



**NEAR EAST UNIVERSITY**

**INSTITUTE OF GRADUATE STUDIES**

**DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING**

**MILD AND SEVERE COVID DETECTION USING DEEP LEARNING**

**M.Sc. THESIS**

**Balla Moussa TRAORE**

**Nicosia**

**June, 2022**

**BALLA MOUSSA  
TRAORE**

**MILD AND SEVERE COVID DETECTION USING  
DEEP LEARNING**

**MASTER THESIS**

**2022**



**NEAR EAST UNIVERSITY**  
**INSTITUTE OF GRADUATE STUDIES**  
**DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING**

**MILD AND SEVERE COVID DETECTION USING DEEP LEARNING**

**M.Sc. THESIS**

**Balla Moussa TRAORE**

**Supervisor**



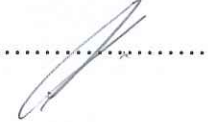
**Assoc. Prof. Dr. Sertan SERTE**

**Nicosia**


**June, 2022**

### Approval


We certify that we have read the thesis submitted by Balla Moussa Traore titled “Mild and Severe COVID Detection Using Deep Learning” and that in our combined opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Sciences.

| Examining Committee    | Name-Surname                    | Signature   |
|------------------------|---------------------------------|---|
| Head of the Committee: | Assist. Prof. Ali Serener       |  |
| Committee Member:      | Assist. Prof. Cemal Kavalcioglu |  |
| Supervisor:            | Assoc. Prof. Sertan Serte       |  |

Approved by the Head of the Department

  
12/9/2022  
.....  
Prof. Dr. Bulent Bilgehen  
Head of Department

Approved by the Institute of Graduate Studies

...../...../2022  
Prof. Dr. Kemal Husnü Can Başer  
  
Head of the Institute  


### **Declaration**

I hereby declare that all information, documents, analysis and results in this thesis have been collected and presented according to the academic rules and ethical guidelines of Institute of Graduate Studies, Near East University. I also declare that as required by these rules and conduct, I have fully cited and referenced information and data that are not original to this study.

Balla Moussa Traore

...../...../.....

### **Acknowledgments**

I would like to express my utmost gratitude to Assist. Prof. Dr. Ali Serener for his constant output and support throughout this study and Assoc. Prof. Dr. Sertan SERTE for allowing me to present this document and accommodating with the various issues encountered along the way.

**Balla Moussa Traore**

## **Abstract**

### **Mild and Severe COVID Detection Using Deep Learning**

**Balla Moussa Traore**

**MSc, Department of Electrical Electronic Engineering**

**June, 2022, 70 pages**

The coronavirus disease of 2019 caused most countries worldwide to go into lockdown due to its highly contagious nature. It had a lot of economic ramifications that are still felt to this day. The first step to fighting a disease is to recognize it. Radiography imagery is one of the methods used to detect COVID-19, and allows doctors to act quickly on it upon discovery.

Machine Learning has come a long way during the last decade, and make use of deep convolutional learning network to recognize everyday objects of images with high accuracy, comparable with that of human beings. With that in mind, the objective is to train deep learning networks to recognize COVID cases. Knowing the fact that computer have faster processing power than human being it allows for faster course of action.

This thesis describes how a deep convolutional neural network was built and trained to detect two different progressions of COVID, that is, mild COVID and severe COVID. Three main questions were to be answered during the study. First, can the network detect COVID? Second, can the network differentiate between a mild COVID case and a severe COVID case? Thirdly, can the network differentiate between COVID and other diseases? The way in which overfitting and vanishing gradient problems were dealt with is also mentioned and talked about in the document. The resulting network contains 25 layers, including multiple convolutional layers with ReLU (Rectified Linear Unit), and some fully connected layers. Five experiments have been set up to test the network performance and answer the previously raised questions. As a result, the non-pretrained network that was designed achieves an accuracy of 88.20 % when classifying X-ray scans of COVID or healthy patients. the model is also quite adept at recognizing CT-scans, as it is able to differentiate between severe and mild cases of COVID with a 93.7% accuracy. Some experiments involved cases of cancer and Pneumonitis in order to

check if the network could also differentiate between cases of COVID and potentially other disease.



## TABLE OF CONTENTS

|  |    |
|--|----|
| List of Tables.....                                  | 11 |
| List of figures .....                                | 12 |
| List of Abbreviations.....                           | 13 |
| CHAPTER I: Introduction .....                        | 14 |
| CHAPTER II: Background and Literature Review .....   | 17 |
| Overview .....                                       | 17 |
| Theoretical Framework .....                          | 17 |
| 2.1 Deep Convolutional neural networks.....          | 17 |
| 2.2 DCNN Architecture .....                          | 17 |
| 2.3 ALEXNET .....                                    | 28 |
| Related Work .....                                   | 30 |
| Summary .....  | 31 |
| CHAPTER III: Methodology .....                       | 33 |
| Overview .....                                       | 33 |
| 3.1 Datasets .....                                   | 33 |
| 3.2 Network architecture.....                        | 38 |
| 3.3 Training .....                                   | 42 |
| A. Experiment 1: X-Ray: Healthy vs Covid.....        | 42 |
| B. Experiment 2: CT-Scan: Healthy vs Malignant.....  | 44 |
| C. Experiment 3: CT-Scan: Severe vs Mild .....       | 45 |
| D. Experiment 4: CT-Scan: Severe vs Pneumonitis..... | 46 |
| E. Experiment 5: CT-Scan: Mild vs Pneumonitis .....  | 48 |
| Summary .....  | 50 |
| CHAPTER IV: Results and Findings.....                | 51 |
| Overview .....                                       | 51 |
| 4.1 Results and Findings .....                       | 51 |
| A. Experiment 1 .....                                | 51 |
| B. Experiment 2.....                                 | 54 |
| C. Experiment 3.....                                 | 56 |
| D. Experiment 4.....                                 | 59 |
| E. Experiment 5.....                                 | 61 |
| Summary .....  | 63 |
| CHAPTER V: Discussion and Conclusion.....            | 64 |

|  |    |
|--|----|
| 5.1 Discussion .....                                     | 64 |
| 5.2 Conclusion .....                                     | 65 |
| REFERENCES .....   | 66 |
| APPENDICES .....   | 70 |
| Appendix A: Binary Classification Code.....              | 70 |
| Appendix B: Code for Resizing Images in the Dataset..... | 73 |

## List of Tables

|               |    |
|---------------|----|
| Table 1.....  | 34 |
| Table 2.....  | 40 |
| Table 3.....  | 42 |
| Table 4.....  | 43 |
| Table 5.....  | 44 |
| Table 6.....  | 44 |
| Table 7.....  | 45 |
| Table 8.....  | 45 |
| Table 9.....  | 47 |
| Table 10..... | 47 |
| Table 11..... | 48 |
| Table 12..... | 49 |
| Table 13..... | 52 |
| Table 14..... | 52 |
| Table 15..... | 54 |
| Table 16..... | 55 |
| Table 17..... | 56 |
| Table 18..... | 57 |
| Table 19..... | 59 |
| Table 20..... | 59 |
| Table 21..... | 61 |
| Table 22..... | 62 |

## List of figures

|                 |    |
|-----------------|----|
| Figure 1 .....  | 18 |
| Figure 2 .....  | 19 |
| Figure 3 .....  | 19 |
| Figure 4 .....  | 20 |
| Figure 5 .....  | 21 |
| Figure 6 .....  | 22 |
| Figure 7 .....  | 23 |
| Figure 8 .....  | 24 |
| Figure 9 .....  | 24 |
| Figure 10 ..... | 25 |
| Figure 11 ..... | 26 |
| Figure 12 ..... | 27 |
| Figure 13 ..... | 28 |
| Figure 14 ..... | 29 |
| Figure 15 ..... | 35 |
| Figure 16 ..... | 36 |
| Figure 17 ..... | 37 |
| Figure 18 ..... | 41 |
| Figure 19 ..... | 43 |
| Figure 20 ..... | 45 |
| Figure 21 ..... | 46 |
| Figure 22 ..... | 48 |
| Figure 23 ..... | 49 |
| Figure 24 ..... | 53 |
| Figure 25 ..... | 54 |
| Figure 26 ..... | 55 |
| Figure 27 ..... | 56 |
| Figure 28 ..... | 58 |
| Figure 29 ..... | 58 |
| Figure 30 ..... | 60 |
| Figure 31 ..... | 61 |
| Figure 32 ..... | 62 |
| Figure 33 ..... | 63 |

## List of Abbreviations

**ANN: Artificial Neural Network**

**AUC: Area Under the Curve**

**CNN: Convolutional Neural Network**

**CPU: Central Processing Unit**

**CT: Computerized Tomography**

**DCNN: Deep Convolutional Neural Network**

**FN: False Negative**

**FP: False Positive**

**GPU: Graphics Processing Unit**

**ReLU: Rectified Linear Unit**

**ROC: Receiving Operating Characteristic**

**SVM: Support Vector Machine**

**Tanh: Hyperbolic Tangent**

**TN: True Negative**

**TP: True Positive**

## CHAPTER I: Introduction

As reported by the World Health Organization, on December 31<sup>st</sup> 2019, they received reports about several cases of an unknown viral pneumonia from Chinese province of Wuhan (World health Organization). It would be found to be a novel corona virus that would soon be declared to be a pandemic that would trigger an unprecedented wave of confinement around the world. Needless to say, countless lives were lost, businesses were shut down, and the economic ramifications are still present.

Among the testing method available for the COVID-19, X-ray imaging is of particular interest in this study. It provides a visual and fast way for physicians to identify infected individuals, and it providing easily accessible materials for deep learning among other fields. Today, artificial intelligence and deep learning are an integral part of many field sectors, such as industrial, military, medicinal applications and more. Convolutional Neural Networks (CNN) are well suited for imaging related classifications, that is, their architecture is easier to train and allows them to reliably find information and features relevant to the pictures through the use of kernels and filters. Their flexibility allows for users to use pre-trained networks and adapt them to suit various situations or needs.

There is a significant amount of pre-trained networks, and one model of interest is AlexNet (Krizhevsky,2012). AlexNet is the topic of multiple studies and will be the main reference for this one. Using transfer learning, the network will be adapted to complete our task our classifying an X-Ray image as COVID case or a Healthy one. Nowadays, a lot research about deep learning in diagnosing COVID-19 is happening with new papers being published every month, and a lot of optimistic results are being found (Nayak et al, 2021; Yi et al, 2021). Therefore, this study aims to bring its own contribution in the research field. It is imperative that physicians are able to quickly and accurately determine the state of a patient, thus making this kind of document relevant for the medicinal and the engineering literature.

This study will be one of many that tries to make use of deep learning to classify visual data into useful diagnosis. It goes without saying that finding fast and accurate ways of detecting COVID cases is of utmost importance in the medical landscape. Finding new and various models and alternative gives more options to the parties

concerned in helping patients being quickly diagnosed in various conditions. The presented model was built to classify X-rays images and CT-scans to help find concerned patient even when operating different tasks.

While data showing images of patient is not scarce, it can be a limitation to find out how the images were obtained and treated. At times during the study, it will be seen that images retrieved from different sources have proved to be a serious challenge in properly measuring the effectiveness of the model. It led to sometimes to almost unrealistically high results, however, it also showed how effective the network was at discerning pictures.

This study will aim at finding if the model presented can provide accuracy and flexibility using different sources of visual data. By the time the reading of this document is done, some interesting questions should have an answer.

- Is the network able to distinguish between two X-Rays involving a COVID case and a healthy one?
- Is the model able to work with both CT-Scan and X-Rays?
- Is the model able to distinguish between a mild COVID case and a severe COVID case using CT-scans?
- Is the model able to distinguish between a case of cancer and a healthy one?
- Is the model able to distinguish between two different diseases?
- Is the model able to distinguish between a case of COVID or a different disease?

We will first take a look at the literature and find more about the work done on our subject; the next section of the chapter will detail a lot more about the related work in the field. It will be seen that there are some already encouraging and already effective models that can be deployed for use, as they show real accuracy sometimes close or better than human test. Secondly, the method used in this model will be explained. A network built from scratch will be employed, that is a model that was not pretrained nor that makes uses of transfer learning. A random weight initialization method will be used, taking advantages of Rectified Linear Unit activation. Our convolutional model will also make use of techniques such as dropout in order to fight against overfitting problems that were definitely encountered during the training process. The document will detail the structure of the

layers forming the network, and extensively discuss AlexNet's model that served as the basis of our model. Finally, the training phase and process of the network will be discussed alongside the results obtained after training, before swiftly bringing a conclusion to this paper.



## **CHAPTER II: Background and Literature Review**

### **Overview**

This part of the chapter will explain the theoretical framework that was followed while building our model. Some important concept related to deep learning and possessing relevance in the context of thesis will be described. It will then include some of the work found in the literature relating to this study.

### **Theoretical Framework**

#### **2.1 Deep Convolutional neural networks**

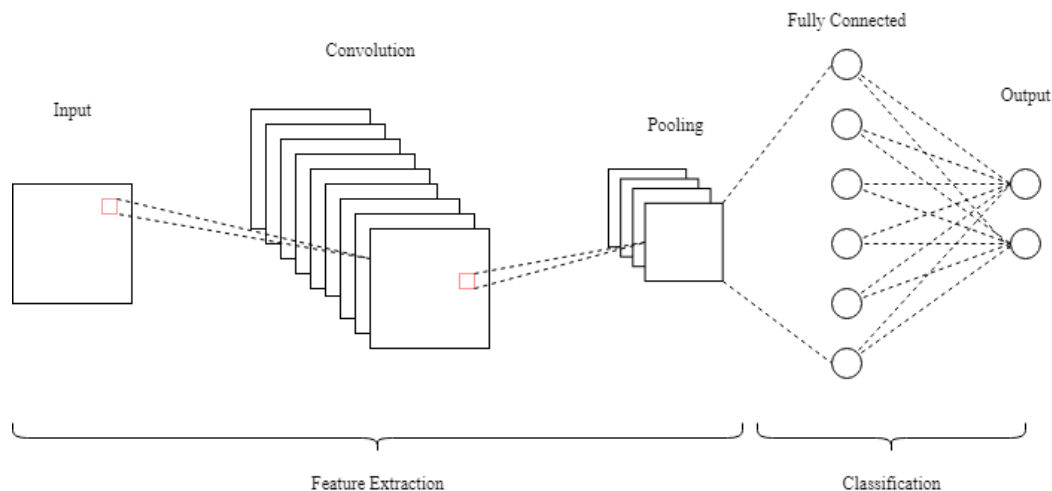
Artificial Neural Networks (ANN), based on the human nervous system, are the main component in the Artificial Intelligence topic. They possess multiples layers which brings complexity in their architecture, allowing them to learn how to perform a wide variety of tasks. One type of neural network is of interest in this document, Deep Convolutional Neural Networks (DCNN). They get their name from the fact that their learning process is based on convolutions between matrices. Their overall architecture includes the combination of one more convolutional layers, pooling layers, and fully connected layers. DCNNs are excellent at dealing with image related problems. They not only require less parameters than ANNs to function, but they are adept are recognizing patterns and extracting features that are propagated through the layers in order to fish for more higher complexity features which are relevant to classify the input or any other image related operations.

