is discussed on deep and shallow CNN to classify lung image patches. A lot of medical image labeling work has been done in creating fast and accurate annotated images which have been labelled according to different specific pathology category (82). Images produced across hospitals and regions may vary in quality, features, color, shape, and textures. With the emergence of high-resolution image scanning technologies, the use of traditional methods to classify different categories of the medical image has proved inefficient(53). Traditional method of image classifications classifies images based on color, textures, and shape (18). These traditional methods give features that generally describe the background of the image in terms of color and texture. Celebi was able to extract features such as color, texture, and shape by feeding the images into the SVM algorithm (24). This achieved sensitivity (93%) and specificity (92%). Support vector methods cons is its performance not consistent, constructing them is slow and also selecting and extracting features is time-consuming (84). In this paper, we will advance from the descriptive traditional method of classifying images to gaining more clarity on what the image is and what falls under which group. The low traditional features are of little interest to this research work. The advancement of deep learning models in computer science and statistics has been boosted by the availability of powerful servers and applications. The technique has been used to address non-medical and medical images. The theoretical concept of the deep learning framework was coined by Hinton et al.(29).

Since 2006, researchers have developed many methods and improved on the existing ones to remedy the challenges faced in training CNN (33). A few notable CNN methods, Alex Krishevsky et al. (5) developed a classical CNN framework that outperforms the

state-of-the-art compared to previous methods of image classification problems. The architectural method for AlexNet is similar to that of LeNet but uses larger parameters and 8 layers to model the 1.2 million ImageNet datasets (69). AlexNet uses five convolutional layers, two fully connected hidden layers, and one fully connected output layer in addition to the ReLU activation function while LeNet uses two convolutional layers, two fully connected hidden layers and one fully connected output layer and sigmoid activation function. The success of the few more lines of AlexNet's implementation methods inspired more effective research works to improve CNN performance. Among the works are ResNet12 (36), SqueezeNet (37) and GoogleNet10. (76). The PCA network model provided a baseline for image face recognition which achieved accuracy (99.8%) on single and not multiple images (25). Other deep learning methods are Visual Geometry Group which is a linear SVM classifiers (13) with accuracy(87.5%), specificity(81%) and sensitivity(93.5%). The successful research studies in this field has attracted different disciplines working in varied domain problems. These approaches and methods was used to solve real-life medical and non-medical image classification challenges. Typically image classification is split into two steps. First is the extraction of features and second is the use of the extracted features to classify the image (73). In a traditional setup, professional doctors use their accumulated years of experience to extract features and to classify the image datasets into different classes (31). This is always complex, time-consuming, and boring, sometimes prone to error depending on the emotional state of the doctor like fatigue. This method is unstable and doesn't produce sufficient repeatable results. The emerging application of medical image classification has advantages of this traditional method (63). There is still research and development in this field to produce reliable and efficient classifiers for further medical diagnosis and study. Doctors can combine the proposed model to classify the image dataset with their prior professional experience.

Chapter 3

Methodology

3.1 Introduction

The use of deep CNN methodology in the previous research work was centered on the movement from shallow to deep network and application of the pre-trained CNN models to new most recent datasets like the COVID-19 patient chest X-ray images. Increasing the depth leads to the extraction of better features about the object with an increase in non-linearity. Non-linearity makes the network more challenging to optimize and easily leads to overfitting.

Due to the limited number of clearly labeled images, unavailability of enough datasets to build the model from scratch, and other technical aspects like memory and computing I opted to transfer learning from the states-of-the-art pre-trained model ResNet18 (36) with a few modifications of the parameters.

3.2 Data Sources

Data from public database NIH Chest-X-ray and COVID-19 2020 database (26) was researched and collected over three months. The dataset with unclear outcomes and poor image quality was archived and only those with physician-diagnosed outcome were used in this study. The X-rays images was collected, grouped, and stored in two folders: the test and training dataset with respective sub-folders labelled 'Covid' and 'Non-covid' as demonstrated in the chart 3.1.

The positive COVID-19 images was certified by a radiologist and previously used in the research paper by Shervin (55) based in Canada. The test and training set set was split in the ratio 40%:60% respectively. The positive COVID-19 images were 184 and split into 100 training covid sets and 84 test covid sets. We retained the certified 184

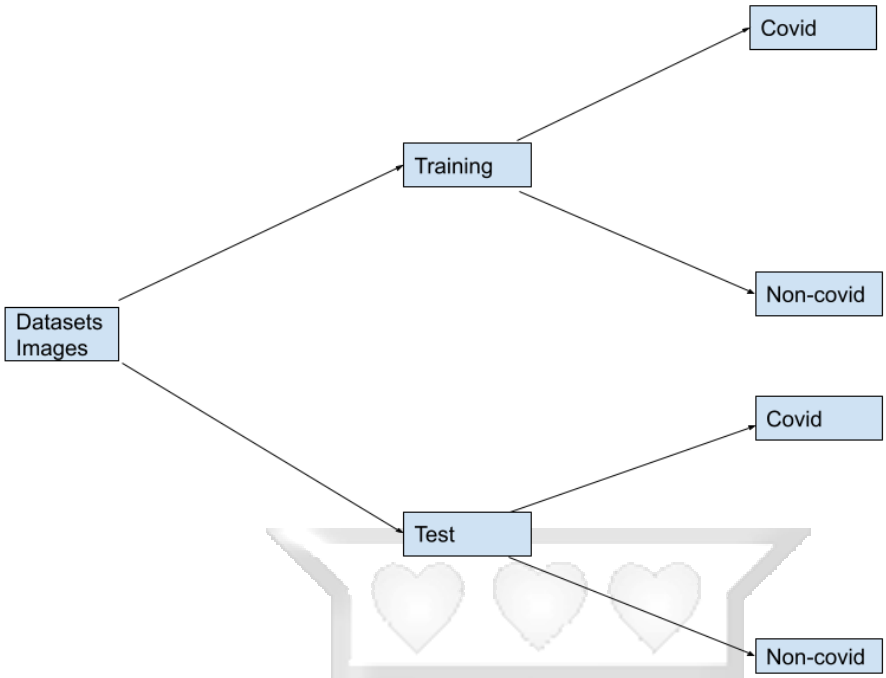


Figure 3.1: Flow chart of data set-up in the computer folders

.

positive COVID-19 datasets. The initial negative COVID-19 datasets was 2400 but after aggregating from different sources we increase the number to 5000 negative-COVID-19. The negative-COVID-19 datasets was aggregated from National Institutes of Health (NIH) and Kaggle. This was split into a 3100 training and 2084 testing set. The negative-COVID-19 datastet contain images of 10 categories (pneumonia, atelectasis, edema, pleural, etc).

3.2.1 Data Preparation

We collected a limited number of certified positive-COVID-19 image cases as a result we cannot use the few data as it is. To remedy this we applied data augmentation procedures i.e flipping or rotating the positive-COVID-19 cases to double our existing database of positive-COVID-19 and optimizes our network (71, 80). We kept the negative-COVID-19 database as it is since it was much richer in numbers. Since the

images were of different resolution we resized them to a 224×224. Color features was not an important object in this study. The image was then fed into a 3 channel giving us the final image input shape of $224 \times 224 \times 3$. We rescale the image pixel values by applying normal standardization procedure to the datasets to save time and to build a stable model.

3.2.2 X-Rays Features and Variables

Figure 3.2 shows labelled Chest X-Rays images with ill-defined peripheral airspace opacity. Three samples of images was taken from patients infected with COVID-19 and the corresponding areas marked as shown below. The pattern seen in the marked areas might be a challenge to identify visually. Since we have limited number of trained experts radiologist such subtle abnormalities or patchy areas can pass undetected.

We apply statistical learning model to separate and predict all the X-Rays images with patchy areas into one distinct class. The observed patchy areas in our X-Rays images was our statistical variables of interest in classifying Chest X-Rays images into COVID-19 and non-COVID-19. Input images is converted into input values and then mapped to output values. White patches in the image takes a higher output values, a signal of abnormalities. The model count such instance and calculate specificity and other model evaluation metrics (47).

The model was used to detect such images, classify them together and output probability values.

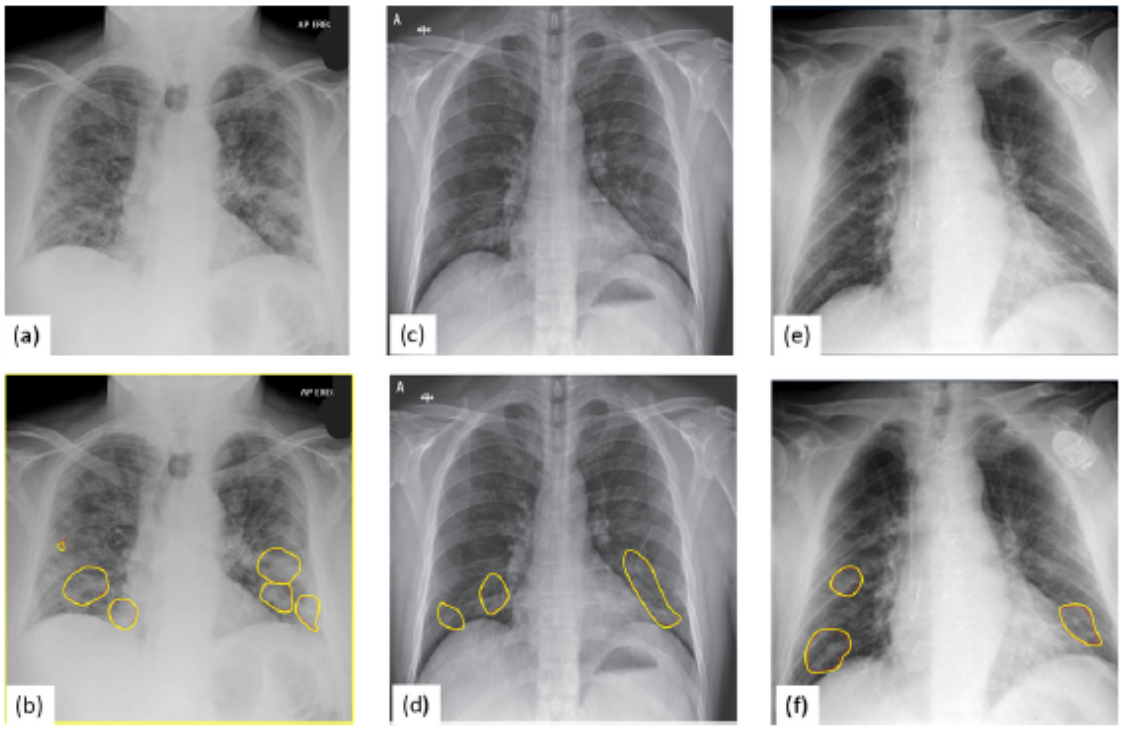


Figure 3.2: Display of patients Chest X-Rays images who were infected with COVID-19 Virus. Image source (55)

.

3.3 Convolution Neural Network Architectural

The human brain is comprised of interconnected billions of neurons. Information is sent from one neuron to another through a process call synapses depending on which part of the brain has been triggered or activated. This information could be of an image, sound, or text. Human beings can discriminate this data by looking at specific features like edges, curves, color, and other attributes. Machines can mirror the same behavior and interpret multiple levels of representation and abstraction using pixel values.

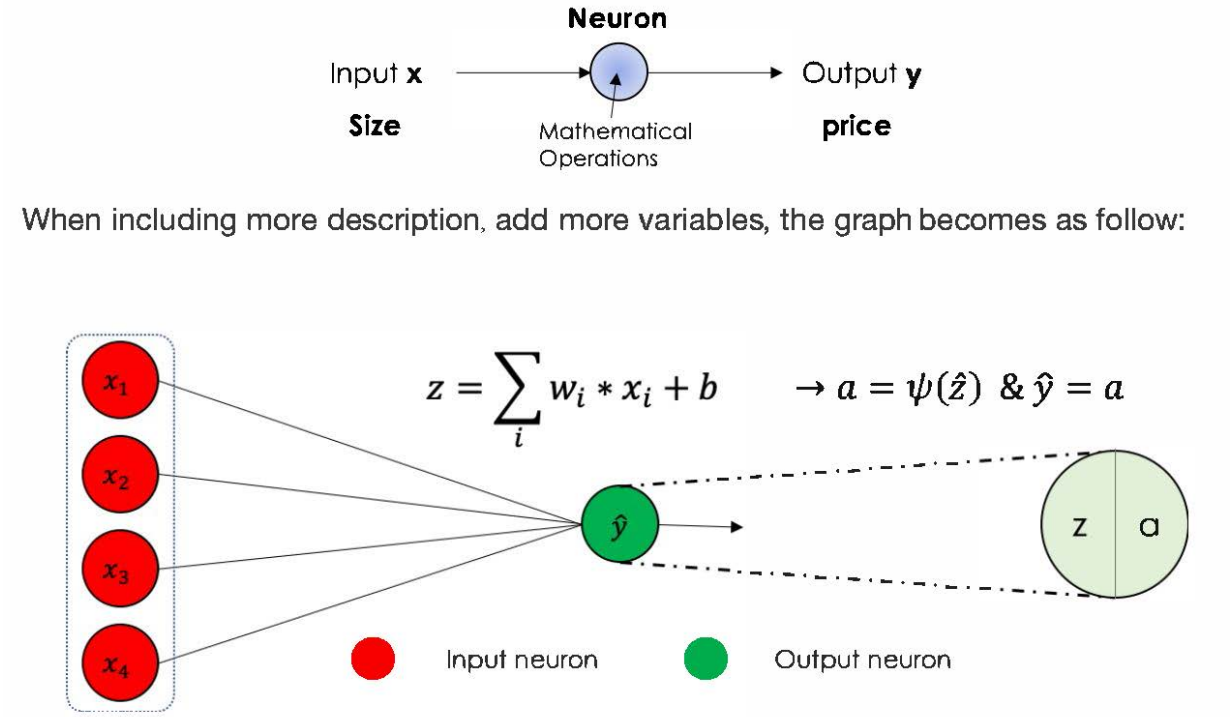


Figure 3.3: Human neuron and mathematical interpretation. *Source:*(Img)

3.3.1 Convolution Layers

Before successful application of deep learning network, computer vision for recognition was dependant on two separable and complementary steps: The input data was transformed through a set of hand operations to a required form. The transformed information is an abstract representation of the input data. The input data is changed in such a way can be separable by a classifier. The transformed data is finally used to train a classifier algorithms to recognize the information of the input signal (49, 51). The used transformation affects the performance of any classifier.

Convolution layer frames can be defined as computational models that allow the computer to extract useful information from the input data by representing multiple levels of abstraction. Unique or important features of the input are amplified at higher layers in the network and become more robust to insignificant variations. The multilayers stack several blocks of modules with alternating linear and nonlinear functions.

A CNN is made up of an input layer **x**, an output layer **y** and a stack of multiple hidden layers **h** where each layer consists of several units. The hidden layer/unit, h_j, receives input from the units of the previous layer and is defined as a weighted combination of the inputs and follows a nonlinear form:

$$h_j = F(b_j + \sum_i w_{i,j} x_i) \tag{3.1}$$

where w_{ij} are the weights controlling the strength of the connection between the input layers units and the hidden unit layers, b_j is the bias of the hidden layer and is added to the weighted sum and passed through the non-linear activation function F(.) to yield the output h_j. The bias b_j allows for the shifting of the activation function left and right.

3.3.2 Image Analysis

A convolutional neural network works in a similarly way by comparing the pixel values in an image. The features in an image are differentiated by the intensity of information it carries. The activation functions activate when it torch the edges, color, shape, or distinct areas of the image. We apply a defined filter to identify this information/abnormalities in an image. The most common filter is the sigmoid which reads and activates when it identifies a curve in an image. We focused on ReLu function in this research.

Figure 3.4 illustrates the image of a dog and bird, with a pixel array of numbers on top of the images. The pixels' values are higher around the surface edges of the objects. This is where we have distinct information. CNN can identify this information

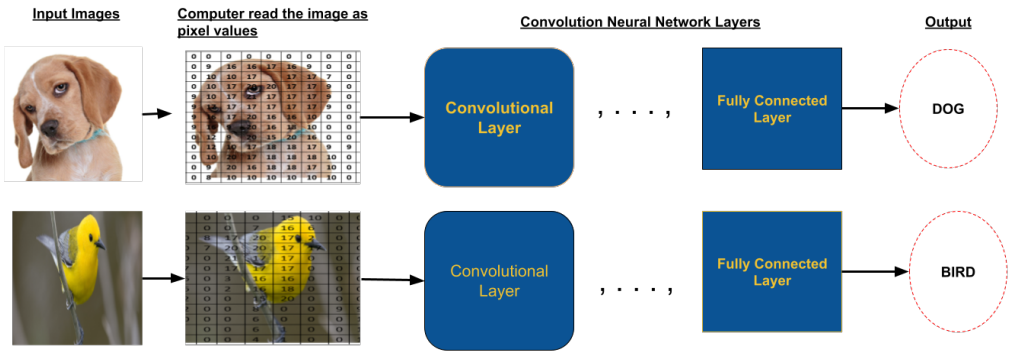


Figure 3.4: Convolution; Image and Filter

by comparing the intensity of the neighboring pixel.

In our corresponding marked images, see 3.2, we analyse the marked white patch areas. The white patched areas contain distinct abnormalities which the model will treat differently by amplifying its' output values.

3.3.3 Filters/Kernel

A **filter** or kernel is convolved on the input image and computes different feature maps through a convolution mathematical operations. The Element-wise nonlinear activation function is applied to the image. Each neuron in a feature map is linked to a receptive region in the previous layer. The filter is convolved and moved across all spatial points location in an input image dataset. Several **filters** can be used to obtain a full feature map.

Simple mathematical representation:

$$y = X * f$$

where the symbol * denotes a convolution operator.

where y = convolved results, X = input image, f = **filter** or kernel.

To illustrate convolution consider an image with resolution size 3 X 3 and a filter of

size 2 X 2.

Figure 3.5: Convolution; Image and Filter.

Figure: 3.5 The **filter** convolves through the patches of the input images at some location, performs an element-wise multiplication (cross-Correlation) between the values in the filter and their corresponding values in the image, and the element-wise products are summed up as demonstrated below. The sum of element-wise products is the output values for the destination pixel in the output image. This process is repeated for all locations. The **Filter** is moved across the surface of the image to the right, down, and so on. The surface area of the image with information such as edges, colors, patterns, and shapes has higher output values (85).
Figure 3.5, X is an input image with a height of 3 and width of 3 while the height and width of the filter f, is 2. This filter gives the **shape** of kernel window.

The output size is given by the input size $n_h \times n_w$ minus the size of the convolution kernel $k_h \times k_w$.

Mathematically:

$$(n_h - k_h + 1) \times (n_w - k_w + 1) \tag{3.2}$$

where n_h and n_w is the n height and width respectively while k_h and k_w are the kernel height and width. This is pairwise computation (42).

The **Output** is a 2×2 dimension as shown below:

$(17\times1 + 17 \times 1 + 16 \times 0 + 10 \times 1) = 44$

$(17 \times 1 + 9 \times 1 + 10 \times 0 + 0 \times 1) = 26$

$(16 \times 1 + 10 \times 1 + 18 \times 0 + 10 \times 1) = 36$

$(10 \times 1 + 0 \times 1 + 10 \times 0 + 0 \times 1) = 10$

3.4 Stride and Padding

The stride method is used to reduce the dimension of a huge input image while padding helps to keep interesting information along the boundaries of the input image since a convolution kernel with height and width greater than 1 will result in a significantly smaller output image which might result in the loss of this information along the boundaries of the input image. We solve the loss of the information along the original boundaries by applying the padding technique and reduce the dimension of a massive image using the stride technique.

3.4.1 Padding

The padding method (57, 77) helps to keep the information along the boundaries of the original input image. We tend to lose a pixel on the perimeter of the input image when we apply the convolution kernel. A single kernel might result in loss of pixel values along the boundaries of the image, more kernel results in more loss of the pixels values. To remedy this we add extra pixels fillers along the boundaries of the input image and set those values to zeros. This increases the size of the input image (34). We add zeros along the boundary of the images to protect loss of information that might be at the boundary of the original image. When the original image with dimension 4×4 is padded, it increases in size to 6×6 (21, 44). From eq 3.2 we add a total of p_h rows of padding half on top and half on the bottom and a total of p_w columns of padding half on the left and half on the right), the output shape will be of the size.

$$(n_h - k_h + 1 + p_h) \times (n_w - k_w + 1 + p_w) \tag{3.3}$$

We choose the odd kernel size of 1, 3, 5, or 7 to preserve the spatial dimension of the input image when padding. Padding is used to give the output image the same height and width as the input image.

3.4.2 Stride

Stride is a method (88) used to reduce the dimension/resolution of a huge input image. Stride is the number of rows and columns traversed per slide/movement. When computing element-wise multiplication (27) we convolve the kernel at the top-left corner of the input image and slide all over the image locations both downwards and to the right. We might move the kernel one element at a time or by skipping several elements in the image locations. We slide the kernel window more than one element skipping the intermediate location for computation efficiency reasons or when downsampling.

When computing the cross-correlation (42), we start with the convolution window at the top-left corner of the input tensor and then slide its over all locations both down and to the right (90).
Mathematically when stride for height is s_h and the stride for the width is s_w , the output shape is given by

$$[(n_h - k_h + p_h + s_h)/s_h] \times [(n_w - k_w + p_w + s_w)/s_w] \tag{3.4}$$

3.5 Activation Function

The image dataset is non-linear and to capture complex information we introduce the *activation function* to the CNN architecture. The activation function captures desirable nonlinear features in an image. Linear activation function produces linear decision boundary which will cut some useful information in an image features.

Figure 3.6 below demonstrates that we cannot separate the red colors from the green colors using a linear decision boundary(right figure).

Mathematically:

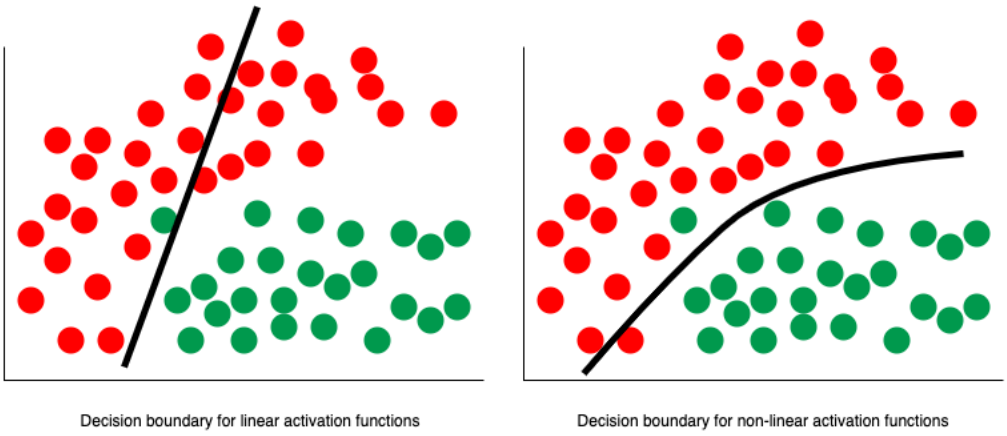


Figure 3.6: Decision boundary for linear and non-linear activation function. *Source:* (CNN)

$$a^l_{i,j,k} = a(y^l_{i,j,k})$$

where

$a^l_{i,j,k}$ is the activation value of convolutional feature $y^l_{i,j,k}$.