#### **2.2 DCNN Architecture**

DCNNs are typically made out of three main types of layers. Those mentioned layers are: the convolutional layers, the pooling layers and the fully connected layers. To that we can include the input and output layers, activation layers and other types of layers needed to improve the learning performance of the network. Figure 1 shows a simple layout of the architecture of a CNN.

Figure 1

*Simplified CNN Architecture Layers (M. Guruchan, 2020)*



The input layer is usually the recipient for the image data which usually comes with a size of a  $H \times W \times 3$  matrix ( $H$  being the height,  $W$  being the width, and 3 representing the number of color channels). The convolution layer, as it will be seen later, is produced through the convolution of filters with the input matrix. An activation function will be then be applied and produce new inputs for the pooling layer, which will then sample them down, which will reduce the number of parameters. The fully connected layer will perform as it would in a standard ANNs by providing scoring value for the classes that are going to be used for the classification purpose.

### 2.2.1 Convolutional Layer

The convolutional layer is the most defining layer in the CNN and DCNN architectures. It essentially works through usage of filters or kernels convolving around the input image. The kernels are small in size but slide along the entire input to compute the scalar product that will build a two-dimensional activation map in the next layer (O'Shea, 2015).

#### A. Convolution

The significance of using convolutions using kernels can be understood if we consider the following example. For sake of our example, let's assume an input image of size  $64 \times 64 \times 3$  going through the layers of our network. In a typical ANN, an efficient way of processing the information provided would for the next layer to have neurons matching the value of the height and width. The resulting number of parameters at this point of the network would be  $64 \times 64 \times 3$  by  $64 \times 64$  which give us

50,331,648 weight connections. A more efficient method would be looking at local regions in the image instead of analyzing weighing every pixel of the picture. Figure 2 shows the connection between a local region in a neuron in the following layer. If the filter window is size selected to be  $3 \times 3$ , we then go from having 50 million parameters to  $5 \times 5 \times 3$  by  $64 \times 64$  neurons equaling 307,200 parameters. This holds true since the weight for the local region in the input layer is kept the same. This means that we go from each neuron having 12,888 weights to them having 75 weights, thus drastically reducing the number of parameters needed (Albawi, 2017).

Figure 2

*Pixel Related Weight Connection of ANN(Albawi,2017)*

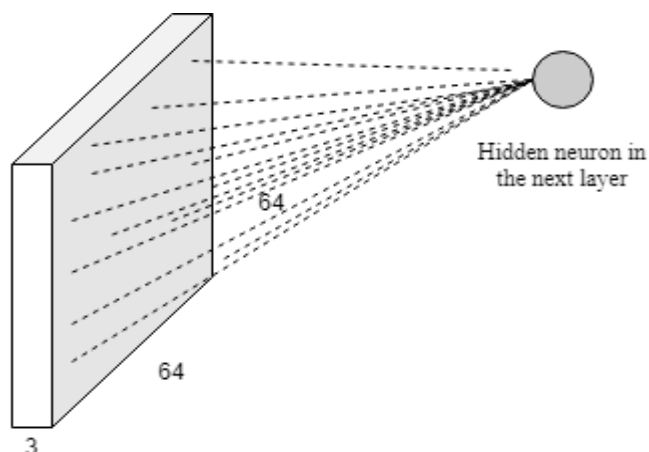
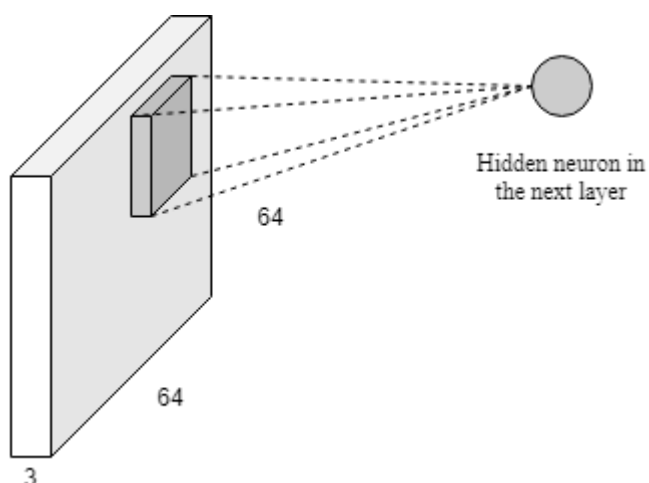


Figure 3

*Local Region Related Weight Connection*



Fixing the weights according the local region not only reduces the number of parameters needed, but also translate into a process similar to gliding a  $5 \times 5 \times 3$

window across the input and storing the output at its corresponding position. This allows the network to detect and retrieve features regardless of their position in the picture. As an example, figure 4 shows how an edge detector  $3 \times 3$  matrix convolving around the image will detect the ‘edge’ features on the input regardless of their position the picture.

Figure 4

*Edge Detection Through Convolution*



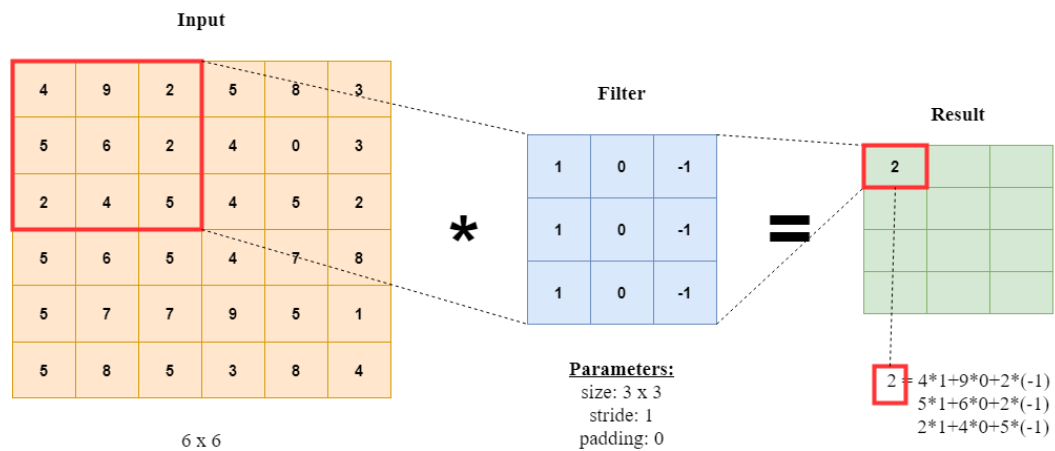
It can be seen that that the edge detecting filter has recognized edges all around the picture without having any limitation with location in the image.

The mathematical expression for a pixel in the next layer is given by equation 1.

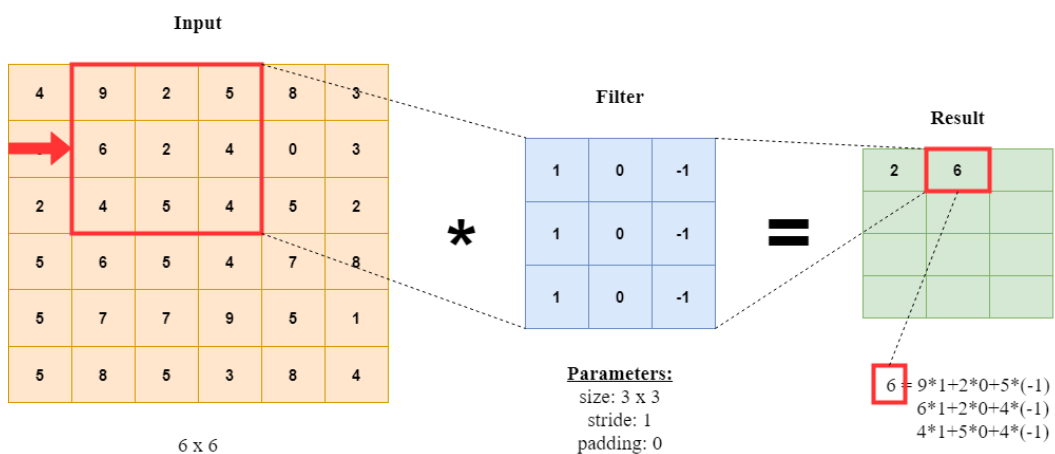
$$P(i, j) = (I * K)[i, j] = \sum_m \sum_n I[m, n]K[i - m, j - n] \quad (1)$$

We have the output in the next layer represented by  $P(i, j)$ ,  $I$  represents the input picture,  $K$  is the kernel matrix, and the convolution operator is  $*$ . Figure 5 shows a visual description of the convolution operation between 2-D matrix and filter.

Figure 5

*Edge Detection through Convolution (Priyono,2018)*

(a)



(b)

As shown by figure 5(a), the filter will convolve with part of the input and produce a result in the corresponding position in the next layer. The convolution window will then slide across the picture with set stride as shown above until the whole image is covered.

**B. Stride**

The stride, along with other parameters such as depth and zero-padding will affect the size of the output produce in the next layer. With those parameters we can reduce or increase the amount data transmitted and processed by the next layers of the network. In actually, the stride controls the amount of overlap between subsequent parts of the input, as shown in figure 6. With heavy overlap the activation produced

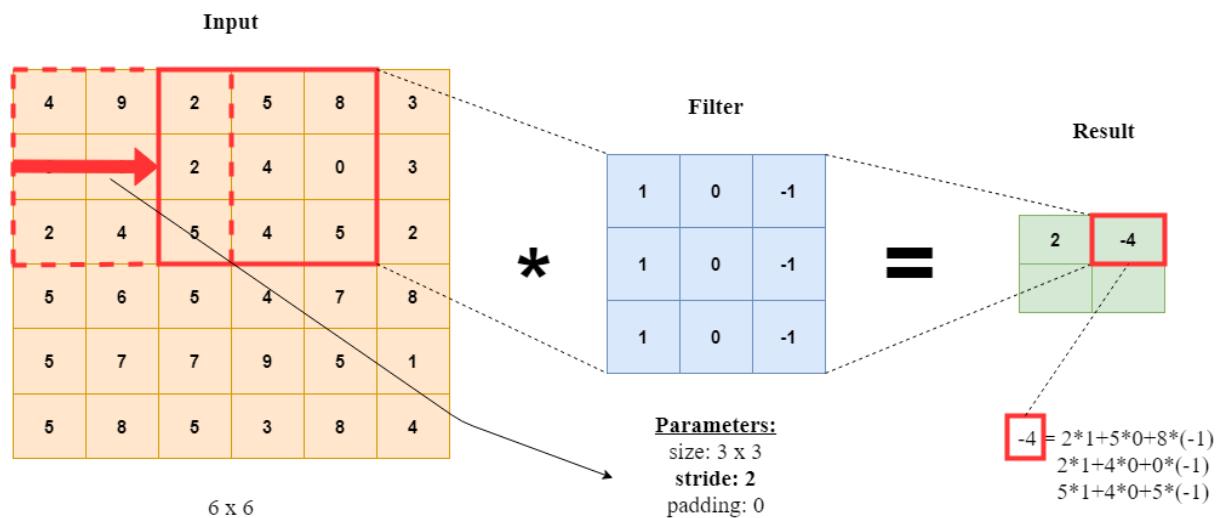
in the next layer will get larger. Equation 2 shows how the output can be computed using the stride, the size of the input and the filter (Albawi,2017).

$$O = 1 + \frac{N - F}{S} \quad (2)$$

where N is the input size, F being the filter dimension, and S is stride window.

Figure 6

*Stride Size 2 Convolution (Priyono,2022)*



When comparing figure 6 and figure 5, the size of the output window can be seen to have decreased. This shows how one can manipulate the output through changing the stride.

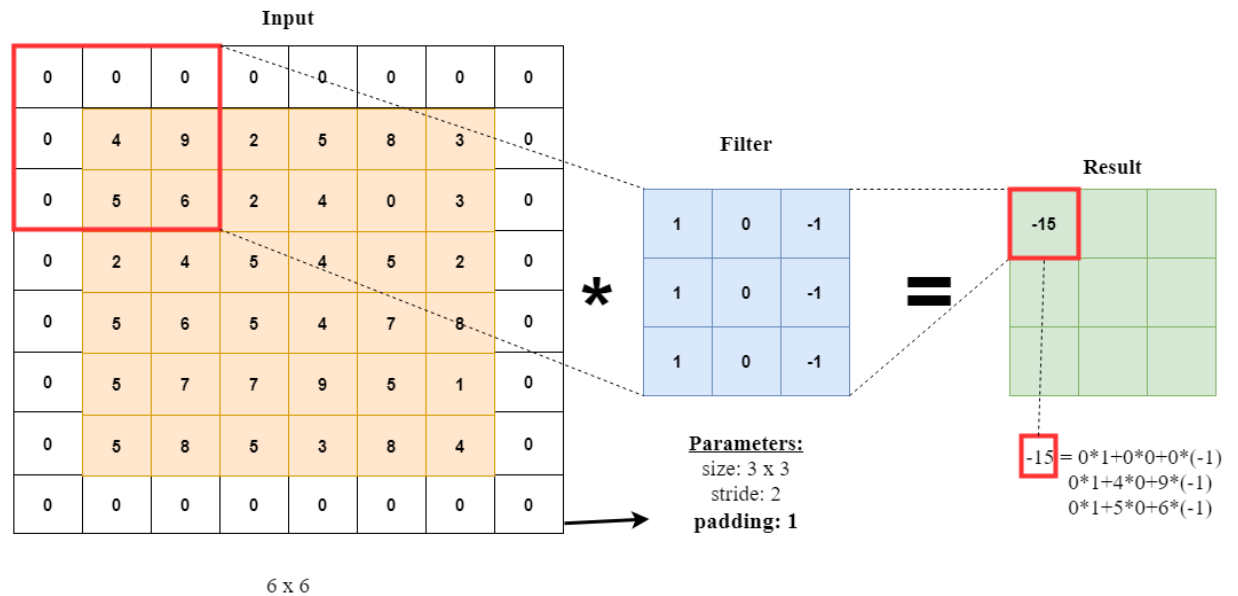
### C. Padding

One might observe that the borders of the picture do not get many opportunities to have their information extracted. In order to avoid loss of information in that area, one parameter of the convolution can be altered, that is padding. For the purpose of our task, we use zero-padding to bring additional rows and columns made up of zeros on the matrix as shown in figure 7. As mentioned above, zero-padding is a parameter that affects the output size in the next layer, as can be seen in the equation 3.

$$O = 1 + \frac{N + 2P - F}{S} \quad (3)$$

where P is the number of row and column added to the input matrix. Figure 7 shows a case where P=1.