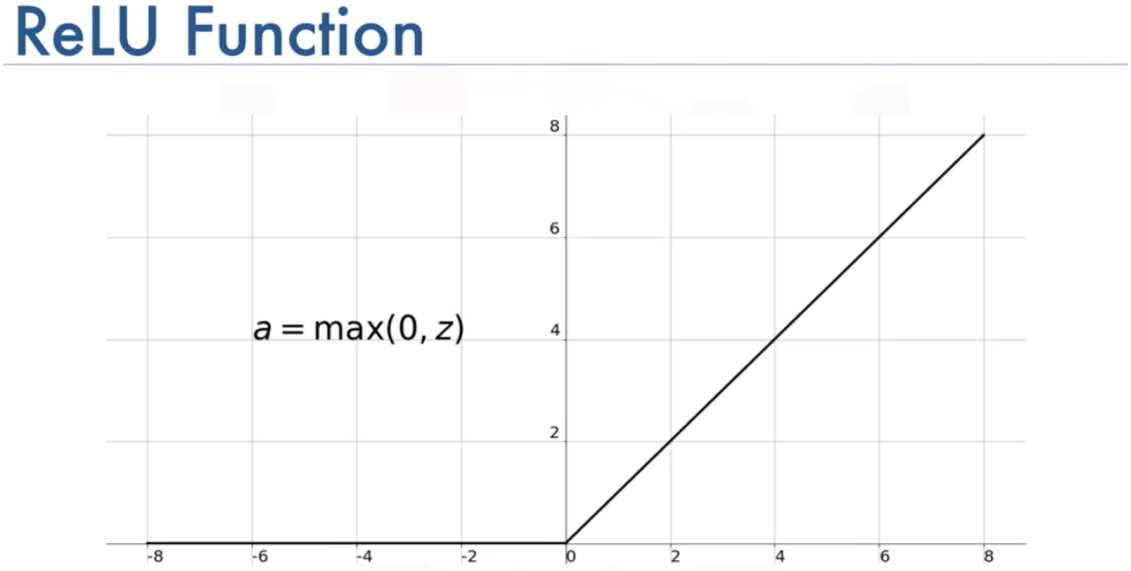
There is a list of activation function and when to use will depend on the problem we are solving. We will use ReLU (8) activation function for the image classification problem because of its consistent gains in improving classification accuracy across deeper models (67).

3.5.1 ReLU

$$f(z) = max(0, z)$$

ReLU (8) is easy to compute and has a non-linearity at $z = 0$

Figure 3.7: Relu Activation Function. *Source:*(Rel)



3.6 Pooling

The pooling layers reduces the resolution size of the input and is placed between two convolutional layer to decrease the number of connections between these layers. It cuts off unwanted parts of the initial convolution layers and outputs only useful features. According to Tobler's First Law of Geography (15), "everything is related to everything else, but near things are more related than distant things."(Tobler, 1970), using the same reasoning on the spatial features of an input image we can confirm that pixel values of input images that are close together are more likely to be alike than those further apart hence initial output of convolution layer will produce similar values for this near/close pixels (52). This is redundant information and we are not learning anything new. To solve this, pooling achieves shift-invariance (33) by pooling the pixel values of the input together. A specific feature map is linked to its counterpart feature map in the following convolutional layer. Pooling operators have no parameters, they are deterministic (10, 86). Typically we calculate the maximum, minimum, or average pixel value in the pooling window (75). The pooling window is focused from the top left of the input image and slide to the right and down, it output a max, min, or average value on the location of the image it hits. A pooling window can be of shape $p \times q$.

Mathematically:

Denote the pooling function as $pool(.)$ for each feature map $a^l_{m,n,k}$ we obtain:

$$y^l_{i,j,k} = pool(a^l_{m,n,k}), \forall (m,n) \in \mathcal{R}_{i,j}$$

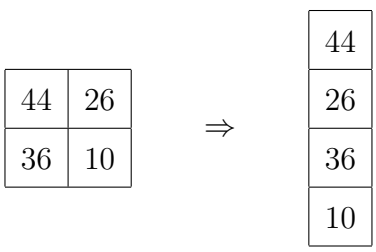
where $\mathcal{R}_{i,j}$ is the neighborhood around location (i, j). The known pooling operations are the **max**, **min**, **sum**, and **average** pooling (22). Under max-pooling, we choose the pool size and apply it to the region of the image as we extract the maximum pixel value in each region and place the max value in the output image of the corresponding pixel. The output width and height of the initial convolution layer are divided by the pooling size.

We can stack several convolutional and pooling layers to extract higher-level features of the object datasets. These convolution and pooling outputs lead to one or more fully connected layers that extract high-level feature representations.

3.7 Fully Connected Layers

The output from the convolution and pooling layers (16, 19); extracted information from the data are used as input in this layer which in turn generates final results (81). A fully connected layer works with **1-dimensional data**; each input image in its row. The output values from the previous convolutional and pooling layers are converted into a one-dimensional data format. Each neuron of the previous layers is connected to every single neuron in the current layer and generates a web of interconnected semantic information (82).

The figure below shows conversions of a matrix to a vector or 1-dimensional data.



Individual values are separate features that represent an image. A fully connected layer performs two mathematical operations; linear and non-linear transformation (91).

The commonly used **classification methods** are **softmax** and **SVM** classifiers (64) which are supervised method respectively. Softmax achieves a better classification performance compared to SVM. It output probability distribution which can be used to classify input data. These methods are combined with CNN to solve classification problems (7).

The linear transformation of the data is of the form:

$$y^l_{i,j,k} = W^T_k . X^l_{i,j} + b^l_k \tag{3.5}$$

where (i, j) is the location of feature value
$k - th$ is feature map
and $l - th$ is the layer.
and
W_k = weights vector,

$X^{i,j}$ = input image centered at location (i, j) of the $l - th$ layer,
b_k = constant bias term of $k - th$ filter of $l_t h$ layer.

W_k weight is a randomly initialized matrix to generate $y_{i,j,k}$ and is shared. The sharing of weight reduces the complexity of the model and makes training of the network easy.
For illustration consider m features from an input image and n neurons. The size of the weights will be (i, j). When $i = 4$ and $j = 2$ the linear transformation will be of the form:

Denote:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \qquad W = \begin{bmatrix} W_{1,1} & W_{1,2} \\ W_{1,2} & W_{2,2} \\ W_{1,3} & W_{3,2} \\ W_{1,4} & W_{4,2} \end{bmatrix} \qquad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Input Weight matrix Bias

This is how it appears in eq 3.5 i.e $y^l_{i,j,k} = W^T_k . X^l_{i,j} + b^l_k$

$$y = \begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} & W_{1,4} \\ W_{1,2} & W_{2,2} & W_{3,2} & W_{4,2} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Consider a vector of weights and bias terms θ, to find an optimum parameter for classification tasks we minimize the defined loss function.

3.8 Training CNN

Our primary purpose is to construct a deep CNN model that is going to classify an image into COVID-19 positive and COVID-19 negative or other disease categories with a certain probability by finetuning the pre-trained ResNet18 CNN model.

The transfer learning version of deep CNN is viable for this task and will extract fixed image features and predict the class of the images: COVID-19 positive or COVID-19 negative. We pre-trained the ConvNet model using a set of 3100 training datasets, finetune the model to obtain a very small loss/error in the network. We evaluate model performance by computing the distance scores between the predicted values of the convolution network and the actual target output values and compare these error scores. The smaller the distance scores(loss) the better the deep CNN model.

3.8.1 Loss Function

We use the cross-entropy loss function (40) and is mathematical expressions is as follows:

$$L = \frac{1}{N} \sum_{i=1}^{N} l(\theta; y^n, o^i). \tag{3.6}$$

We minimize the total loss over the whole datasets in eq 3.6 to find the best fitting set of parameters. One of the commonly used methods to optimize the CNN network is the stochastic gradient descent (33).

The predicted loss function $L(f(x^i; W))$ is compared with the actual values of y^i.

3.8.1.1 Loss Optimization

The loss function is optimized by updating the weights in the network until we find the weight that results in a minimum loss over the trained datasets (50). This is called loss optimization and is expressed mathematically as follows:

$$W^* = \arg\min_{W} L(f(x^i; W), y^i). \tag{3.7}$$

where W is a vector of weights and W^* are the updated new weights that minimize the loss in eq 3.7. We compute the *gradient* $= \frac{\partial L}{\partial W}$ with respect to the weights at a picked point and iterate the process until we achieve the lowest minimum. Each iteration returns a new weight.

Implementation:

1. Initialize the weight randomly and draw them from normal distribution $W \sim \mathcal{N}(\mu, \sigma^2)$.
2. Iterate until it converges
3. Compute gradient $\frac{\partial L}{\partial W}$
4. Update the weights $W^* = W - n * \frac{\partial L}{\partial W}$
5. Return the final weights

3.8.2 Softmax Layers

Softmax (40) turns the output pixel values into probabilities. It predicts the output of the deep CNN model and is used in multiclass classification problems. The outputs of the Softmax transform are always in the range [0, 1] and sums up to 1. Hence, they form a probability distribution.
It is mathematically expressed as:

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}.$$

Where $s(x_i)$ is the probability for the given numbers $i = 1, 2, \ldots, n$, e = Euler's number, e (mathematical constant), x_i is the given numbers for $i = 1, 2, \ldots, n$ and $\sum_{j=1}^{n} e^{x_j}$ the summation of all j. The bigger the value of x_i, the larger its probability.
Softmax helps us quantify and predict the probability of an object belonging to a particular class. This is useful in training and evaluating the best performing CNN algorithm.

We compute **cross-entropy loss** (40) which accounts for the certainty of each prediction. The mathematical expression of cross-entropy loss:

$$L = -\ln(p_C).$$

where L = binary cross-entropy loss, c is the correct image class or output digit, p_c is the predicted probability for class c and ln is the natural log. A lower L, $(loss)$ is better than a larger loss.

3.9 Proposed CNN: ResNet18

There has been a significant improvement on deep convolutional neural networks (66) since the notable success of the previous research works; AlexNet 2012 (5). We will transfer learning from the pre-trained model and state-of-the-art residual learning ResNet (37, 87) which won the ILSVRC 2015 classification competition (78) with training on million of ImageNet datasets to our COVID-19 image datasets. The purpose is to apply

the learned knowledge from the pre-trained ResNet model to the X-ray images to solve the classification problem which would have been time-consuming in constructing the deep ResNet model from the scratch due to insufficient certified COVID-19 samples. ResNet-18 is a deep CNN with 18 layers presented in its' framework. It can classify images into 1000 object groups and the size of the input image is 224x224. It was implemented in 2015 (36) and replaced VGG-16 layers using 101 layers (35). ResNet has demonstrated that it performs efficiently with more layers. (70, 74).

Residual learning is expressed mathematically as follows: consider $H(x)$ to be a mapping that is fitted by some stacked layers and x input images. Following the assumption that multiple non-linear layers asymptotically approximate complex function, we can equally assume that they can asymptotically approximate residual function (36). This is expressed as $H(x) - x$. The stack layers can approximate $H(x)$ but we express these layers to approximate the residual function $F(x) := H(x) - x$.

The advantage of RestNet-18 is it addresses the vanishing gradient problem using identity mapping and can be trained efficiently without increasing the percentage of the training error (12, 36). Residual networks provides residual connections straight to earlier layers unlike other non-residual neural model (79).

In this paper, we will use the residual network with 18 layers i.e ResNet18 (87). We will combine the existing parameters of the pre-trained ResNet18 and finetune the models to use updated parameters through freezing the parameters of the pre-trained model. Table 3.1 below showcases the ResNet18 framework.

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

Table 3.1: ResNet18 Framework. *Source:*(Res)

Table 3.1 displays a convolution layer with kernel size and strike of 7×7 and 2 respectively. The input from the first layer is added to the output of the two-second convolution layers which has been obtained by the 3×3 maximum pool layer with kernel size 3×3, 64. This forms the first block of residual. The results from this block are added to the two convolution layer with kernel size of 3×3 and 128. This is the third part of the residual. The outputs from the third residual block are added through skip connection with the output of the two convolution layers, kernel size 3×3, 256 to form the fourth residual block. This block is again added through skip connection with the output of the two convolution layers 3×3, 512 to form the fifth residual block. The average pooling is applied to the fifth residual block and fed into the fully connected layers. This is 1-dimension output which is finally fed into softmax layers.

The ResNet18 algorithm will be implemented using Python and R software. Optimization algorithm is adopted from **torch.optim** package and only parameters of final layer are being optimized. The implementation follows the success in (45, 46, 72). The image is randomly sampled from a 224 × 224 cropped image. We choose the learning rate (0.01%), weight decay (10%), momentum (90%), cut-off threshold (0.1), batch

normalization (BN) (41) and fully connected layers (72).

The learning rates are significant parameters in fine-tuning the deep CNN model. Large learning rates will result in a model divergence or unstable training with small rates will lead to slow convergence of the model. After fine-tuning and retraining the model we used a learning rate of 0.01% which was the optimum learning rate. We started with a tiny learning rate and increase it after each training until a big loss or loss expansion was observed. The learning rate was chosen one step below the learned rate at the point the loss was minimal. For example, we choose a learning rate of 0.01% when the loss was low at 0.1% and so on. The learning rates experimented in this study was not discussed. A momentum of 90% was selected to accelerate the training and learning rates time and to helps with convergence. The decay Gamma parameter in the algorithm specifies the learning rate and we evaluated 5 decay rates (10%, 20%, 25%, 35%, and 40%) and their influence on the performance of the model. A decay learning rate of 10% gave us the best performing model so we selected it. 20 images was set to be trained with 100 epochs. These helps with updating the weights in the network. The presence of noise in the model was addressed by applying Adam Optimizer method (68) which leverages the features of AdaGrad and RMSProp (20). **Overfitting** was addressed in the algorithm by horizontally flipping the input images. This is a data augmentation technique (62) and was used to increase small datasets to a relatively large dataset since large datasets are a remedy for overfitting while underfitting (11) was addressed by the increased layers or capacity of the model.

Chapter 4

Results

4.1 Chest X-Rays Exploration

The X-ray datasets comprised of two groups: The COVID-19 positive and the COVID-19 negatives. We used 60% data for training and 40% for testing. The test set acted as a validation set as well.

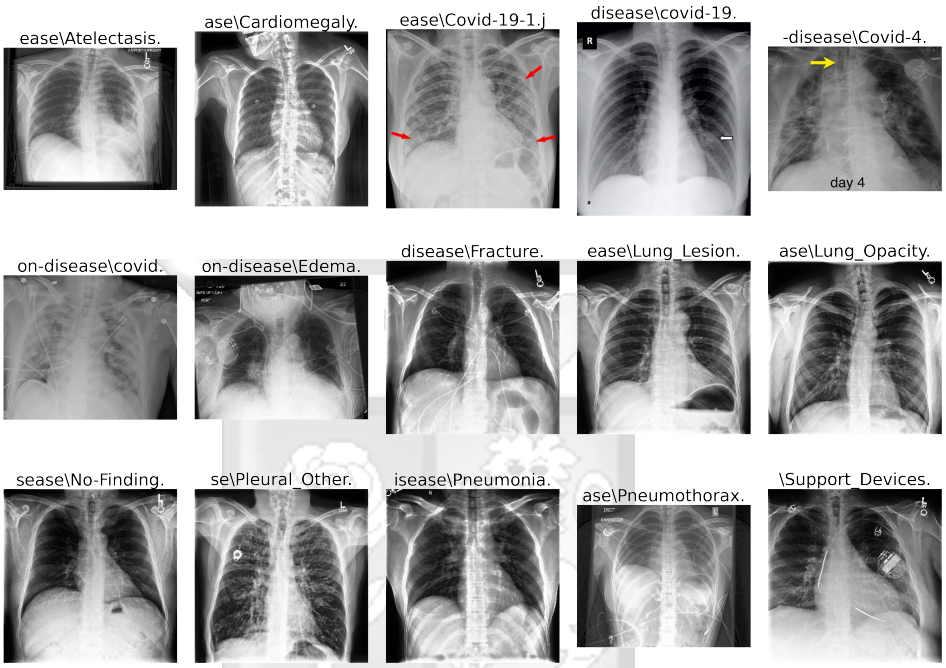
Figure 4.1 displays sample labels of patient's X-rays images infected with COVID-19 virus and other common chest diseases. The images are of high and low resolution and were collected from different sources (26). The noise was added by doubling and flipping the images and other x-rays images added to the universe (62). The model needs to achieve significant precision on this dataset. In a real clinical setup, it is improbable to take X-rays images under a highly controlled environment that will result in quality, clean, and high-resolution images. It is impossible in a clinic/hospital with limited or poor image scanning equipment in the developing world. All images are normalized to produce a standard distribution so that the model doesn't perform strangely. We down-sample all images to a resolution of 224 $\times$ 224 before feeding into the network.

4.2 Model Analysis and Performance Metrics

4.2.1 Model Analysis

We pre-trained ConvNet ResNet18 as a **fixed feature extractor** where we freeze the parameters so that the gradients are not computed backward. Here the ResNet18 model is loaded and pre-trained set to true. The test loss (**12.85%**) and accuracy (**95.97%**) respectively. The best test accuracy (**96.00%**) and training time was 472m 26s. When fine-tuning the model the new parameters was set as a requirements of the algorithm

Figure 4.1: COVID-19 Positive and COVID 19 Negative. Image source (26, 55)



to true otherwise false. The **parameters** used in this model were adopted from the state-of-the-art algorithm of ResNet18 (36, 45). Table 4.1 displays the parameters and their optimal values.

Table 4.1: Training Model Parameters

ResNet18 Pre-trained	
Parameter	**Value**
Learning rate	0.01 %
Momentum	90 %
Epochs	100
cut off threshold	10 %
Batch size	20
Criterion	nn.CrossEntropyLoss
Optimizer	Adam
Gamma	0.1

When finetuning the ResNet18 model we *did not* freeze the parameters as we allowed the model to **learn and optimize** new parameters. The initial parameters used in the pre-trained model was used here as starting values and updated as the model finetune, train and learns from the data.

The **finetuned training model** achieved test loss (**6.96%**) and accuracy (**97.31%**) respectively. The best test accuracy (**97.4%**) at epoch 13. The training time was 1016m 35s.

4.2.2 Performance Evaluation

We will evaluate the performance of our trained model of deep ResNet18 learning model using the 40% test dataset (9). The test datasets are are input to the trained model to validate the classification model. The performance metrics are *Sensitivity/recall and specificity at a selected threshold, AUC and ROC curve, precision, F1-score, accuracy, and Cohens kappa* (14).

Table 4.2: Model Evaluation Metrics

ResNet18	
Metrics	**Scores**
Recall	75 %
AUC	98.82 %
Precision	87.5 %
Accuracy	98.56 %
F1-score	80.77 %
Cohens kappa	80.03 %

Table 4.3: Sensitivity and Specificity (28, 54) for ResNet18 models under different cut off threshold

ResNet18		
Threshold	Sensitivity	Specificity
0.1	0.75	0.995
0.2	0.595	0.999
0.3	0.464	1

$$Sensitivity = \frac{\text{Correct prediction of COVID-19 positive}}{\text{Total number of COVID-19 positive dataset}} \tag{4.1}$$

$$Specificity = \frac{\text{Correct prediction of COVID-19 negative}}{\text{Total number of COVID-19 negative dataset}} \tag{4.2}$$

$$Precision = \frac{\text{True COVID-19 Positive}}{\text{True COVID-19 Positive + False COVID-19 positive}} \tag{4.3}$$

$$F1 - score(61) = 2 \times \frac{\text{Precision x Recall}}{\text{Precision + Recall}} \tag{4.4}$$

4.2.3 Confusion Matrix

The confusion matrix (38) shows the number of correctly classified images is 75% and 59.5% at cut-off points 0.1 and 0.2 respectively. The classification sensitivity improves with an increase in the number of quality images.

Figure 4.2: Confusion Matrix at thresh = 0.1 & 0.2 for test datasets

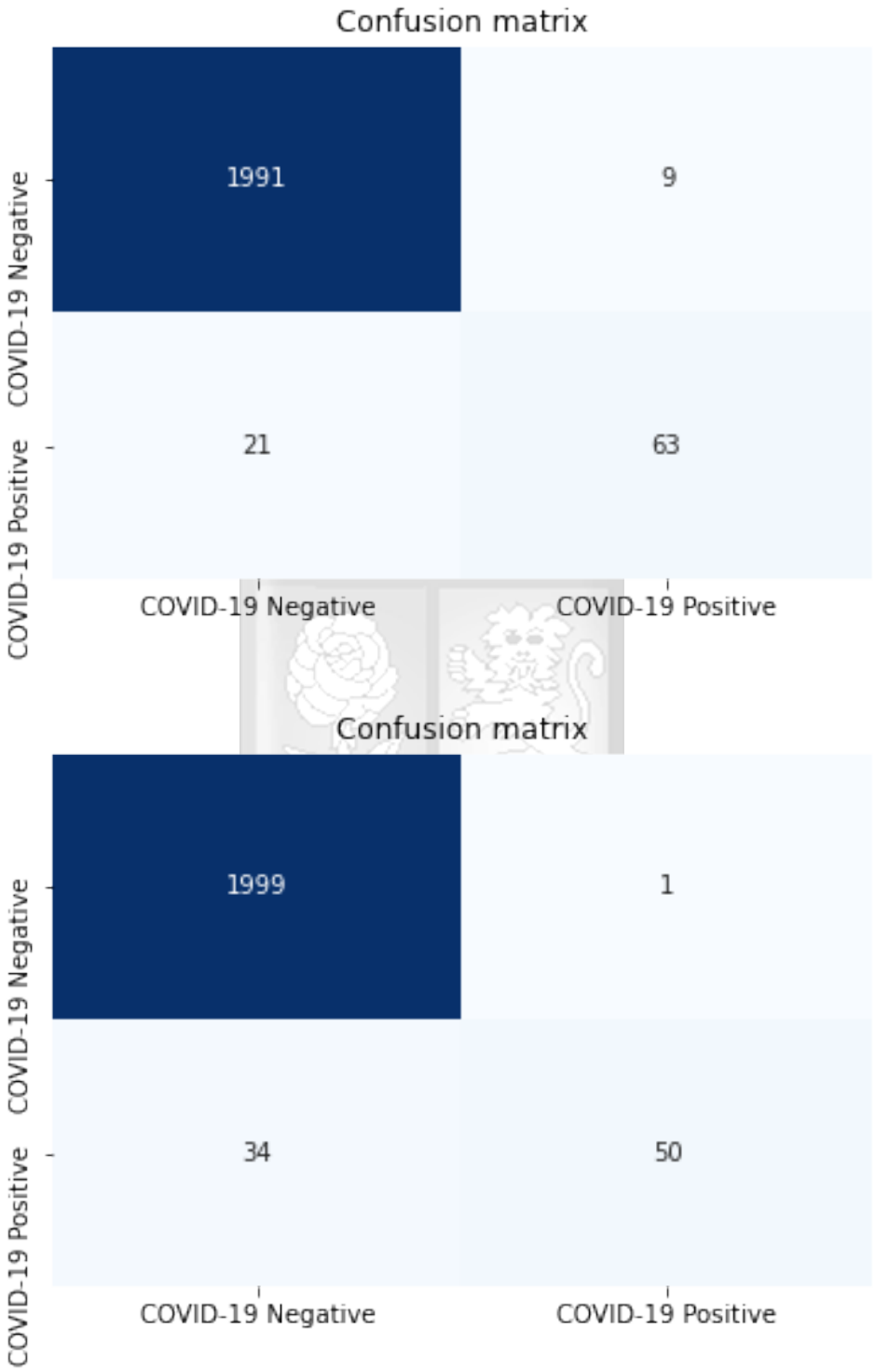
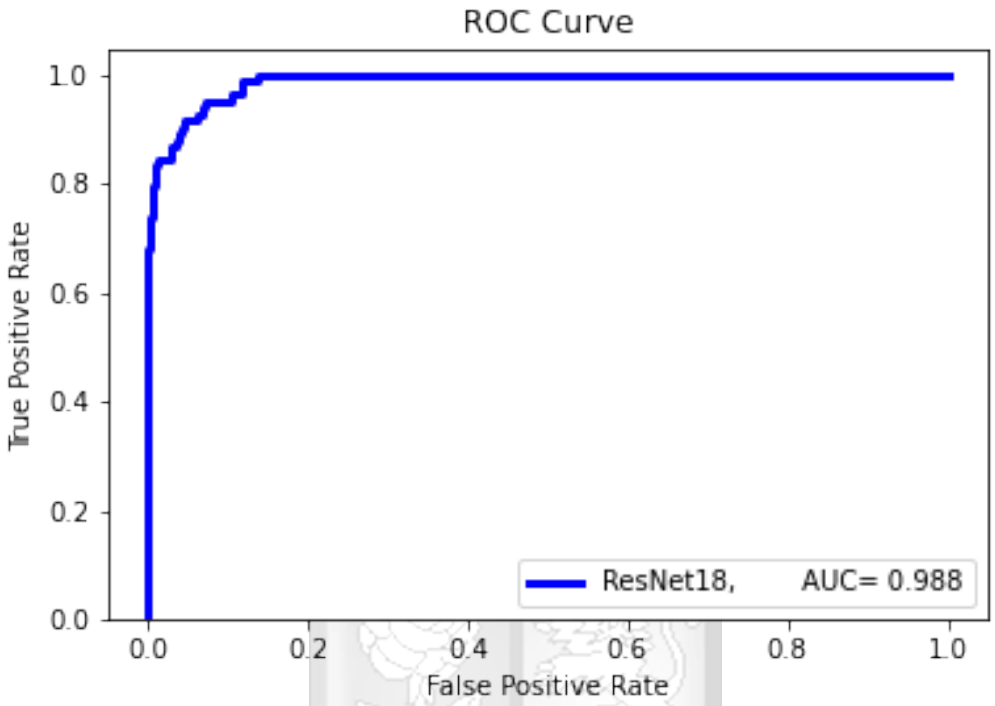