Figure 7

*2-D Convolution with Padding (Priyono,2022)*

It can be seen that the output size increased with the padding, and the borders features can be registered in the next layer.

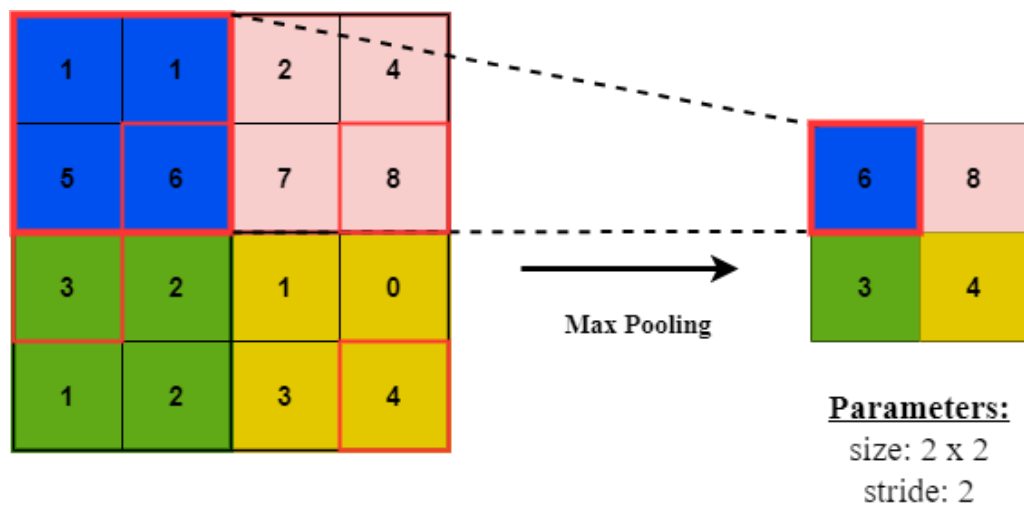
### 2.2.2 Pooling Layer

A pooling layer's main goal is to reduce the complexity of the input by down-sampling it. This process leads to a reduction of parameters used in the network, further reducing the computing time. They are usually used to steadily lower the dimension of the output during the computation.

The most common pooling methods are Max Pooling and Average Pooling. A popular way of using the pooling layer is by setting its window size to  $2 \times 2$  with stride set to 2, both horizontal and vertical. This means that the maximum value contained within that window will be returned in the case of max pooling (figure 8), and the average of all the values inside that window will be returned in an average pooling case (figure 9). Moreover, while setting the stride to be 2 and having the kernel size to  $3 \times 3$ , an overlap is created. It is an efficient way of reducing parameters while keeping extra spatial information.

Figure 8

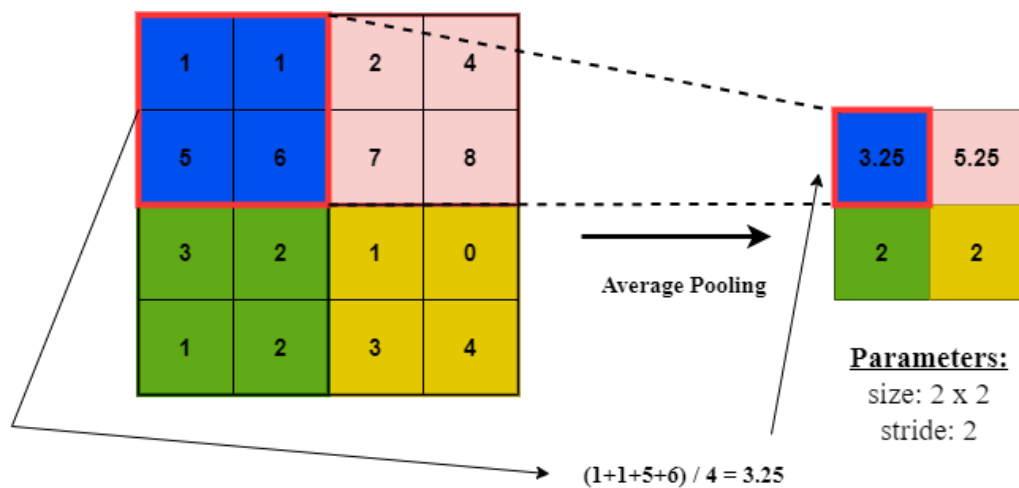
$2 \times 2$  Kernel Max Pooling (Priyono,2018)



The max pooling function will return 6 in the next layer as it is the maximum value in the window. The output position will match that of the previous layer so as to maintain location sensitive data.

Figure 9

$2 \times 2$  Kernel Average Pooling (Priyono,2018)

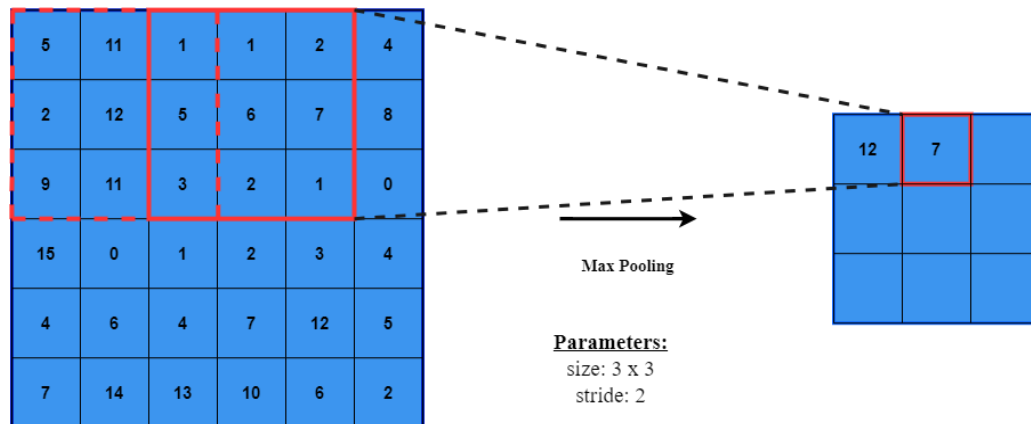


The average pooling layer will find the average value of the data inside the kernel and return it to the next layer.



Figure 10

*3×3 Kernel with Stride 2 Max Pooling (Priyono,2018)*



The kernel size matching the stride would have returned a  $2 \times 2$  matrix as output, however by keeping the stride as 2 an overlap is formed; therefore, the output presents more information about the inner part of the matrix.

### 2.2.3 Fully Connected Layer

Reminiscent of the more traditional artificial neural networks, the fully connected layer is arranged by having each node of it connected to every one of those in the previous layer and potentially the next layer. The fully connected layer presents a lot of complexity because of the number of nodes and the number connections they each have, in other words the number of parameters to work out is quite significant.

As mentioned previously, one of the main advantages of DCNN is the reduction of those parameters through the use of the processes discussed above, however fully connected layers are still used. AlexNet (Krizhevsky, 2012) is a popular example of a DCNN using the dropout technique as tool to reduce the number of parameters and therefore the computational requirements during training.

### 2.2.4 Activation function

An activation function or also called squashing function is used to “squash” or “activate” a node in the network. It defines the output of a given node and defines its “usefulness” in the next layers. The activation function also helps limits the value of data into norms. An example would the sigmoid function where the value of a node after activation ranges between 0 and 1. More than one activation function can be used in a network; typically, the output layer makes use of a different activation

function better suited for its purpose. the most commonly used activation functions are:

- Sigmoid function
- Hyperbolic Tangent
- Rectified Linear Unit

#### 2.2.4.1 Sigmoid

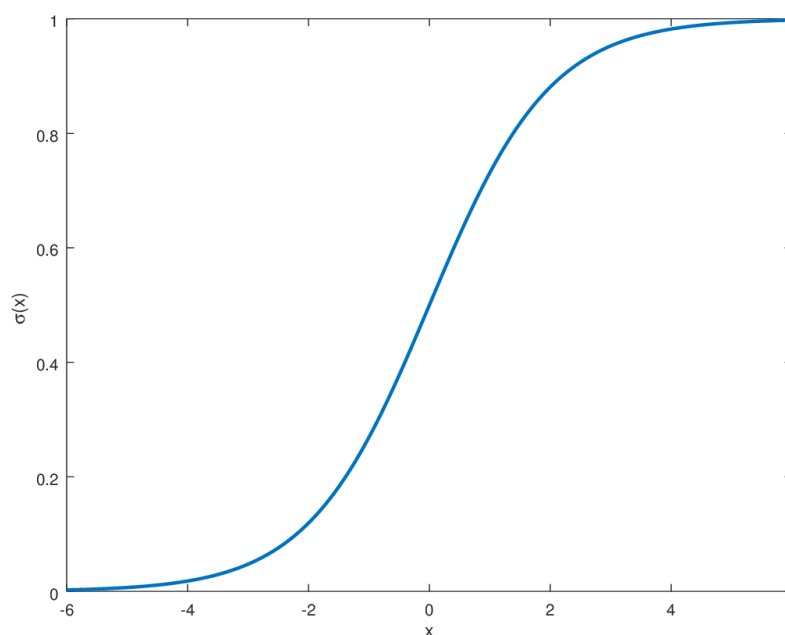
As mentioned before, after taking any real value from a node the sigmoid function outputs a value between 0 and 1, and is defined by equation 4. It is quite useful for assign a probability and boats effectiveness with binary classification.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (4)$$

Figure 11 shows the graphical representation of the sigmoid function.

Figure 11

*Sigmoid Function*



The sigmoid function is shaped like an S. there is a sharp increase around the inflexion point and its value seems to plateau at its extremities. The downside with sigmoid is that it can tend to reach a local minimum and therefore get stuck during training.

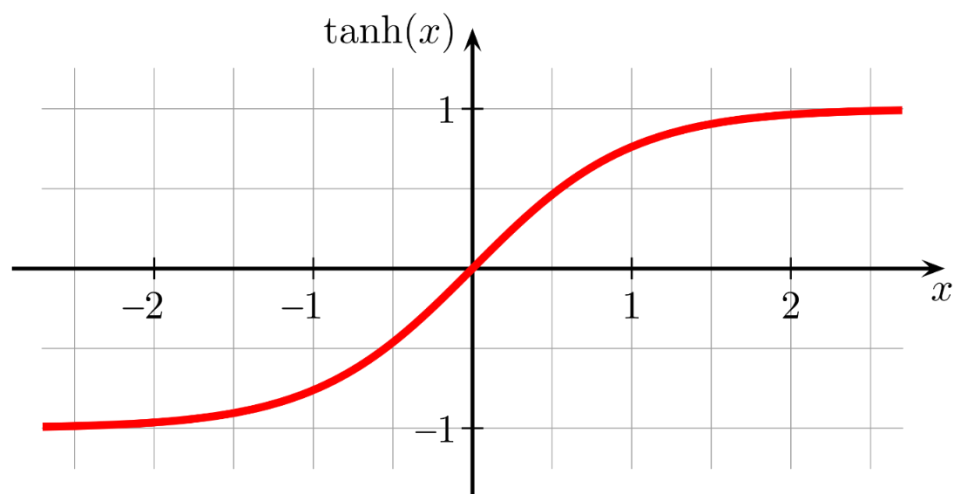
### 2.2.4.2 Hyperbolic Tangent

The Hyperbolic tangent or tanh also possess a S looking shape as sigmoid, however, it is on a bigger range. Tanh is defined by equation 5 and is represented graphically by figure 12.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (5)$$

Figure 12

*Hyperbolic Tangent(tanh) Function*



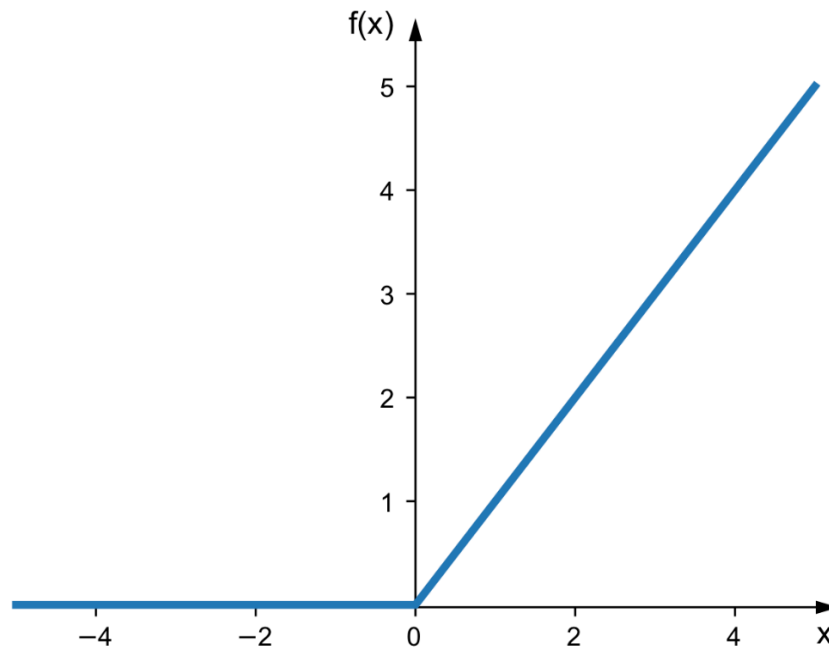
The bigger range allows for the values to be activated and seen as “strongly” negative or positive.

### 2.2.4.3 Rectified Linear Unit

ReLU is one of the most commonly used function in convolutional neural networks and deep learning. It is a simple function that presents advantages over the two previously mentioned activation functions, like its robustness to vanishing gradient where the model stops because it is unable to properly update the gradient. It is defined by equation 6 and is graphically represented by figure 13.

$$ReLU(x) = \max(0, x) \quad (6)$$

Figure 13

*Rectified Linear Unit (ReLU) Function*

The ReLU function sets all negative values to zero, which represents its strength but also its weakness since it creates “dead” units in the network.

### 2.3 ALEXNET

One the most important aspect of this project is AlexNet. It is a groundbreaking deep convolutional network that was entered in the ImageNet LSVRC-2010 contest that requires classifying 1.2 million images into one thousand different categories. The model produced a top-1 error rate of 37.5% and a top-5 error rate of 17%. Those results were outperforming their closest competitor by a substantial amount. When dealing with such large dataset, overfitting becomes a major issue, and as such AlexNet has ways of dealing with that problem. The resulting network has 8 layers where the removal of any particular one would reduce the performance despite consisting of 1% of the model’s total number of parameters. It is important to note that the model was trained on two GPU due to the large amount of data and the computational resource available at the time.

#### 2.3.1 Dataset

The dataset used in the model consist of images of different size that were fixed to a  $256 \times 256$  resolution. The number of images used ranges in the million and should be classified into 1000 classes representing different objects or animals.

### 2.3.2 Architecture

The model contains five convolutional layers and 3 fully connected layers as seen in figure 14. The network uses ReLU as an activation function due to its significantly shorter training time compared to others like tanh or sigmoid function. Since overfitting represents the main problem of the model, making the choice of using ReLU makes that much more sense.

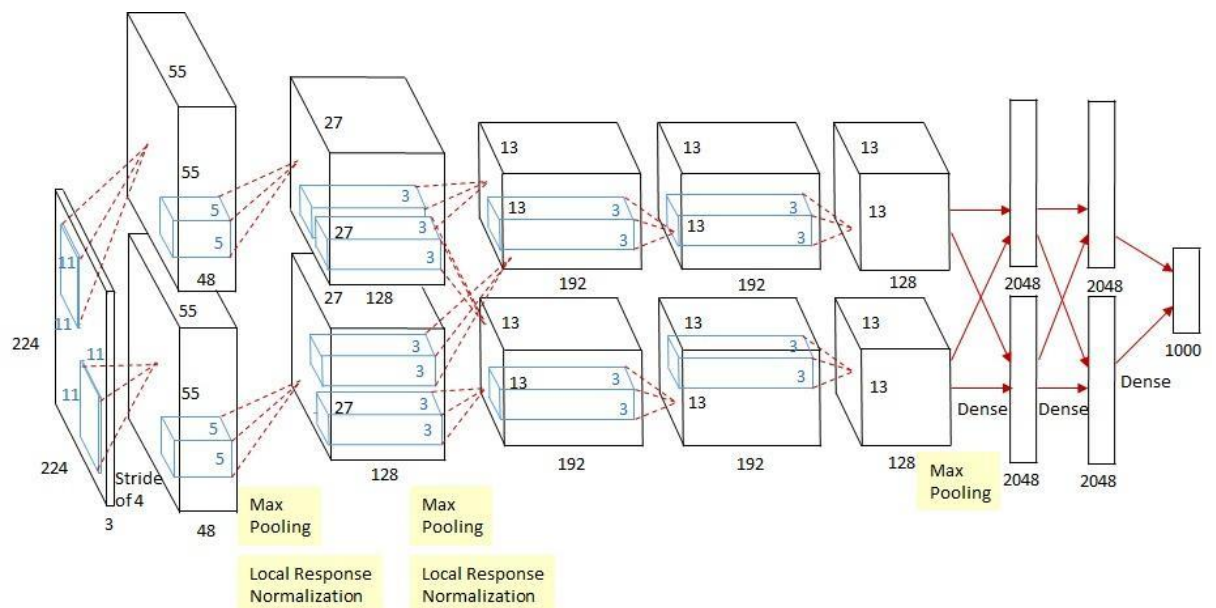
Pooling layers often return output from adjacent regions without overlapping. However, in this case, it was found that overlapping the pooling area reduces the error rate by 0.4% and generally reduces overfitting.

### 2.3.3 Overall Architecture

AlexNet consists of five convolutional layers and three fully connected layers. The last fully connected layer returns an output given to a 1000-way Softmax that will classify into the equally numbered labeled classes.

Figure 14

*AlexNet Architectural Map (Krizhevsky,2012)*



The layers following the input are divided into two GPUs operating in cross-parallelization. Half of the parameters are deployed on each GPU, however they still communicate in some layers. Max pooling layers are set in conjunction with local response normalization on the first two convolutional layers, before being set on the fifth one right before the fully connected layers.

### 2.3.4 Overfitting

Two ways are used to deal with overfitting this net, data augmentation and dropout.

Data augmentation, being an easy way of diminishing overfitting, consists of artificially enlarging a dataset while conserving its properties.

The “dropout” technique is another method used to deal with overfitting. It consists of dropping a certain number of hidden neurons in the fully connected layer, by setting their output to zero. In AlexNet’s case, the dropout probability is set to 50%. The neurons that are dropped, do not contribute in forwarding data in the subsequent layers nor are they involved in back-propagation. In principle, this forces the network to learn more solid features and stops any co-dependency between neurons since the presence of any particular becomes uncertain. The dropout technique in this way prevents substantial overfitting.

### Related Work

The model proposed by Hilmizen et al. (2020) in their study uses the concatenation of two transfer learning models using two datasets being CT scans and X-Ray images. The idea is to use the different characteristics the two modalities can provide and find the complementarity they share. They used different pretrained ImageNet networks such as, ResNet50, DenseNet121, Xception for the CT-scans images and VGG16, MobileNet, and InceptionV3 for the X-Ray images. The concatenated network models were shown to perform better than the individual networks by themselves. The ResNet50-VGG16 and DenseNet121-MobileNet had the same score on accuracy, sensitivity and specificity and were overall the top performing network with the results being 99.87%, 99.74%, 100% respectively.

Another model in the literature makes use of a three-stepped model that includes a feature extractor, followed an algorithm that sorts the said features and select the most important ones to be finally classified using an SVM classifier (Jin,2021). In the study, AlexNet is the network used for the feature extraction step. Using ReliefF the ten most important features extracted from the previous step are then sorted based on their importance. This results in reducing training time by removing unnecessary information. At the end, the SVM will handle the classification process the network. The proposed model showed a  $98.642 \pm 0.398\%$  overall accuracy which was the best performance when compared to five other models.

DeTraC is a process adopted by Abbas et al. (2020) in their study to improve on their previous deep convolutional neural network. DeTraC, in their work stands for Decompose, Transfer, and Compose. First, the pre-trained DCNN model of DeTraC is trained in order to extract local features of each element of the dataset. The decomposition layer is then applied, to provide simplification to the data structure. That process consists of partitioning each class in the input dataset in multiple sub-classes. After training, a class composition layer is applied to reassemble the sub-classes and to refine the last classification of the images. The results show significant improvement in accuracy, sensitivity, specificity in each of the DCNN models in which DeTraC was applied to.

FCOD is another network model proposed by Panahi et al. (2020), in an objective to reduce the computational cost and the issues with overfitting common with DCNNs having a significant number of parameters, reducing the detection time for COVID cases while offering high accuracy, a “depth-wise” separation of convolutional layers seemed to be solution that made sense. Instead of adopting a standard convolution filter, the model instead divides the input into channels where each of them will be met with a convolutional filter. There, a point-wise convolution will occur, where the output channels will be mixed.

The literature shows multiple other works, such as CVDNet from Ouchicha et al. (2020), where they propose a deep learning network based on two parallel paths with different kernel sizes aiming to study both local and global features of the images. ShuffleNet, built by Zhang (2017), is a DCNN which design is built for low computing devices such as mobile devices. Its architecture presents parameters that requires less complexity while maximizing the accuracy of the network. It serves as the basis for the model proposed by Ozyurt (2021) to provide automatic detection of COVID-19 cases. The model is designed by applying a feature extraction protocol using ReliefF and NCA in order to find and select the distinctive features. It also includes using transfer learning with ShuffleNet’s pretrained layers. Finally, the extracted features serve as input in the classifying layer of the network.

### **Summary**

This chapter showed important concept such as the major layers that builds a deep neural network. The convolutional layer filters the images for important features while the pooling layer reduces the size and the complexity of the input. A controlled

use of padding and stride allows for one to control the dimension of the input into the next layers. The choice of an activation function is an essential step into building the network as it they offer different advantages but different cons that are to be considered when setting the network.



## CHAPTER III: Methodology

### Overview

This section of the document explores the method used in this project. It documents the datasets used in the network, its architecture and the experiment set in order to reach the goals set at the beginning of the document.

### 3.1 Datasets

This section of the document will describe the different datasets used in the experiments conducted in the study. In order to test the flexibility of the model, five different experiments will be conducted. The datasets in this study includes two categories that our network is going to classify into. Each category includes a training and testing set.

The first dataset used in this document was obtained from Kaggle (Viradiya,2021) and includes thousands of radiography images showing x-ray scans of COVID afflicted patient and healthy ones. It includes patients of a wide range of age, gender, weight and body type picked at random. For the purpose of this study, the training sets include 3000 pictures to be examined while the testing or validation sets include 500 images each. In an effort to reduce the number of parameters and through trial and error, the number of pictures selected per case was kept at 3000.

The second dataset was retrieved from Mendelay Data (Alyasriy,2020). This set includes CT-scans showing patient afflicted with lung cancer and healthy patient. They were obtained by taking slices from DICOM format of pictures. They include slices from 110 patients with various age, gender and living conditions. The training set for the purpose of the study includes 511 images for malignant cancer cases and 366 healthy cases, whereas the testing set includes 50 images each. Those number were selected due the limited number of pictures, but during training they proved to be enough.

The third dataset in the study was retrieved form the Stoic Grand Challenge 2021 database. This set has CT-scans of patient with mild cases COVID or severe cases of COVID. The inclusion of this set of images is to observe how precise our network can be to differentiate between two types of actual cases of COVID. The set is made up of 3D views of 1000 cases of COVID. However, for the purposes of this network.

The “.mha” format of the set was converted into DICOM images before going through another conversion of the pictures into “.png” images. Similar to the second dataset, the slices that were selected show different sides and angles of the scans. Once again, the number of images was selected in an effort to reduce computing power while maximizing the accuracy. The training set is made out of 1190 images for the mild COVID case and 1234 images for the severe COVID case. The testing set is made out of 150 images for each case.

Finally, the fourth dataset used in the study from Kaggle (Lami,2021). This data involves patient suffering from radiation Pneumonitis as a result of their cancer treatment. The images show 1062 instances of CT-scan. They offer different angles and cuts much more similar to the previous dataset, which makes it a more interesting classification case for it.

Table 1 offers clarity in the numbers of images used for each set of data used in our training. It can be seen that x-rays have the most available image number. The Medelay data has the lowest number of images available and Pneumonitis set has a number of images that matches the Stoic 2021 set.

Table 1

*Training Samples Breakdown*

|                | Radiography Set |       | Medelay Data Set |         | Stoic 2021 Set |        | Pneumonitis Set |
|----------------|-----------------|-------|------------------|---------|----------------|--------|-----------------|
|                | X-Ray           |       | CT-Scans         |         | CT-Scans       |        | CT-Scans        |
|                | Healthy         | COVID | Malignant        | Healthy | Mild           | Severe | Pneumonitis     |
| Training       | 3000            | 3000  | 511              | 366     | 1190           | 1234   | 952             |
| Test           | 500             | 500   | 50               | 150     | 150            | 150    | 110             |
| Total Training | 6000            |       | 877              |         | 2424           |        |                 |
| Total Test     | 1000            |       | 100              |         | 300            |        |                 |
| Total          | 7000            |       | 977              |         | 2724           |        | 1062            |

Since the original dataset was made up of images with different and unfitting sizes from our network input layer, a resize of the pictures were made in order to obtain the required 227x227 size needed for training. Figure 15, figure 16 and figure 17 show examples of the images used in the network.