Figure 4.2 top illustrates confusion matrix at threshold (0.1) and bottom (0.2).

4.2.4 Receiver Operating Characteristics, Area Under the Curve

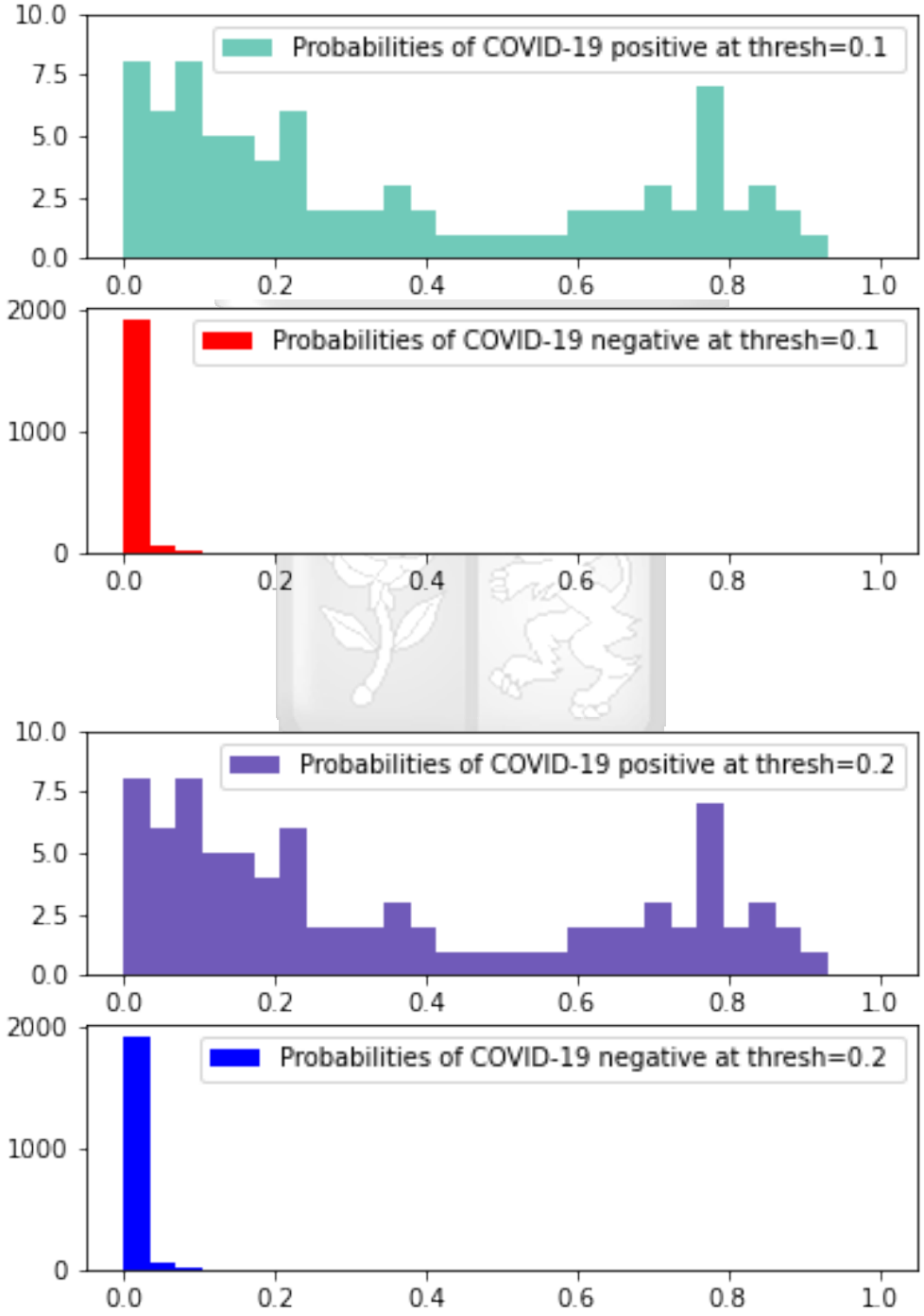
Figure 4.3: Receiver Operating Characteristics Under Test Datasets



4.2.5 Confidence Interval

The graph fig 4.4 is a visual representation of the distance score using the histogram. A score below the thesholds is COVID-19 negative while scores above the threshold are COVID-19 positive.

Figure 4.4: Distance scores Histogram under thresh=0.1 & 0.2 for test datasets



Chapter 5

Discussion and Conclusion

5.1 Discussion

Confusion matrix in fig 4.2 shows significant a performance in detecting COVID-19 with performance of over **75%**. This is interpreted 75% of all positive COVID-19 datasets is identified and classified as positive COVID-19 images by the model. On the other hand, 99.5% of negative COVID-19 was classified as negative COVID-19.

A close look at the cut-off threshold of 0.1 and 0.2 shows that the model performs well at a lower cut-off point. A significant number of COVID-19 positive images is classified with a probability greater than 0.1. The best threshold for these datasets is at 0.1 but as we increase the datasets the cut-off threshold can be increased and still achieve a higher classification performance. The first graph fig 4.4 demonstrates over 75% of positive COVID-19 images was detected with probability greater than the threshold of 0.1. On the second graph the negative COVID-19 images were all below the threshold of 0.1. This is an indication that the model performed well in detecting both the COVID-19 positive and COVID-19 negative datasets.

The ROC curve in fig 4.3 demonstrates that COVID-19 cases can be identified by the model ResNet18 with an accuracy of (AUC = 98.88%). Using this model to accurately detect COVID-19 cases can be reliable. A close inspection of the loss function (test loss = 12.85%) when the model is used as a feature extracted but when finetuning that is allowing the model to learn from the data and updates its parameters the test loss is is cut by half (test loss = 6.96%) and test accuracy improved by 97.4%. In a super-controlled environment, we expect the (test loss = 0%) which is not the case in this study.

A close look at other metrics, eq 4.3 (precision = 87.5%), eq 4.1 (recall = 75 %), eq 4.4 (F1-score = 80.77%), and eq 4.2 (specificity = 99.5%), all illustrate that the model

was able to detect and classify the COVID-19 positive cases. Precision(87.5%) was used instead of accuracy because it is the most reliable metric.

5.2 Conclusion

Deep ResNet18 convolutional neural network and transfer learning (65) was used to identify and classify COVID-19 positive images from COVID-19 negative. The COVID-19 positive images were examined and certified as COVID infected while the COVID-19 negatives samples are any datasets collected and stored in public databases before the breakout of the COVID virus in 2019. These comprise of the classes (Pneumonia, normal, no finding, Edema, Pleural Effusion, and normal chest x-ray images). The initial training to extract features was done using the pre-trained model version of ResNet18. The second training was finetuning the model to learn on its own and produce its new parameters from the fed datasets. The finetuning improved the model. The model analysis shows that the finetune ResNet18 model using transfer learning performed better than the traditional state-of-the-art pre-trained ResNet18 model constructed from ImageNet datasets. We added several noises by flipping images, adding the non-COVID-19 images from different sources, manipulating the images to double the size of the training image set. The pre-trained model results achieved (test accuracy = **95.97%**) and (test loss = **12.85%**) respectively. The finetuned model on the test dataset at threshold (0.1) achieved accuracy (test accuracy = **97.31%**) and (test loss = **6.96%**) with (precision = **87.5%**), (F1-score = **80.77%**), (sensitivity = **75%**) and (specificity = **99.5%**). We desire that this paper will help radiologists and medical doctors in the developing world to detect the abnormalities in chest X-ray images and offer first hand information aid in the inspection, detection, and diagnosis of chest diseases. By the time I am concluding this paper we had limited COVID-19 images available. Further extension of this research work will focus on using much larger COVID-19 or non-COVID-19 datasets with more focus on systematic review around data acquisition, data certification, model development and pitfalls, and explanation construction. In addition to demonstrating and expanding on the statistical aspect of the research work (39). The success of the model can be extended to real clinical trials.

References

[Img] https://towardsdatascience.com/deep-learnings-mathematics-f52b3c4d2576. Accessed: 2021-05-11.

[CNN] https://makshay.com/neural-network-basics-the-perceptron. Accessed: 2021-05-10.

[Rel] https://medium.com/@toprak.mhmt/activation-functions-for-deep-learning-13d8b9b Accessed: 2021-05-10.

[Res] https://www.researchgate.net/figure/ResNet-18-Architecture_tbl1_322476121. Accessed: 2021-05-21.

[5] (December 2012). *Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems*. NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1.

[6] A. Cruzroa, J. C. C. and Gonzalez, F. A. (2011). Visual pattern mining in histology image collections using bag of features,. *Artificial Intelligence in Medicine*, 52(2):91–106.

[7] Agarap, A. F. (2017). An architecture combining convolutional neural network (cnn) and support vector machine (svm) for image classification. *arXiv preprint arXiv:1712.03541*.