Figure 15

*Sample of the Radiography Training Dataset*

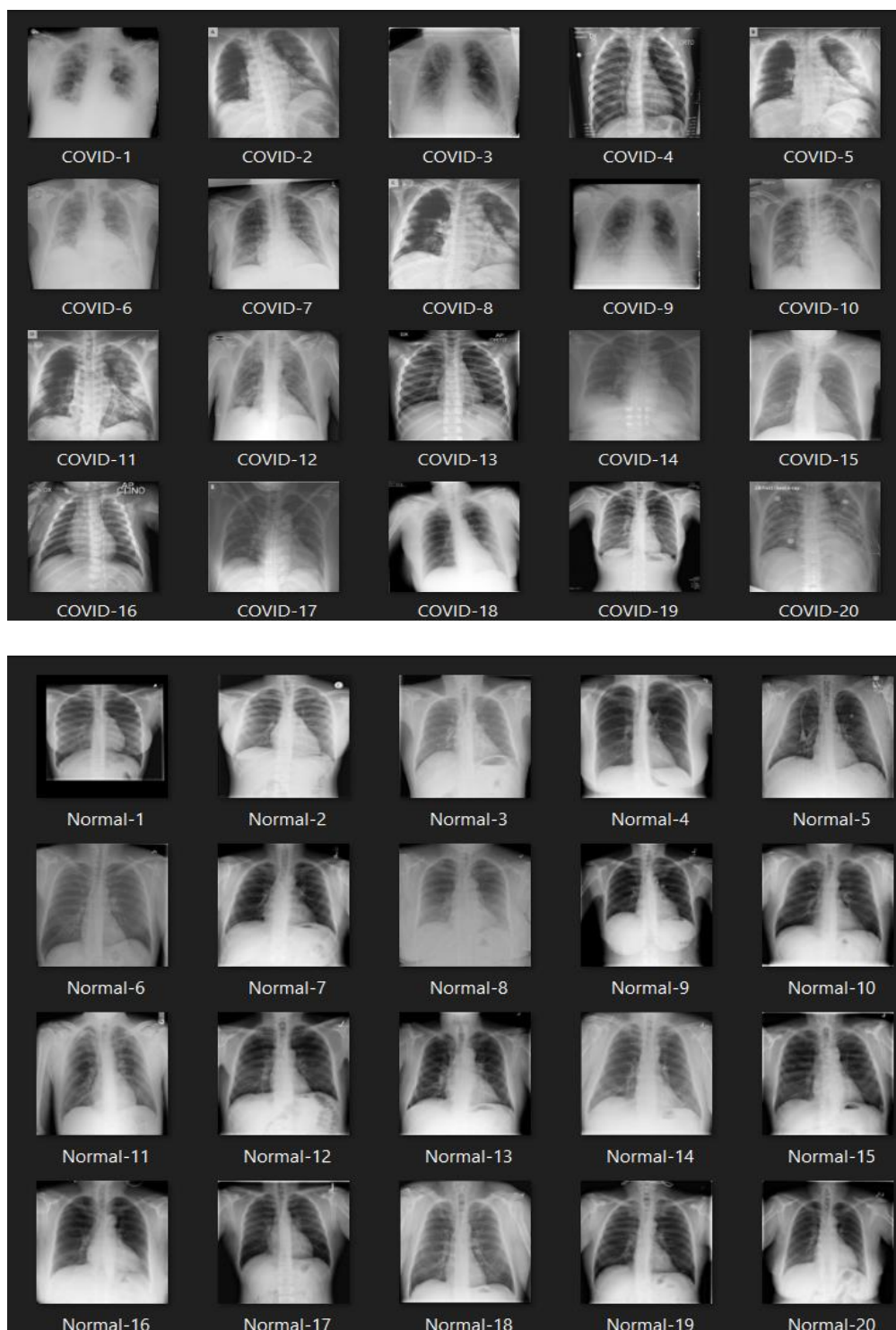


Figure 16

*Sample of the Lung Cancer CT-Scan Training Set*

Figure 17

*Sample of Severe and Mild COVID CT-Scan Training Set*

The similarity between the images is striking and would definitely require a medical license to properly observe and interpret.

### 3.2 Network architecture

The network was designed personally coded with AlexNet as a base. As a result, the presented network while very similar to AlexNet has some differences. This network is not using pretrained data to improve its accuracy nor is it using transfer learning. The initial weights of the network have therefore been randomly initialized using a GPU array. The random initialization is a better choice since we use ReLU as an activation function which is an effective tool against problems that might arise like a vanishing gradient. All the convolutional layers are followed by a ReLU layer for better activation and optimization of the parameters. The first two fully connected layers are also connected to ReLU layers but the third is connected to a softmax layer that will classify into the two concerned classes.

The Convolutional network was built with 25 layers designed as such:

1. The first layer has a number of  $227 \times 227 \times 3$  inputs, as our input images need to be resized to 227 by 227 pixels.
2. Convolutional layer with 96 filters of window size  $227 \times 227 \times 3$  with stride window  $4 \times 4$ .
3. ReLU layer
4. Cross Channel Normalization with 5 channels per element
5. Maximum pooling layer of window size  $3 \times 3$  with stride window  $2 \times 2$ .
6. Grouped Convolutional Layer with 2 groups of 128  $5 \times 5$  convolution windows with stride 1 and padding 2.
7. ReLU layer
8. Cross Channel Normalization with 5 channels per element
9. Maximum pooling layer of window size  $3 \times 3$  with stride window  $2 \times 2$ .
10. Convolutional layer with 384 filters of window size  $3 \times 3$  with stride 1 and padding 1.
11. ReLU layer
12. Grouped Convolutional Layer with 2 groups of 192  $3 \times 3$  convolution windows with stride 1 and padding 1.
13. ReLU layer
14. Grouped Convolutional Layer with 2 groups of 128  $3 \times 3$  convolution windows with stride 1 and padding 1.
15. ReLU layer

16. Maximum pooling layer of window size  $3 \times 3$  with stride window  $2 \times 2$ .
17. Fully connected layer with 4096 outputs
18. ReLU layer
19. 50 % Dropout Layer
20. Fully connected layer with 4096 outputs
21. ReLU layer
22. 50% Dropout Layer
23. Fully connected Layer with 2 outputs
24. SoftMax Layer
25. Classification Output Layer

The different activation map sizes and their learnable are detailed in table 2 in the next page of the document. A diagram showing the layout of the network can be seen in figure 18.

Table 2

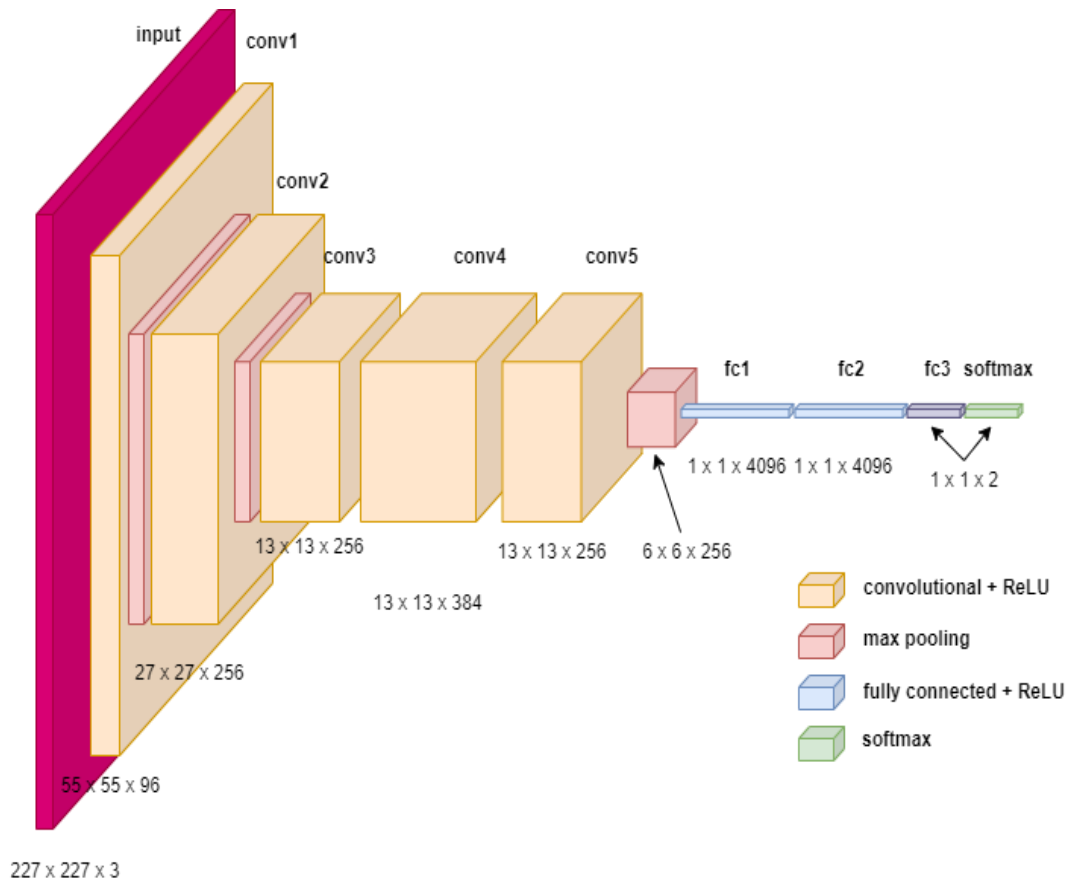
*Activation Map and Learnables Size per Layers*

|    | <b>Name</b> | <b>Type</b>                    | <b>Activation</b> | <b>Learnables</b>                       |
|----|-------------|--------------------------------|-------------------|---|
| 1  | imageinput  | Image Input                    | 227×227×3         |   |
| 2  | conv_1      | Convolution                    | 55×55×96          | Weights 11×11×3×96<br>Bias 1×1×96       |
| 3  | relu_1      | ReLU                           | 55×55×96          |   |
| 4  | crossnorm_1 | Cross<br>Channel Normalization | 55×55×96          |   |
| 5  | maxpool_1   | Max Pooling                    | 27×27×96          |   |
| 6  | conv_2      | Convolution                    | 27×27×256         | Weights 5×5×48×128×2<br>Bias 1×1×128×2  |
| 7  | relu_2      | ReLU                           | 27×27×256         |   |
| 8  | crossnorm_2 | Cross<br>Channel Normalization | 27×27×256         |   |
| 9  | maxpool_2   | Max Pooling                    | 13×13×256         |   |
| 10 | conv_3      | Convolution                    | 13×13×384         | Weights 3×3×256×384<br>Bias 1×1×384     |
| 11 | relu_3      | ReLU                           | 13×13×384         |   |
| 12 | conv_4      | Convolution                    | 13×13×384         | Weights 3×3×256×384<br>Bias 1×1×192×2   |
| 13 | relu_4      | ReLU                           | 13×13×384         |   |
| 14 | conv_5      | Convolution                    | 13×13×256         | Weights 3×3×192×128×2<br>Bias 1×1×128×2 |
| 15 | relu_5      | ReLU                           | 13×13×256         |   |
| 16 | maxpool_3   | Max Pooling                    | 6×6×256           |   |
| 17 | fc_1        | Fully Connected                | 1×1×4096          | Weights 4096×9216<br>Bias 4096×1        |
| 18 | relu_6      | ReLU                           | 1×1×4096          |   |
| 19 | dropout_1   | Dropout                        | 1×1×4096          |   |
| 20 | fc_2        | Fully Connected                | 1×1×4096          | Weights 4096×4096<br>Bias 4096×1        |
| 21 | relu_7      | ReLU                           | 1×1×4096          |   |
| 22 | dropout_2   | Dropout                        | 1×1×4096          |   |
| 23 | fc_3        | Fully Connected                | 1×1×2             | Weights 2×4096<br>Bias 2×1              |
| 24 | softmax     | Softmax                        | 1×1×2             |   |
| 25 | classoutput | Classification Output          | –                 |   |

The table can be obtained from MATLAB by inputting a line of command. The enormous number of parameters can be observed from the table.



Figure 18

*Network Diagram*

As we can see in figure 18, the size of the input images changes multiple times in the network. At the start, the image input of  $227 \times 227 \times 1$  is input in the first layer of the network. The first convolutional layer, possessing 96 filters, changes the input dimension to  $55 \times 55 \times 96$ . The ReLU and cross normalization layers throughout the network do not change the size of the activation maps, which is why the figure does not display those layers in an effort to keep clarity and avoid redundancy. The first maximum pooling layer will reduce the size to  $27 \times 27 \times 96$  before passing through the 256 filters present in the next convolutional layer resulting in a  $27 \times 27 \times 256$  output size. The next max pooling layer and the third convolutional layer in net will further reduce the size to  $13 \times 13 \times 384$ . The final convolutional layer in the network has 256 filters with  $3 \times 3$  windows, which results in the activation map to get a size of  $13 \times 13 \times 256$ . The next layer, called maxpool\_3, decrease the size further to obtain a  $6 \times 6 \times 256$  map. The first two fully connected layers of the net will keep the activation map at size  $1 \times 1 \times 4096$  before the final fully connected layer reduces it a final  $1 \times 1 \times 2$  dimension.

### 3.3 Training

This section of the study is going to describe the different experiments that were made using deep learning. The objective was to find how versatile, flexible and accurate the network can be.

The datasets include X-rays and CT-scans from different sources and complexity in classification. The first two experiment are straightforward and compare cases of patient afflicted with a disease or not. Meanwhile, the remaining experiments are about patient with either different diseases or different progression of the disease. The network was training multiple times or sessions in all those cases while trying to avoid major changes in parameters.

In each of the experiments, the network was trained using the Stochastic Gradient Descent with Momentum method on an NVIDIA Geo Force GTX 1650i GPU. Initially, the maximum number of epochs for training was to 10 with initial learn rate factor of 0.0001. While no gradient threshold was set, the L2 normalization method was set to compute it. Changes made during training, if any will be mentioned in the next sections.

The next section will document the layout of the training for each experiments done.

#### A. Experiment 1: X-Ray: Healthy vs Covid

The first experiment will involve X-Ray pictures comparing COVID patients and healthy patients. Table 3 show the number distribution of the images per case, where the positive class is “COVID”.

Table 3

*Images Numbers per Case (X-Ray: Healthy vs COVID)*

|              | Healthy | COVID |
|--------------|---------|-------|
| Training Set | 3000    | 3000  |
| Testing Set  | 500     | 500   |

Initially, 1000 images per case were chosen but, doing so lead to underwhelming results during training in the region of 60% - 70%. The training parameters did not require any changes but a gradual increase in training images through trial and error gave us the results that will be presented in the next section.

### Training parameters

Table 4 shows the parameters used to train the network using this dataset.

Table 4

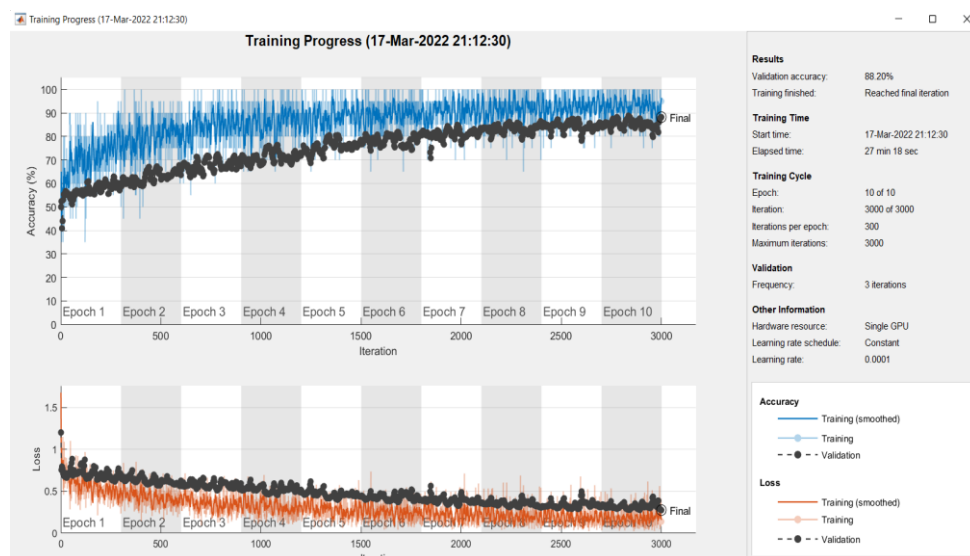
*Training parameters selected (X-Ray: Healthy vs COVID)*

|                             |   |
|-----------------------------|---|
| <b>Mini-Batch Size</b>      | 20  |
| <b>Max Epochs</b>           | 10  |
| <b>Initial Learn Rate</b>   | 0.0001                                    |
| <b>Training Method</b>      | Stochastic Gradient Descent with Momentum |
| <b>Normalization Method</b> | L2  |

Under these conditions the training progress that occurred is documented by figure 19.

Figure 19

*Training progress (X-Ray: Healthy vs COVID)*



We can observe a steady increase in the accuracy up until the end of the training session.

### **B. Experiment 2: CT-Scan: Healthy vs Malignant**

Experiment 2 concerns CT-Scans of patient with malignant lung cancer against healthy ones. Table 5 describes the number of images per cases.

Table 5

*Images Numbers per Case (CT-Scan: Healthy vs Malignant)*

|              | Healthy | Malignant |
|--------------|---------|-----------|
| Training Set | 511     | 366       |
| Testing Set  | 50      | 50        |

Almost the available data were using during this experiment and proved enough to produce really accurate results.

### **Training parameters**

The training parameters are reflected in table 6. They produced the best overall results after multiple training sessions.

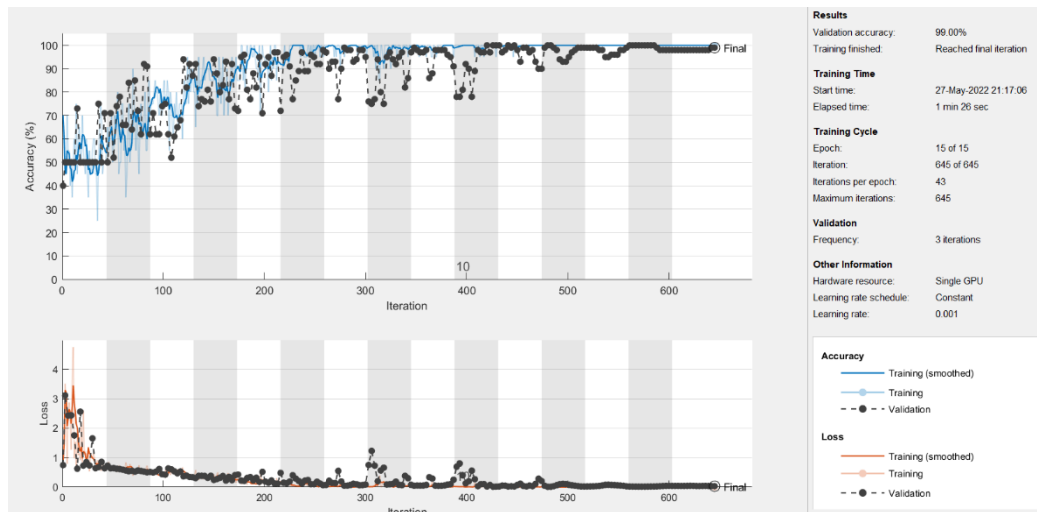
Table 6

*Training parameters selected (CT-Scan: Healthy vs Malignant)*

|                             |   |
|-----------------------------|---|
| <b>Mini-Batch Size</b>      | 20  |
| <b>Max Epochs</b>           | 15  |
| <b>Initial Learn Rate</b>   | 0.001                                     |
| <b>Training Method</b>      | Stochastic Gradient Descent with Momentum |
| <b>Normalization Method</b> | L2  |

In figure 20 we can observe the training progress made in one of the sessions.

Figure 20

*Training progress (CT-Scan: Healthy vs Malignant)*

There is a steady increase in accuracy until around 200+ iterations. From there the accuracy stabilizes in the high 90%.

**C. Experiment 3: CT-Scan: Severe vs Mild**

The numbers shown in table 7 have shown to produce the best results. Initially, the number of images were chosen in an effort to match the number of available Malignant cancer images but, they were unable to provide satisfying results. Therefore, an increase in number was needed and ultimately proved to be successful.

Table 7

*Images Numbers per Case (Severe vs Mild)*

|              | Severe | Mild |
|--------------|--------|------|
| Training Set | 1234   | 1190 |
| Testing Set  | 150    | 150  |

**Training parameters**

Table 8

*Training parameters selected (Severe vs Mild)*

|                        |    |
|------------------------|----|
| <b>Mini-Batch Size</b> | 20 |
|------------------------|----|

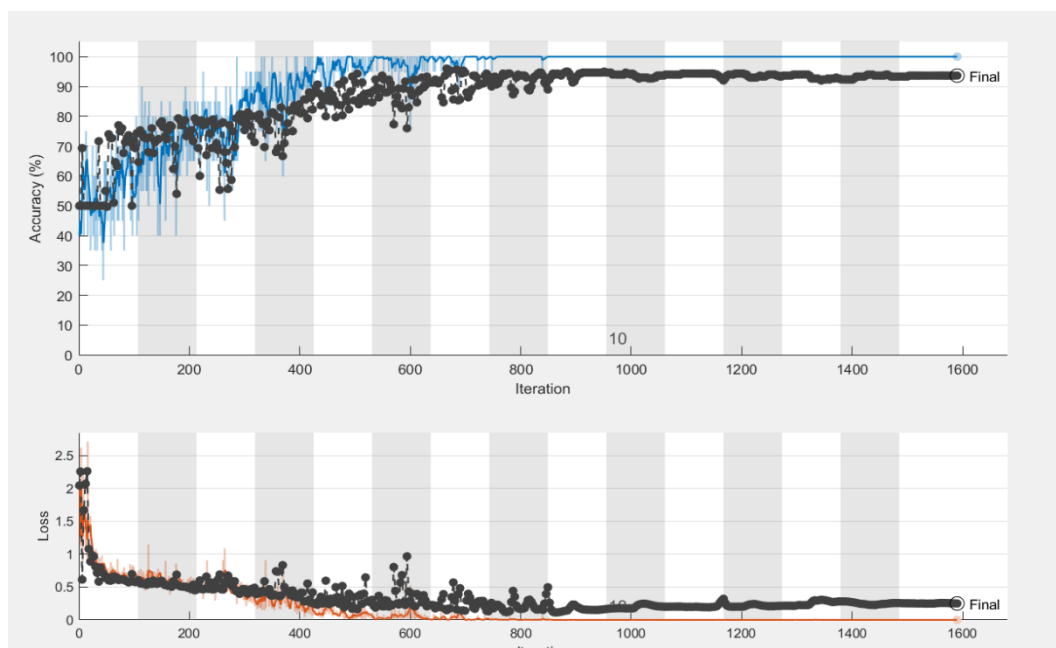
Table 8 (continued)

|                             |   |
|-----------------------------|---|
| <b>Max Epochs</b>           | 15  |
| <b>Initial Learn Rate</b>   | 0.001                                     |
| <b>Training Method</b>      | Stochastic Gradient Descent with Momentum |
| <b>Normalization Method</b> | L2  |

The training progress in one of the sessions of this experiment are provided by figure 21.

Figure 21

*Training progress (Severe vs Mild)*



After a steady increase, the network's accuracy starts to stabilize around the 500 iterations.

#### **D. Experiment 4: CT-Scan: Severe vs Pneumonitis**

The network has consistently produced high accuracy in any setting, parameters or image number throughout the multiple training processes. Therefore, the number of images from the previous datasets were kept the same.

Table 9

*Images Numbers per Case (Severe vs Pneumonitis)*

|              | Severe | Pneumonitis |
|--------------|--------|-------------|
| Training Set | 1234   | 952         |
| Testing Set  | 150    | 110         |

### **Training parameters**

The training parameters for the training session are shown in table 10.

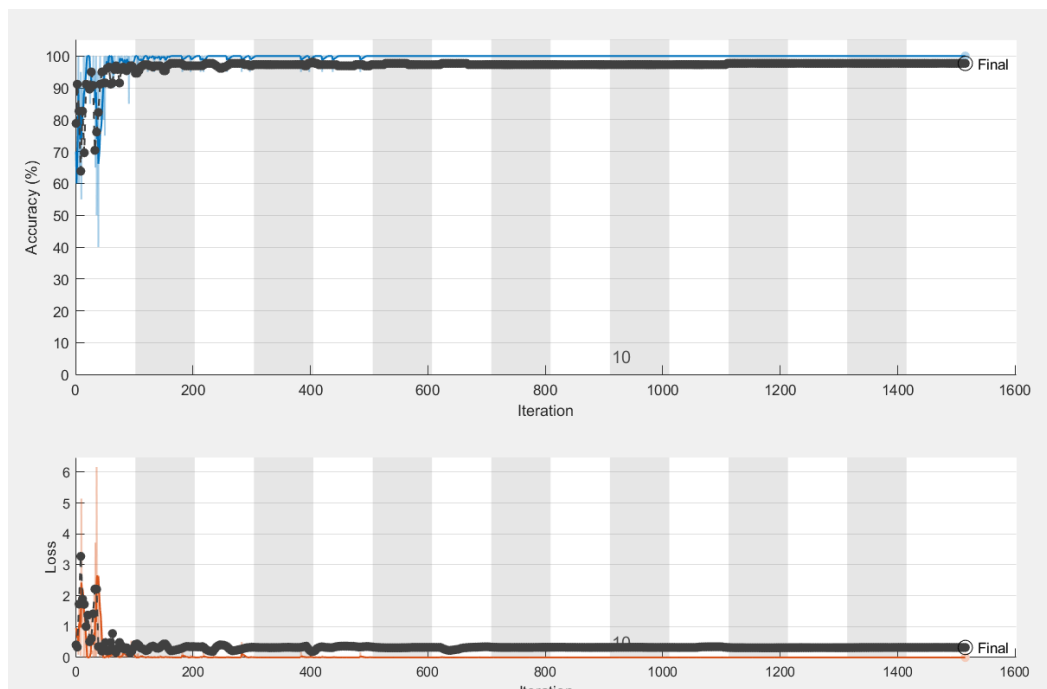
Table 10

*Training parameters selected (Severe vs Pneumonitis)*

|                             |   |
|-----------------------------|---|
| <b>Mini-Batch Size</b>      | 20  |
| <b>Max Epochs</b>           | 15  |
| <b>Initial Learn Rate</b>   | 0.001                                     |
| <b>Training Method</b>      | Stochastic Gradient Descent with Momentum |
| <b>Normalization Method</b> | L2  |

Figure 22 shows the training progress of the experiment.

Figure 22

*Training progress (Severe vs Pneumonitis)*

We can observe a sharp increase in accuracy that stabilize itself as soon as the 100 iterations. The network has quickly learnt and acquired really high accuracy from that point.

**E. Experiment 5: CT-Scan: Mild vs Pneumonitis**

This experiment operated almost identically in the same fashion as the previous one. The rationale behind the selected number of images is the same as in the previous experiment. The layout is shown in table 11.

Table 11

*Images Numbers per Case (Mild vs Pneumonitis)*

|              | Mild | Pneumonitis |
|--------------|------|-------------|
| Training Set | 1190 | 950         |
| Testing Set  | 150  | 110         |

**Training parameters**

The parameters are the same as previously mentioned in other experiments and shown in table 12.



Table 12

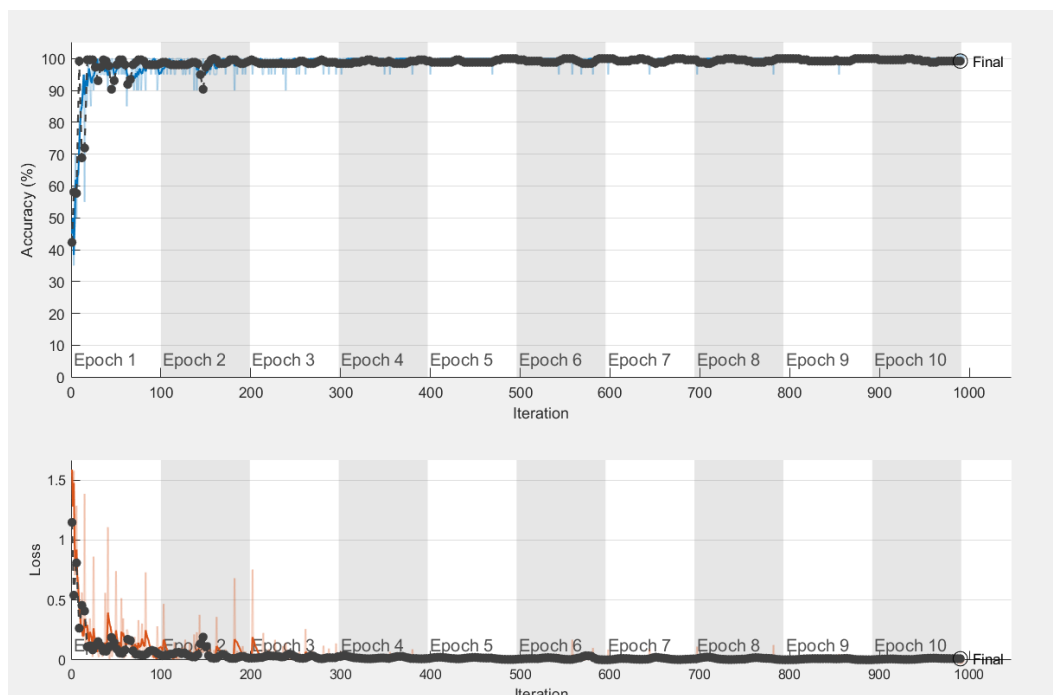
*Training parameters selected (Mild vs Pneumonitis)*

|                             |   |
|-----------------------------|---|
| <b>Mini-Batch Size</b>      | 20  |
| <b>Max Epochs</b>           | 15  |
| <b>Initial Learn Rate</b>   | 0.001                                     |
| <b>Training Method</b>      | Stochastic Gradient Descent with Momentum |
| <b>Normalization Method</b> | L2  |

These parameters gave the best results and the progress of the training in one of the sessions can be seen in figure 23.