[8] Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

[9] Ahuja, S., Panigrahi, B. K., Dey, N., Rajinikanth, V., and Gandhi, T. K. (2021). Deep transfer learning-based automated detection of covid-19 from lung ct scan slices. *Applied Intelligence*, 51(1):571–585.

[10] Akhtar, N. and Ragavendran, U. (2020). Interpretation of intelligence in cnn-pooling processes: a methodological survey. *Neural Computing and Applications*, 32(3):879–898.

[11] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., and Farhan, L. (2021). Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8(1):1–74.

[12] Ardakani, A. A., Kanafi, A. R., Acharya, U. R., Khadem, N., and Mohammadi, A. (2020). Application of deep learning technique to manage covid-19 in routine clinical practice using ct images: Results of 10 convolutional neural networks. *Computers in Biology and Medicine*, 121:103795.

[13] Awais, M., Müller, H., Tang, T. B., and Meriaudeau, F. (2017). Classification of sd-oct images using a deep learning approach. In *2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 489–492. IEEE.

[14] Ayyachamy, S., Alex, V., Khened, M., and Krishnamurthi, G. (2019). Medical image retrieval using resnet-18. In *Medical Imaging 2019: Imaging Informatics for Healthcare, Research, and Applications*, volume 10954, page 1095410. International Society for Optics and Photonics.

[15] Bacon, F. and Tobler, W. (2010). Data, information and knowledge.

[16] Bai, C., Huang, L., Pan, X., Zheng, J., and Chen, S. (2018). Optimization of deep convolutional neural network for large scale image retrieval. *Neurocomputing*, 303:60–67.

[17] Balogh, E. P., Miller, B. T., and Ball, J. R. (2015). Improving diagnosis in health care.

[18] Barata, C., Ruela, M., Francisco, M., Mendonça, T., and Marques, J. S. (2013). Two systems for the detection of melanomas in dermoscopy images using texture and color features. *IEEE systems Journal*, 8(3):965–979.

[19] Basha, S. S., Dubey, S. R., Pulabaigari, V., and Mukherjee, S. (2020). Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*, 378:112–119.

[20] Bera, S. and Shrivastava, V. K. (2020). Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification. *International Journal of Remote Sensing*, 41(7):2664–2683.

[21] Boddeti, V. N. (2012). *Advances in correlation filters: vector features, structured prediction and shape alignment*. PhD thesis, Carnegie Mellon University.

[22] Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118.

[23] Carayon, P., Hundt, A. S., Karsh, B., Gurses, A. P., Alvarado, C., Smith, M., and Brennan, P. F. (2006). Work system design for patient safety: the seips model. *BMJ Quality & Safety*, 15(suppl 1):i50–i58.

[24] Celebi, M. E., Kingravi, H. A., Uddin, B., Iyatomi, H., Aslandogan, Y. A., Stoecker, W. V., and Moss, R. H. (2007). A methodological approach to the classification of dermoscopy images. *Computerized Medical imaging and graphics*, 31(6):362–373.

[25] Chan, T.-H., Jia, K., Gao, S., Lu, J., Zeng, Z., and Ma, Y. (2015). Pcanet: A simple deep learning baseline for image classification? *IEEE transactions on image processing*, 24(12):5017–5032.

[26] Cohen, J. P., Morrison, P., Dao, L., Roth, K., Duong, T. Q., and Ghassemi, M. (2020). Covid-19 image data collection: Prospective predictions are the future. *arXiv preprint arXiv:2006.11988*.

[27] Cong, J. and Xiao, B. (2014). Minimizing computation in convolutional neural networks. In *International conference on artificial neural networks*, pages 281–290. Springer.

[28] Dixon, W. J. and Mood, A. M. (1948). A method for obtaining and analyzing sensitivity data. *Journal of the American Statistical Association*, 43(241):109–126.

[29] G. E. Hinton, S. O. and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets,. *Neural Computation*, 18(7):1527–1554.

[30] Gaynor, M., Wyner, G., and Gupta, A. (2014). Dr. watson? balancing automation and human expertise in healthcare delivery. In *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, pages 561–569. Springer.

[31] Giri, C. (2020). 3d convolution neural networks for medical imaging; classification and segmentation: A doctor's third eye. Master's thesis, University of Agder.

[32] Glasser, O. (1993). *Wilhelm Conrad Röntgen and the early history of the Roentgen rays*. Number 1. Norman Publishing.

[33] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., and Wang, G. (2015). Recent advances in convolutional neural networks. *CoRR*, abs/1512.07108.

[34] Hashemi, M. (2019). Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation. *Journal of Big Data*, 6(1):1–13.

[35] Hassan, M. (2019). Resnet (34, 50, 101): Residual cnns for image classification tasks. *Neurohive. io*.

[36] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

[37] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[38] Heidari, M., Mirniaharikandehei, S., Khuzani, A. Z., Danala, G., Qiu, Y., and Zheng, B. (2020). Improving the performance of cnn to predict the likelihood of covid-19 using chest x-ray images with preprocessing algorithms. *International journal of medical informatics*, 144:104284.

[39] Hryniewska, W., Bombiński, P., Szatkowski, P., Tomaszewska, P., Przelaskowski, A., and Biecek, P. (2020). Checklist for responsible deep learning modeling of medical images based on covid-19 detection studies. *arXiv preprint arXiv:2012.08333*.

[40] Huang, J.-T., Li, J., Yu, D., Deng, L., and Gong, Y. (2013). Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7304–7308. IEEE.

[41] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

[42] Jeon, J. and Lee, S. (2018). Reconstruction-based pairwise depth dataset for depth image enhancement using cnn. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 422–438.

[43] Jutel, A. (2009). Sociology of diagnosis: a preliminary review. *Sociology of health & illness*, 31(2):278–299.

[44] Kim, J., Lee, J. K., and Lee, K. M. (2016). Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654.

[45] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

[46] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

[47] Lee, H. and Song, J. (2019). Introduction to convolutional neural network using keras; an understanding from a statistician. *Communications for Statistical Applications and Methods*, 26:591–610.

[48] Lee, J.-G., Jun, S., Cho, Y.-W., Lee, H., Kim, G. B., Seo, J. B., and Kim, N. (2017). Deep learning in medical imaging: general overview. *Korean journal of radiology*, 18(4):570.

[49] Liu, J.-e. and An, F.-P. (2020). Image classification algorithm based on deep learning-kernel function. *Scientific programming*, 2020.

[50] Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

[51] Lotte, F., Bougrain, L., Cichocki, A., Clerc, M., Congedo, M., Rakotomamonjy, A., and Yger, F. (2018). A review of classification algorithms for eeg-based brain–computer interfaces: a 10 year update. *Journal of neural engineering*, 15(3):031005.

[52] Lv, Z., Zhang, P., and Atli Benediktsson, J. (2017). Automatic object-oriented, spectral-spatial feature extraction driven by tobler's first law of geography for very high resolution aerial imagery classification. *Remote Sensing*, 9(3).

[53] M. R. Zare, A. Mueen, M. A. and Seng, W. C. (2013). Automatic classification of medical x-ray images: hybrid generative-discriminative approach,. *IET Image Processing*, 7(5):523–532.

[54] Maior, C. B., Santana, J. M., Lins, I. D., and Moura, M. J. (2021). Convolutional neural network model based on radiological images to support covid-19 diagnosis: Evaluating database biases. *Plos one*, 16(3):e0247839.

[55] Minaee, S., Kafieh, R., Sonka, M., Yazdani, S., and Soufi, G. J. (2020). Deep-covid: Predicting covid-19 from chest x-ray images using deep transfer learning. *Medical image analysis*, 65:101794.

[56] Mwadulo, M. W. (2016). A review on feature selection methods for classification tasks.

[57] Nguyen, A.-D., Choi, S., Kim, W., Ahn, S., Kim, J., and Lee, S. (2019). Distribution padding in convolutional neural networks. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4275–4279. IEEE.

[58] Organization(WHO), W. H. (2012). International classification of diseases (icd).

[59] Ozturk, T., Talo, M., Yildirim, E. A., Baloglu, U. B., Yildirim, O., and Acharya, U. R. (2020). Automated detection of covid-19 cases using deep neural networks with x-ray images. *Computers in biology and medicine*, 121:103792.

[60] Pathak, Y., Shukla, P. K., Tiwari, A., Stalin, S., and Singh, S. (2020). Deep transfer learning based classification model for covid-19 disease. *Irbm*.

[61] Pehrson, J. and Lindstrand, S. (2020). Support unit classification through supervised machine learning.

[62] Perez, L. and Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

[63] Pham, D. L., Xu, C., and Prince, J. L. (2000). Current methods in medical image segmentation. *Annual review of biomedical engineering*, 2(1):315–337.

[64] Qi, X., Wang, T., and Liu, J. (2017). Comparison of support vector machine and softmax classifiers in computer vision. In *2017 Second International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pages 151–155. IEEE.

[65] Rahman, T., Chowdhury, M. E., Khandakar, A., Islam, K. R., Islam, K. F., Mahbub, Z. B., Kadir, M. A., and Kashem, S. (2020). Transfer learning with deep convolutional neural network (cnn) for pneumonia detection using chest x-ray. *Applied Sciences*, 10(9):3233.

[66] Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., et al. (2017). Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*.

[67] Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

[68] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[69] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. 115(3):211–252.

[70] Shen, F., Gan, R., and Zeng, G. (2016). Weighted residuals for very deep networks. In *2016 3rd International Conference on Systems and Informatics (ICSAI)*, pages 936–941. IEEE.

[71] Shijie, J., Ping, W., Peiyi, J., and Siping, H. (2017). Research on data augmentation for image classification based on convolution neural networks. In *2017 Chinese automation congress (CAC)*, pages 4165–4170. IEEE.

[72] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[73] Song, M. and Civco, D. (2004). Road extraction using svm and image segmentation. *Photogrammetric Engineering & Remote Sensing*, 70(12):1365–1371.

[74] Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Training very deep networks. *arXiv preprint arXiv:1507.06228*.

[75] Sun, M., Song, Z., Jiang, X., Pan, J., and Pang, Y. (2017). Learning pooling for convolutional neural network. *Neurocomputing*, 224:96–104.

[76] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[77] Tang, H., Ortis, A., and Battiato, S. (2019). The impact of padding on image classification by using pre-trained convolutional neural networks. In *International Conference on Image Analysis and Processing*, pages 337–344. Springer.

[78] Tsang, S.-H. (2018). Review: Resnet—winner of ilsvrc 2015 (image classification, localization, detection).

[79] Veit, A., Wilber, M., and Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. *arXiv preprint arXiv:1605.06431*.

[80] Wang, J., Perez, L., et al. (2017). The effectiveness of data augmentation in image classification using deep learning. *Convolutional Neural Networks Vis. Recognit*, 11.

[81] Wang, S.-H., Phillips, P., Sui, Y., Liu, B., Yang, M., and Cheng, H. (2018). Classification of alzheimer's disease based on eight-layer convolutional neural network with leaky rectified linear unit and max pooling. *Journal of medical systems*, 42(5):1–11.

[82] Xiaosong Wang, Yifan Peng, L. L. Z. L. M. B. R. S. (2017). Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases, pp. 3462-3471.

[83] Y. Zhang, Z. Dong, A. L. e. a. (2015). Magnetic resonance brain image classification via stationary wavelet transform and generalized eigenvalue proximal support vector machine,. *Journal of Medical Imaging and Health Informatics*, 5(7):1395–1403.

[84] Yadav, S. S. and Jadhav, S. M. (2019). Deep convolutional neural network based medical image classification for disease diagnosis. *Journal of Big Data*, 6(1):1–18.

[85] Yang, M.-H., Kriegman, D. J., and Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE Transactions on pattern analysis and machine intelligence*, 24(1):34–58.