Figure 23

*Training progress (Mild vs Pneumonitis)*



A rapid increase in accuracy before the 100-iteration mark followed by a steady continuation is observed. These results and their implication will be discussed in the next section but, it is important to remember that this is the training progress made during one session out of many.

## Summary

The datasets were obtained from different sources and provide images of several lung conditions. The dataset in the study includes X-Rays and CT images. The network presents an architecture that, while very similar to AlexNet, is not a result of transfer learning, but instead built from scratch with the random initial biases along with it. It has 25 layers, five of which are convolutional. Five experiments were set up where different comparison cases were tested as such:

- Experiment 1: Healthy vs COVID
- Experiment 2: Healthy vs Cancer
- Experiment 3: Severe COVID vs Mild COVID
- Experiment 4: Severe COVID vs Pneumonitis
- Experiment 5: Mild COVID vs Pneumonitis

## CHAPTER IV: Results and Findings

### Overview

In this part of the document, a review of the results obtained during the training sessions that occurred in each experiment will be provided.

### 4.1 Results and Findings

Firstly, a table showing the accuracy obtained during each of the training session will be presented. The table also shows the average accuracy and highlights the best results obtained. Afterwards, a second table will show more detailed results of the best training session. In addition to the accuracy, the other results that will be provided are:

- Sensitivity
- Specificity
- Area-Under-Curve

Equation 7 and equation 8 describe the sensitivity and the specificity.

$$Sensitivity = \frac{TP}{TP + FN} \quad (7)$$

where TP = True Positive

FN = False Negative

$$Specificity = \frac{TN}{TN + FP} \quad (8)$$

where TN = True Negative

FP = False Positive

Finally, the confusion matrix and the ROC curve will be provided as figures to give further knowledge about the network.

#### A. Experiment 1

Table 13 show the results of the training session performed under the conditions already provided earlier.

Table 13

*Accuracy per training session (Experiment 1)*

|                    | Accuracy     |
|--------------------|--------------|
| Training Session 1 | 0.852        |
| Training Session 2 | 0.882        |
| Training Session 3 | 0.845        |
| Training Session 4 | 0.85         |
| Training Session 5 | 0.858        |
| Average            | 0.8574       |
| Best               | <b>0.882</b> |

The average accuracy obtained during the training session is 85.74% and the best validation accuracy as seen in table 13 is shown to be **88.20%**.

More details about the best result are given in table 14.

Table 14

*Accuracy, Sensitivity, Specificity and AUC values (Experiment 1)*

| Accuracy | Sensitivity | Specificity | AUC    |
|----------|-------------|-------------|--------|
| 0.882    | 0.886       | 0.878       | 0.9525 |

This result is confirmed using the mean from the diagonal of the confusion matrix shown in figure 24. The positive class in the confusion matrix is ‘COVID’, while the negative one is ‘Normal’.

Figure 24

*Confusion matrix (Experiment 1)*

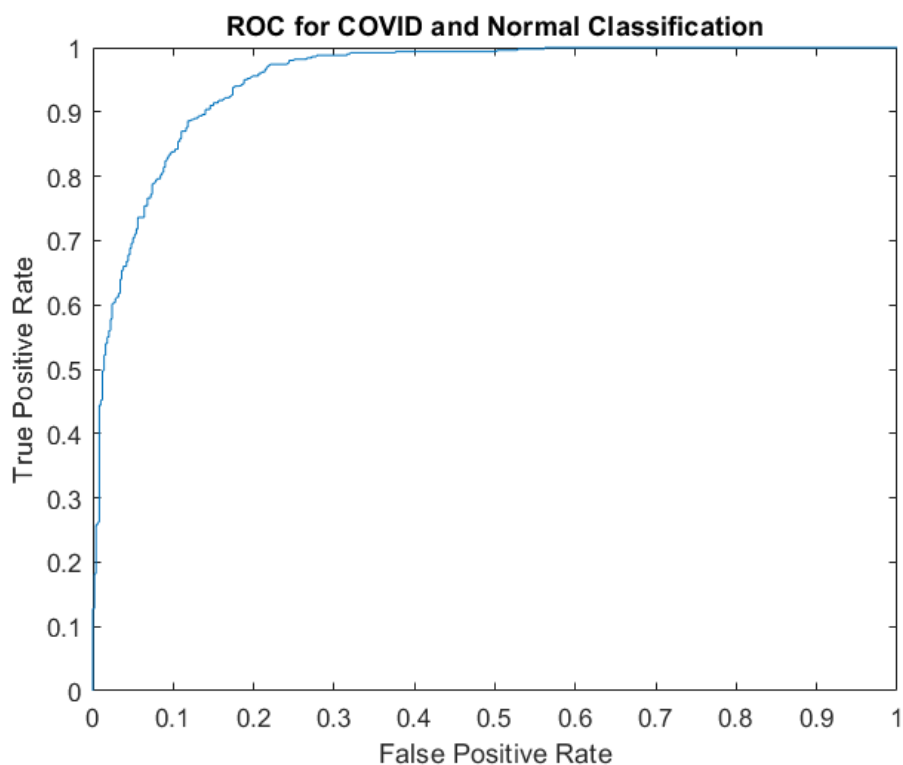
**Confusion Matrix**

|                     |                     |                              |                              |                              |
|---------------------|---------------------|------------------------------|------------------------------|------------------------------|
| <b>Output Class</b> | COVID               | <b>443</b><br>44.3%          | <b>61</b><br>6.1%            | <b>87.9%</b><br><b>12.1%</b> |
|                     | Normal              | <b>57</b><br>5.7%            | <b>439</b><br>43.9%          | <b>88.5%</b><br><b>11.5%</b> |
|                     |                     | <b>88.6%</b><br><b>11.4%</b> | <b>87.8%</b><br><b>12.2%</b> | <b>88.2%</b><br><b>11.8%</b> |
|                     | <b>Target Class</b> | COVID                        | Normal                       |                              |

It is interesting to note that the network performs better when finding true negatives showing an **87.8%** accuracy in that class, whereas it increases to an **88.6%** accuracy when trying to find true positives.

Figure 25

*ROC Curve (Experiment 1)*



The ROC has provided an Area Under the Curve value of **0.9525**.

### **B. Experiment 2**

The results obtained during the five training sessions are shown in table 15.

Table 15

*Accuracy per training session (Experiment 2)*

|                    | Accuracy |
|--------------------|----------|
| Training Session 1 | 0.99     |
| Training Session 2 | 0.98     |
| Training Session 3 | 0.98     |
| Training Session 4 | 0.99     |
| Training Session 5 | 0.99     |
| Average            | 0.986    |

Table 15 (Continued)

|      |             |
|------|-------------|
| Best | <b>0.99</b> |
|------|-------------|

The best result obtained is 99% accuracy, while the average accuracy is 98.6%. Even when perform at its worst the network is highly accurate in its predictions. Table 16 offers more detailed information about the best result.

Table 16

*Accuracy, Sensitivity, Specificity and AUC values (Experiment 2)*

| Accuracy | Sensitivity | Specificity | AUC |
|----------|-------------|-------------|-----|
| 0.99     | 0.98        | 1           | 1   |

The numbers on table 16 show that the network can hardly perform better. The AUC and the specificity are maximum meaning the network only on very rare occasion produce a false positive.

Figure 26

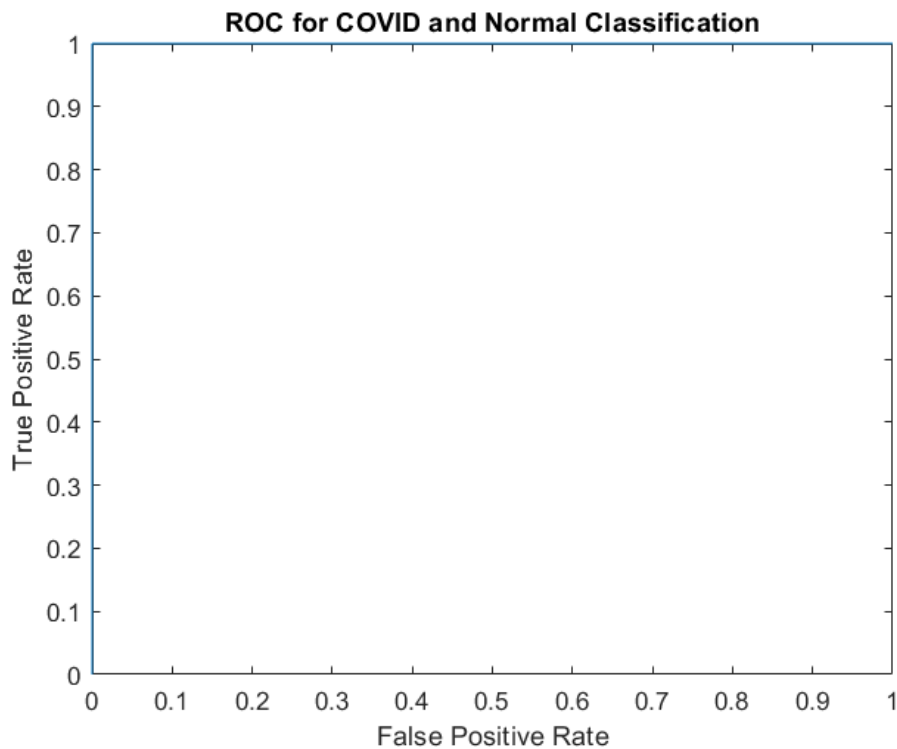
*Confusion Matrix (Experiment 2)*

|              |                 | Confusion Matrix |              |               |
|--------------|-----------------|------------------|--------------|---------------|
|              |                 | Malignant cases  | Normal cases |               |
| Output Class | Malignant cases | 49<br>49.0%      | 0<br>0.0%    | 100%<br>0.0%  |
|              | Normal cases    | 1<br>1.0%        | 50<br>50.0%  | 98.0%<br>2.0% |
|              |                 | 98.0%<br>2.0%    | 100%<br>0.0% | 99.0%<br>1.0% |
|              |                 | Malignant cases  | Normal cases |               |
|              |                 | Target Class     |              |               |

The confusion matrix illustrates how accurately the network performed when testing.

Figure 27

*ROC Curve (Experiment 2)*



The ROC produces an area of 1. The network performs as best as it can.

### C. Experiment 3

The training session conducted during experiment 3 lead to the results provided by table 17.

Table 17

*Accuracy per training session (Experiment 3)*

|                    | Accuracy |
|--------------------|----------|
| Training Session 1 | 93.7     |
| Training Session 2 | 93       |
| Training Session 3 | 93.3     |
| Training Session 4 | 92.7     |



Table 17 (Continued)

|                    |       |
|--------------------|-------|
| Training Session 5 | 91.7  |
| Average            | 92.88 |
| Best               | 93.7  |

The network achieved 92.88% accuracy on average which shows great performance. The best performance obtained 93.7% accuracy.

Table 18 provide additional details about the best outcomes during training.

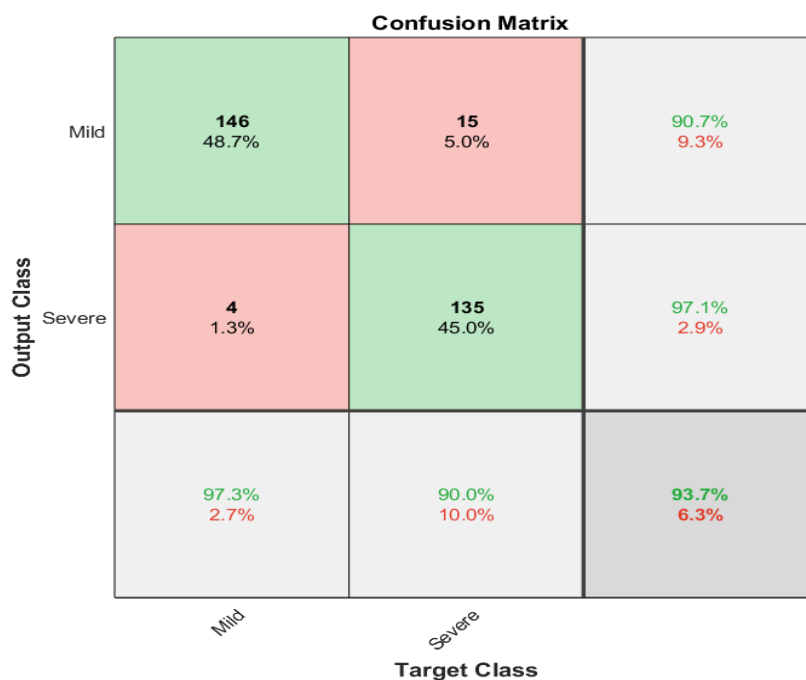
Table 18

*Accuracy, Sensitivity, Specificity and AUC values (Experiment 3)*

| Accuracy | Sensitivity | Specificity | AUC    |
|----------|-------------|-------------|--------|
| 0.937    | 0.947       | 0.92        | 0.9805 |

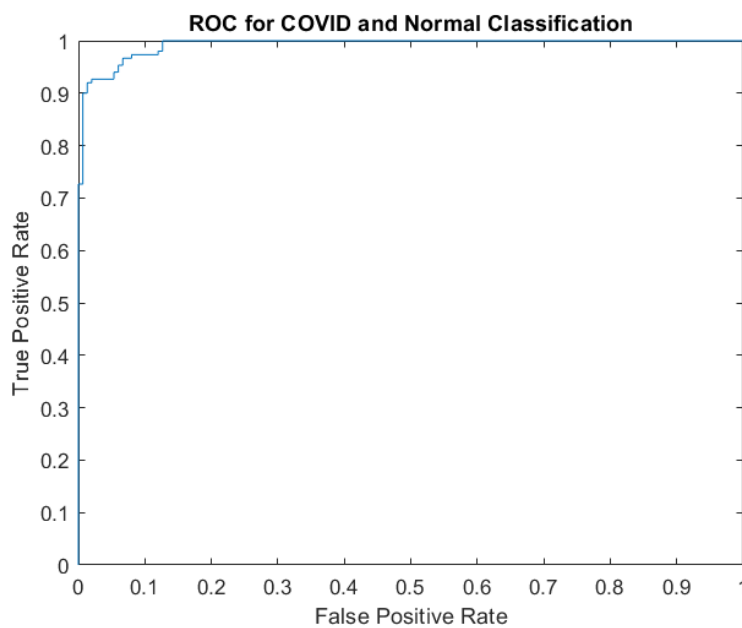
The Sensitivity and specificity respectively 94.7% and 92% show a low occurrence of false diagnostic.