[86] Yu, D., Wang, H., Chen, P., and Wei, Z. (2014). Mixed pooling for convolutional neural networks. In *International conference on rough sets and knowledge technology*, pages 364–375. Springer.

[87] Yu, X. and Wang, S.-H. (2019). Abnormality diagnosis in mammograms by transfer learning based on resnet18. *Fundamenta Informaticae*, 168(2-4):219–230.

[88] Zaniolo, L. and Marques, O. (2020). On the use of variable stride in convolutional neural networks. *Multimedia Tools and Applications*, 79(19):13581–13598.

[89] Zhao, Z., Li, X., Liu, F., Zhu, G., Ma, C., and Wang, L. (2020). Prediction of the covid-19 spread in african countries and implications for prevention and control: A case study in south africa, egypt, algeria, nigeria, senegal and kenya. *Science of the Total Environment*, 729:138959.

[90] Zhou, M., Pan, Z., Liu, Y., Zhang, Q., Cai, Y., and Pan, H. (2019). Leak detection and location based on islmd and cnn in a pipeline. *IEEE Access*, 7:30457–30464.

[91] Zoumpourlis, G., Doumanoglou, A., Vretos, N., and Daras, P. (2017). Non-linear convolution filters for cnn-based learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4761–4769.

Appendix A

Python Codes

We will break down the python codes into the training and inference section. The training section discusses the use of the pre-trained model as a feature extractor, we will not alter the parameter of the state-of-the-art algorithm. We then supply our parameters and fine-tune the model. The inference section will use the saved fine-tuned model and measure the model performance tests on the test datasets.

A.1 Trained model

```
1
2  # Laban Bore
3  # April 2021
4  # Training code for covid detection by finetuning ResNet18
5
6  # Inference code for covid detection
7
8  from __future__ import print_function, division
9  import torch, os, copy, time, pickle
10 import torch.nn as nn
11 import torch.nn.functional as F
12 import torch.optim as optim
13 from torch.optim import lr_scheduler
14 import torchvision # packages for loading the data
15 from torchvision import datasets,models, transforms
16 from torch.autograd import Variable
17 import matplotlib.pyplot as plt
18 import numpy as np
19 from PIL import Image
20 import pandas as pd
21 from torchvision.datasets.folder import IMG_EXTENSIONS
```

```
from torchvision.datasets import ImageFolder
from sklearn.metrics import confusion_matrix
import glob, pickle
import seaborn as sn
import argparse
import cv2

start_time= time.time()

plt.ion()   # interactive mode

# parser.add_argument("-f", "--fff", help="a dummy argument to fool
    ipython", default="1")
# set the epochs number of iteration
# parser.add_argument('--dataset_path', type=str, default='./data/',
#                       help='training and train/test dataset') # path
    for test and train dataset

parser = argparse.ArgumentParser(description='COVID-19 Positive
    Detection from X-ray Images')
parser.add_argument('--test_covid_path', type=str, default='C:/Users/
    Borel/x-rays-datasets/train/covid/',
                        help='Positive COVID-19 test samples directory')
parser.add_argument('--test_non_covid_path', type=str, default='C:/
    Users/Borel/x-rays-datasets/test/non',
                        help='Negative COVID test samples directory')
parser.add_argument('--epochs', type=int, default=100,
                    help='number of epochs to train (default: 100)')
parser.add_argument('--trained_model_path', type=str, default='C:/
    Users/Borel/covid_resnet18_epoch2.pt',
                        help='The path and name of trained model')
parser.add_argument("-f", "--fff", help="a dummy argument to fool
    ipython", default="1")
parser.add_argument('--cut_off_threshold', type=float, default= 0.2,
                    help='cut-off threshold. Any sample with
    probability higher than this is considered COVID-19 (default: 0.2)'
    )
parser.add_argument('--batch_size', type=int, default=20,
```

```
                    help='input batch size for training (default: 20)'
    )
parser.add_argument('--num_workers', type=int, default=0,
                    help='number of workers to train (default: 0)')
parser.add_argument('--learning_rate', type=float, default=0.0001,
                    help='learning rate (default: 0.0001)') # a faster
    learning rate of .01%
parser.add_argument('--momentum', type=float, default=0.9,
                    help='momentum (default: 0.9)') #set the momentum
    of 90%
args = parser.parse_args()

# Data augmentation and normalization for training
data_transforms = {
    'train': transforms.Compose([
        transforms.Resize(224),
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224,
    0.225])
    ]),
    'test': transforms.Compose([
        transforms.Resize(224),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224,
    0.225])
    ])
} # function to transform the train and test data

data_dir= "C:/Users/Borel/x-rays-datasets/" # path to the local
    directory of the data

#data_dir = args.dataset_path

image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x),
    data_transforms[x])
```

```
                  for x in ['train', 'test']} #apply the
    data_transform func to dataset in the folder train and test

dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x],
    batch_size= args.batch_size,
                                              shuffle=True, num_workers
    = args.num_workers) # load the images under the folders in batches
                for x in ['train', 'test']}

dataset_sizes = {x: len(image_datasets[x]) for x in ['train', 'test']}
     # define the size of the images
class_names = image_datasets['train'].classes  ## 0: child, and 1:
    nonchild # this shows the folders ['covid', 'non']
class_names_test = image_datasets['test'].classes
dataset_sizes

def imshow(inp, title= None):
    """Imshow for Tensor.""" #show image
    inp = inp.numpy().transpose((1, 2, 0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    inp = std * inp + mean
    inp = np.clip(inp, 0, 1)
    plt.imshow(inp)
    if title is not None:
        plt.title(title)
    plt.pause(0.001)  # pause a bit so that plots are updated

    plt.imshow()

# Get a batch of training data and visualize a few images
inputs, classes = next(iter(dataloaders['train']))

# Make a grid from batch
out = torchvision.utils.make_grid(inputs)

imshow(out, title=[class_names[x] for x in classes])

```

```
115 def train_model(model, criterion, optimizer, scheduler, batch_size,
    num_epochs= 20): #construct the model using train dataset
116     since = time.time()
117 
118     best_model_wts = copy.deepcopy(model.state_dict())
119     best_acc = 0.0
120     train_acc= list()
121     test_acc= list()
122 
123     for epoch in range(num_epochs):
124         print('Epoch {}/{}'.format(epoch+1, num_epochs))
125         print('-' * 10)
126 
127         # Each epoch has a training and validation/test phase
128         for phase in ['train', 'test']:
129             if phase == 'train':
130                 scheduler.step()
131                 model.train()  # Set model to training mode
132             else:
133                 model.eval()   # Set model to evaluate/test mode
134 
135             running_loss = 0.0
136             running_corrects = 0
137             running_prec= 0.0
138             running_rec = 0.0
139             running_f1  = 0.0
140 
141             # Iterate over data.
142             cur_batch_ind= 0
143             for inputs, labels in dataloaders[phase]:
144                 #print(cur_batch_ind,"batch inputs shape:", inputs.
    shape)
145                 #print(cur_batch_ind,"batch label shape:", labels.
    shape)
146                 inputs = inputs.to(device)
147                 labels = labels.to(device)
148 
149 
150                 # zero the parameter gradients
```

```
151                 optimizer.zero_grad()
152
153                 # forward
154                 # track history if only in train
155                 with torch.set_grad_enabled(phase == 'train'):
156                     outputs = model(inputs)
157                     _, preds = torch.max(outputs, 1)
158                     loss = criterion(outputs, labels)
159
160                     # backward + optimize only if in training phase
161                     if phase == 'train':
162                         loss.backward()
163                         optimizer.step()
164
165                 # statistics
166                 running_loss += loss.item() * inputs.size(0)
167                 running_corrects += torch.sum(preds == labels.data)
168
169                 cur_acc= torch.sum(preds == labels.data).double()/
    batch_size
170                 cur_batch_ind +=1
171                 print("\npreds:", preds)
172                 print("label:", labels.data)
173                 print("%d-th epoch, %d-th batch (size=%d), %s acc= %.3
    f \n" %(epoch+1, cur_batch_ind, len(labels), phase, cur_acc ))
174
175                 if phase=='train':
176                     train_acc.append(cur_acc)
177                 else:
178                     test_acc.append(cur_acc)
179
180             epoch_loss= running_loss / dataset_sizes[phase]
181             epoch_acc = running_corrects.double() / dataset_sizes[
    phase]
182
183             print('{} Loss: {:.4f} Acc: {:.4f} \n\n'.format(
184                 phase, epoch_loss, epoch_acc))
185
186             # deep copy the model
```

```
            if phase == 'test' and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_epoch= epoch
                best_model_wts = copy.deepcopy(model.state_dict())

    time_elapsed = time.time() - since
    print('Training complete in {:.0f}m {:.0f}s'.format(
        time_elapsed // 60, time_elapsed % 60))
    print('Best test Acc= %.3f at Epoch: %d' %(best_acc,best_epoch) )

    # load best model weights
    model.load_state_dict(best_model_wts)
    return model, train_acc, test_acc

def visualize_model(model, num_images= 64):
    was_training = model.training
    model.eval()
    images_so_far = 0
    fig = plt.figure()

    with torch.no_grad():
        for i, (inputs, labels) in enumerate(dataloaders['test']):
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)

            for j in range(inputs.size()[0]):
                images_so_far += 1
                ax = plt.subplot(num_images/8, 8, images_so_far)
                ax.axis('off')
                ax.set_title('predicted: {}'.format(class_names[preds[
    j]]))
                imshow(inputs.cpu().data[j])

                if images_so_far == num_images:
                    model.train(mode=was_training)
                    return
```

```
225         model.train(mode=was_training)
226
227 #1. ConvNet as fixed feature extractor
228
229 # Here, we need to freeze all the network except the final layer.
230 # We need to set requires_grad == False to freeze the parameters so
    that the gradients are not computed in backward()
231
232 ## Load the pretrained model and reset final fully connected layer
233 model_conv = torchvision.models.resnet18(pretrained=True) #load the
    pretrained ResNet18 model
234 for param in model_conv.parameters():
235     param.requires_grad = False
236
237 # Parameters of newly constructed modules have requires_grad=True by
    default
238
239 # Here the size of each output sample is set to 2.
240 # Alternatively, it can be generalized to nn.Linear(num_ftrs, len(
    class_names)).
241 num_ftrs = model_conv.fc.in_features
242 model_conv.fc = nn.Linear(num_ftrs, 2)
243
244 model_conv = model_conv.to(device)
245 criterion = nn.CrossEntropyLoss()
246 #criterion = nn.BCELoss()
247
248 # Observe that only parameters of final layer are being optimized as
249 # opoosed to before.
250 optimizer_conv = optim.SGD(model_conv.fc.parameters(), lr= args.
    learning_rate, momentum= args.momentum)
251
252 # Decay LR by a factor of 0.1 every 7 epochs
253 exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=7,
    gamma=0.1)
254
255 # Train and evaluate
256 if __name__ == "__main__":
```

```
257     model_conv, train_acc, test_acc = train_model(model_conv,
        criterion, optimizer_conv, exp_lr_scheduler, args.batch_size,
        num_epochs= args.epochs)
258     model_conv.eval()
259     torch.save(model_conv, './covid_resnet18_epoch%d.pt' %args.epochs
        ) # save the model here for inferences
260
261 end_time= time.time()
262 print("total_time tranfer learning=", end_time - start_time)
263
264 visualize_model(model_conv) #visualize the model
265
266 # 2. Finetuning the convnet
267
268 # The model updates the parameters and use the new parameters. We will
      use the new parameters from one used initially in the pre-trained
      model
269 # We need to set requires_grad == TRUE to use updated parameters, the
      gradients are computed in backward()
270 model_ft = models.resnet18(pretrained=True)
271 num_ftrs = model_ft.fc.in_features
272
273 # Here the size of each output sample is set to 2.
274 # Alternatively, it can be generalized to nn.Linear(num_ftrs, len(
      class_names)).
275 model_ft.fc = nn.Linear(num_ftrs, 2)
276
277 model_ft = model_ft.to(device)
278
279 criterion = nn.CrossEntropyLoss()
280
281 # Observe that all parameters are being optimized
282 optimizer_ft = optim.SGD(model_ft.parameters(),  lr= args.
      learning_rate, momentum= args.momentum)
283
284 # Decay LR by a factor of 0.1 every 7 epochs
285 exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7,
      gamma=0.1)
286
```

```
# Train and evaluate
if __name__ == "__main__":
    model_ft, train_acc, test_acc = train_model(model_ft, criterion,
    optimizer_ft, exp_lr_scheduler, args.batch_size, num_epochs= args.
    epochs)
    model_ft.eval()
    torch.save(model_ft, './covid_resnet18_epoch%d.pt' %args.epochs )
    #save the model for inference part.

visualize_model(model_ft)
```

A.2 Inferences

```

# Utility function to find sensitivity and specificity for different
    cut-off thresholds
def find_sens_spec( covid_prob, noncovid_prob, thresh):
    sensitivity= (covid_prob >= thresh).sum()   / (len(covid_prob)+1e
    -10)
    specificity= (noncovid_prob < thresh).sum() / (len(noncovid_prob)
    +1e-10)
    print("sensitivity= %.3f, specificity= %.3f" %(sensitivity,
    specificity))
    return sensitivity, specificity

class_names = ['covid','non']
fig, axs = plt.subplots(3,5, figsize=(100, 100), dpi=100)
fig.subplots_adjust( hspace=-0.5,wspace=0.2)
axs = axs.ravel()
for i, img in enumerate(glob.glob('C:/Users/Borel/x-rays-datasets/
    common-disease/*')):
    image = cv2.imread(img)
    axs[i].axis('off')
    axs[i].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    axs[i].set_title(img[-20:-3],fontsize= 50)
    #plt.savefig('./chest-xrays-images.png')
    fig.savefig('chest-xrays-images.png') # sample display of positive
```

```
    and negative covid-19 images

# Test on trained model
model_name= args.trained_model_path
model= torch.load(model_name, map_location='cpu')
#model.eval()

# loading new images
imsize= 224
loader = transforms.Compose([transforms.Resize(imsize),
                             transforms.CenterCrop(224),
                             transforms.ToTensor(),
                             transforms.Normalize([0.485, 0.456,
   0.406], [0.229, 0.224, 0.225])
                             ])

def image_loader(image_name):
    """load image, returns cuda tensor"""
    image = Image.open(image_name).convert("RGB")
    image = loader(image).float()
    image = Variable(image, requires_grad=True)
    image = image.unsqueeze(0)  #this is for VGG, may not be needed
   for ResNet
    return image

sm = torch.nn.Softmax()
# Get the predicted probabilities of all samples
test_covid  = glob.glob("%s*" %args.test_covid_path) #84
test_non    = glob.glob("%s*" %args.test_non_covid_path) #2000

covid_pred= np.zeros([len(test_covid),1]).astype(int)
non_pred  = np.zeros([len(test_non),1]).astype(int)

covid_prob= np.zeros([len(test_covid),1])
non_prob   = np.zeros([len(test_non),1])

for i in range(len(test_covid)):
    cur_img= image_loader(test_covid[i])
    model_output= model(cur_img)
```

```
    cur_pred = model_output.max(1, keepdim=True)[1]
    cur_prob = sm(model_output)
    covid_prob[i,:]= cur_prob.data.numpy()[0,0]
    print("%03d Covid predicted label:%s" %(i, class_names[int(
    cur_pred.data.numpy())]) )

for i in range(len(test_non)):
    cur_img= image_loader(test_non[i])
    model_output= model(cur_img)
    cur_pred = model_output.max(1, keepdim=True)[1]
    cur_prob = sm(model_output)
    non_prob[i,:]= cur_prob.data.numpy()[0,0]
    print("%03d Non-Covid predicted label:%s" %(i, class_names[int(
    cur_pred.data.numpy())]) )

# Find sensitivity and specificity
thresh= 0.1
sensitivity_40, specificity= find_sens_spec( covid_prob, non_prob,
    thresh)
# Find sensitivity and specificity
thresh= 0.2
sensitivity_40, specificity= find_sens_spec( covid_prob, non_prob,
    thresh)
# Find sensitivity and specificity
thresh= 0.3
sensitivity_40, specificity= find_sens_spec( covid_prob, non_prob,
    thresh)

# Derive labels based on probabilities and cut-off threshold
covid_pred = np.where( covid_prob  >thresh, 1, 0)
non_pred   = np.where( non_prob  >thresh,   1, 0)

# Derive confusion-matrix
covid_list= [int(covid_pred[i]) for i in range(len(covid_pred))]
covid_count = [(x, covid_list.count(x)) for x in set(covid_list)]

non_list= [int(non_pred[i]) for i in range(len(non_pred))]
non_count = [(x, non_list.count(x)) for x in set(non_list)]

```

```
y_pred_list= covid_list+non_list
y_test_list= [1 for i in range(len(covid_list))]+[0 for i in range(len
    (non_list))]

y_pred= np.asarray(y_pred_list, dtype=np.int64)
y_test= np.asarray(y_test_list, dtype=np.int64)

cnf_matrix = confusion_matrix(y_test, y_pred)
np.set_printoptions(precision=2)

# Plot normalized confusion matrix
df_cm = pd.DataFrame(cnf_matrix, index = [i for i in class_names],
                                 columns = [i for i in class_names])

ax = sn.heatmap(df_cm, cmap=plt.cm.Blues, annot=True, cbar=False, fmt=
    'g', xticklabels= ['COVID-19 Negative','COVID-19 Positive'],
    yticklabels= ['COVID-19 Negative','COVID-19 Positive'])
ax.set_title("Confusion matrix")
plt.savefig('./confusion_matrix.png') #dpi = 200

# plot the predicted probability distribution
bins = np.linspace(0, 1, 30)
plt.subplot(211)
plt.hist(covid_prob, bins, color= '#70cab9', histtype = 'bar', label='
    Probabilities of COVID-19 positive at thresh=0.1 ')
plt.ylim([0,10])
plt.legend(loc='upper right')
plt.subplot(212)
plt.hist(non_prob, bins, color= 'red', label='Probabilities of COVID
    -19 negative at thresh=0.1  ')
plt.legend(loc='upper right')
plt.savefig('./scores_histogram-thesis-best.png') #dpi = 200

# ROC Curve and AUC
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot

y_test_res18= [1 for i in range(len(covid_prob))]+[0 for i in range(
```

```
    len(non_prob))]
y_pred_res18= [covid_prob[i] for i in range(len(covid_prob))]+[
    non_prob[i] for i in range(len(non_prob))]

auc_res18 = roc_auc_score(y_test_res18, y_pred_res18)
ns_fpr_res18, ns_tpr_res18, _ = roc_curve(y_test_res18, y_pred_res18)

plt.figure()
pyplot.plot(ns_fpr_res18, ns_tpr_res18,  color='blue', linewidth=3,
    label='ResNet18,         AUC= %.3f' %auc_res18)
pyplot.ylim([0,1.05])
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.title("ROC Curve")
# show the legend
pyplot.legend(loc='lower right')
plt.savefig('./ROC_covid19-thesis-best.png') #dpi = 200

#accuracy
#precision
#recall

from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import confusion_matrix

precision = precision_score(y_test_res18, y_pred)
print('precision: %f' % precision)

accuracy = accuracy_score(y_test_res18, y_pred)
print('accuracy: %f' % accuracy)

recall = recall_score(y_test_res18, y_pred)
print('recall: %f' % recall)

f1 = f1_score(y_test_res18, y_pred)
```

```
160 print('f1: %f' % f1)
161
162 kappa = cohen_kappa_score(y_test_res18, y_pred)
163 print('Cohens kappa: %f' % kappa)
164
165 auc = roc_auc_score(y_test_res18, y_pred_res18)
166 print('ROC AUC: %f' % auc)
167
168
169 matrix = confusion_matrix(y_test_res18, y_pred)
170 print(matrix)
171 #precision_score(y_test_res18, y_pred_res18, average=None)
172 #auc = roc_auc_score(y_test_res18, y_pred)
173 #print('ROC AUC: %f' % auc)
174
175 end_time= time.time()
176 tot_time= end_time- start_time
177 print("\nTotal Time:", tot_time)
```

Appendix B

Similarity Report

Curiginal

Document Information

Analyzed document	Msc_Graduate-Thesis_Statistical_Science_Laban_Bore.pdf (D114144369)
Submitted	2021-10-03 22:56:00
Submitted by	
Submitter email	borelaban@gmail.com
Similarity	8%
Analysis address	library.strath@analysis.urkund.com

Sources included in the report

W	URL: https://su-plus.strathmore.edu/bitstream/handle/11071/6789/Consumer%20credit%20risk%20modelling%20using%20machine%20learning%20algorithms.pdf?sequence=3&isAllowed=y Fetched: 2021-06-15 08:15:58	6
SA	**Face recognition report.pdf** Document Face recognition report.pdf (D107947409)	3
W	URL: https://arxiv.org/pdf/2007.14777 Fetched: 2021-05-31 00:43:41	1
SA	**Image_Classification_using_DL__A_Survey(3).pdf** Document Image_Classification_using_DL__A_Survey(3).pdf (D108434188)	1
W	URL: https://www.researchgate.net/publication/350202823_Deep-Chest_Multi-Classification_Deep_Learning_Model_for_Diagnosing_COVID-19_Pneumonia_and_Lung_Cancer_Chest_Diseases Fetched: 2021-10-03 23:01:00	1
W	URL: https://jeremyweidner.github.io/Stop_Sign_Detection.pdf Fetched: 2020-12-20 22:37:19	1
W	URL: https://www.medrxiv.org/content/medrxiv/early/2021/02/08/2021.02.06.21251271.source.xml Fetched: 2021-05-05 08:21:08	4
SA	**RR.pdf** Document RR.pdf (D109057024)	2
SA	**Xiaoyu Chen_Dissertation_xc866.pdf** Document Xiaoyu Chen_Dissertation_xc866.pdf (D112371002)	11
W	URL: https://arxiv.org/pdf/2011.05543 Fetched: 2021-08-06 14:22:39	3
SA	**final.pdf** Document final.pdf (D72406536)	1
	URL: https://arxiv.org/pdf/1807.10406	

Appendix C

Ethical Approval Letter

Strathmore UNIVERSITY

7th July 2021

Mr Bore Laban,
borelaban@gmail.com

Dear Mr Bore,

RE: Finetune ResNet18: Classification of COVID-19 Positive Images Using Deep Convolution Neural Network

This is to inform you that SU-IERC has reviewed and **approved** your above **master's** research proposal. Your application reference number is **SU-IERC1058/21.** The approval period is **7th July 2021 to 6th July 2022.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, MTA) will be used
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-IERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-IERC within 48 hours of notification
iv. Any changes, anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-IERC within 48 hours
v. Clearance for export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days upon completion of the study to SU-IERC.

Prior to commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and also obtain other clearances needed.

Yours sincerely,

for: Dr Virginia Gichuru,
Secretary; SU-IERC

STRATHMORE UNIVERSITY INSTITUTIONAL ETHICS REVIEW COMMITTEE (SU-IERC)
07 Jul 2021
TEL: +254 (0)70303400
P.O BOX 59857-00200
NAIROBI-KENYA

Cc: Prof Fred Were,
Chairperson; SU-IERC

Ole Sangale Rd, Madaraka Estate. PO Box 59857-00200, Nairobi, Kenya. Tel +254 (0)703 034000
Email admissions@strathmore.edu www.strathmore.edu