Figure 28

*Confusion Matrix (Experiment 3)*

The confusion matrix highlights how many correct classifications were made in relation to the whole test dataset.

Figure 29

*Confusion Matrix (Experiment 3)*

From the ROC we can infer a high value of AUC which is confirmed to be 0.9805.

#### D. Experiment 4

After completing the different training sessions, table 19 provides the results obtained.

Table 19

*Accuracy per training session (Experiment 4)*

|                    | Accuracy |
|--------------------|----------|
| Training Session 1 | 97.69    |
| Training Session 2 | 96.54    |
| Training Session 3 | 96.92    |
| Training Session 4 | 97.31    |
| Training Session 5 | 96.92    |
| Average            | 97.07    |
| Best               | 97.69    |

The best result obtained was 97.07%, as shown in figure 30 and table 20, the network has really high sensitivity and overall performs really well.

The best results obtained during the training sessions are further expanded in table 20.

Table 20

*Accuracy, Sensitivity, Specificity and AUC values (Experiment 4)*

| Accuracy | Sensitivity | Specificity | AUC    |
|----------|-------------|-------------|--------|
| 0.9769   | 1           | 0.96        | 0.9861 |

When the best performance is obtained the network makes no mistake in determining which case is exposed in the image fed to it.

Figure 30

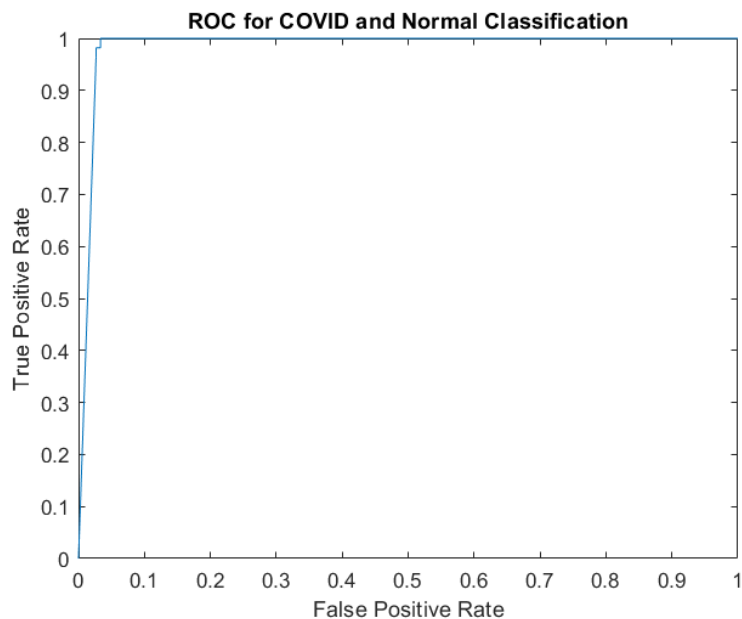
*Confusion Matrix (Experiment 4)*

**Confusion Matrix**

|                     |                     |                     |                     |               |
|---------------------|---------------------|---------------------|---------------------|---------------|
| <b>Output Class</b> | Pneumonitis         | <b>110</b><br>42.3% | <b>6</b><br>2.3%    | 94.8%<br>5.2% |
|                     | Severe              | <b>0</b><br>0.0%    | <b>144</b><br>55.4% | 100%<br>0.0%  |
|                     |                     | 100%<br>0.0%        | 96.0%<br>4.0%       | 97.7%<br>2.3% |
|                     | <b>Target Class</b> | Pneumonitis         | Severe              |               |

The confusion matrix shows how accurate the classifications were under the best performing training session.

Figure 31

*ROC Curve (Experiment 4)*

An AUC of 0.9861 provided by the ROC curve show that the network performs in excellent conditions.

### **E. Experiment 5**

The results through multiple cycle of training for experiment 5 are shown in table 21

Table 21

*Accuracy per training session (Experiment 5)*

|                    | Accuracy |
|--------------------|----------|
| Training Session 1 | 99.23    |
| Training Session 2 | 99.23    |
| Training Session 3 | 98.46    |
| Training Session 4 | 97.69    |
| Training Session 5 | 98.85    |
| Average            | 98.69    |
| Best               | 99.23    |

The average accuracy in experiment 5 (99.2%) is very similar to that of experiment 4. The reasons for such high accuracy are same as discussed in the previous experiment above. The best result obtained is 100% which shows again that the network is highly capable of classifying between two datasets from different sources.

Table 22

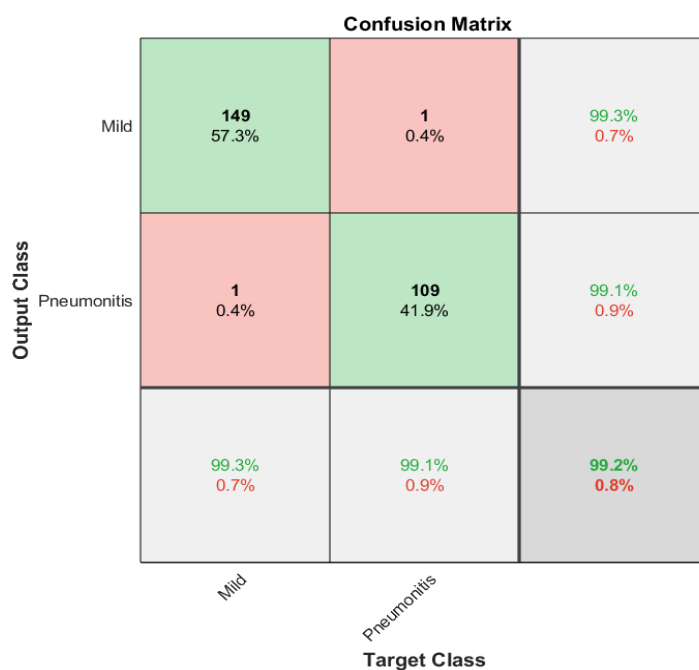
*Accuracy, Sensitivity, Specificity and AUC values (Experiment 5)*

| Accuracy | Sensitivity | Specificity | AUC    |
|----------|-------------|-------------|--------|
| 0.9923   | 0.993       | 0.991       | 0.9998 |

Table 22 shows that in its best conditions the model displays near perfect accuracy. Figure 32 shows the confusion matrix.

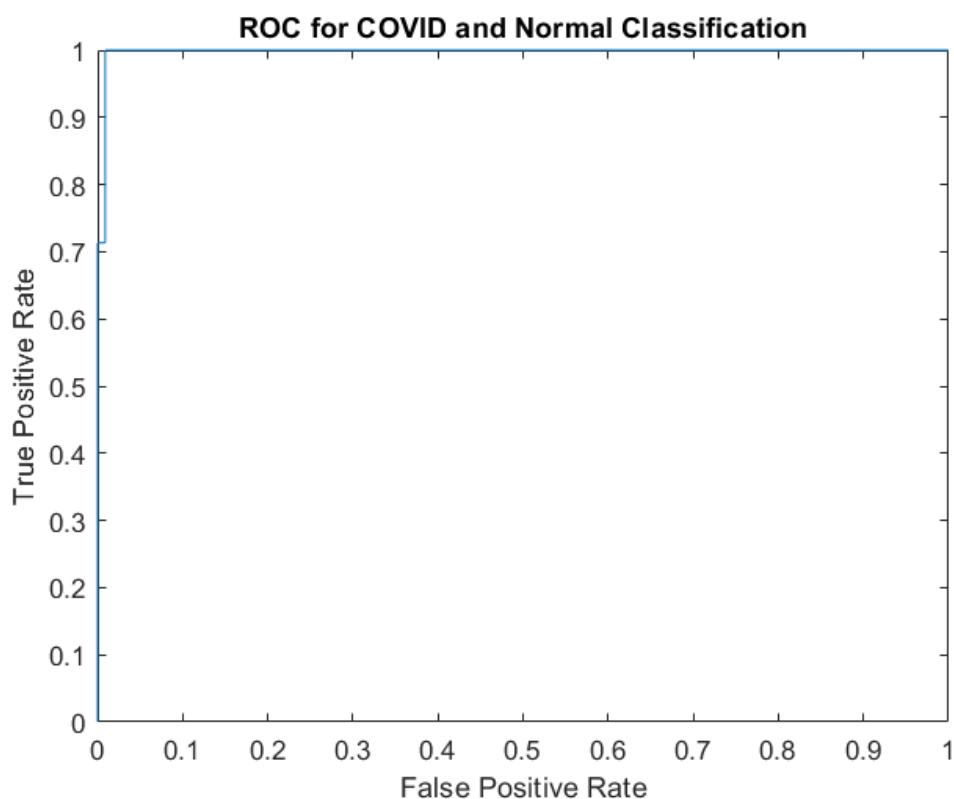
Figure 32

*Confusion Matrix (Experiment 5)*



The confusion matrix shows exactly how accurately the network was able to perform. It made only 2 mistakes, which shows excellent accuracy

Figure 33

*ROC Curve (Experiment 5)*

The ROC is similar to the one in experiment 4. The AUC is 0.9998 and can let us conclude that the network is performing really highly.

### Summary

Table 23 below shows the average accuracy and the best accuracy obtained from all five experiments.

Table 23

*Summary of the results from the experiments*

|              | <b>Average</b> | <b>Best</b> |
|--------------|----------------|-------------|
| Experiment 1 | 85.74%         | 88.20%      |
| Experiment 2 | 98.6%          | 99%         |
| Experiment 3 | 92.88%         | 93.7%       |
| Experiment 4 | 97.07%         | 97.69%      |
| Experiment 5 | 98.69%         | 99.23%      |

It can be seen from the table that even at its worst the network produces a 85.74% accuracy. It comes from the only experiment performed with X-rays. It reinforces the fact that CT are in fact a distinctly more accurate mean of detection.

## CHAPTER V: Discussion and Conclusion

### 5.1 Discussion

The five experiments were designed in order to answer the questions asked at the beginning of the document. Just as in any study involving deep learning, the goal is to obtain the highest accuracy level possible.

When reviewing the results and finding provided above we can draw multiple conclusions and most importantly we can provide answer to the aforementioned interrogations had at first.

Aggregating all the results obtained throughout all the sessions in all the experiments, the lowest reported accuracy value is 84.5%. This shows that even at when performing at its worst, our model still displays satisfying accuracy in its findings.

Experiment 1 provided an accuracy of 88.2%. As a reminder, experiment 1 presented a situation where the model was provided chest X-rays images showing either cases of COVID or healthy ones. The high accuracy obtained showed that the network is indeed able to distinguish between a case of COVID and a normal X-ray scan, thus answering one the questions provided.

During experiment 2, we find that the model is even better at treating CT-scans to distinguish between a healthy patient and a cancer case. Therefore, not only is the network able to recognize another disease than COVID when trained to but also operates just as well on CT-scan and X-Rays.

The situation provided in experiment 3 is most interesting as it raises the question of observe different state of the same disease, which surely would bring more complexity to the task. However, since the network provided an accuracy of 92.8% on average, we can conclude that the model is well suited to solve this kind of problem.

Experiment 4 and 5, tries to provide answers as to whether the model can distinguish between two different conditions, even with different progressions. The results obtained showed the network's high proficiency at accomplishing the given task.



It is also interesting to note that while network could not break the 90% accuracy barrier when classifying x-rays images, it did so in every experiment involving CT-scans. This leads to two conclusions that can be both true. First, the network is more proficient at dealing with CT-scans. Second, CT-scan provide more information on the different conditions they are used for detection. In any case, CT-scans provide more accurate diagnostics than X-rays.

## **5.2 Conclusion**

Today, fortunately, vaccines have been developed and are already in the distribution phase thanks to the work of various health professionals. However, there still is huge number of cases to be treated or even found, and since it is highly contagious disease, new cases are still bound to emerge. Therefore, a fast and accurate detecting tool is a must in this fight against the pandemic. The goal of this study is to provide such tool that helps discovering and acting quick against the disease. Using AlexNet's design as the backbone of the project, we adapted it to our study case and managed to obtain good results.

Earlier in the study, some questions, which served as goals, were asked. After designing the deep convolutional neural network, some experiment or situation were set up in order to answer those questions. The findings show that the net is indeed capable of classifying normal cases from COVID. The net is indeed capable of classifying normal cases from cancer cases. It is indeed capable of classifying COVID cases from cancer cases. It is also capable of distinguishing different progression of COVID. The presented model is able to realize those operations using chest x-rays and CT-scan.

Despite the quality of our non-pre-trained model, the results and findings provided leave room for improvement. An accuracy of 88.2% is still a good result but, can be improved. Perhaps, with the use of transfer learning or by pre-training or by setting more appropriate initial biases higher results can be obtained. The data sources were limitations in study that could be overcome by finding more uniform sources. One could also collect their set of images to suit their model better. The fact that images were altered in order fit the model or in an effort to reduce computing cost is also to be considered, as under more optimal condition the model would undoubtedly produce better and more consistent results.

## REFERENCES

- “Stoic Grand Challenge” [Online]. Available: <https://stoic2021.grand-challenge.org/>
- “World Health Organization”. [Online]. Available: <https://www.who.int/emergencies/diseases/novel-coronavirus-2019/interactive-timeline?gclid=EAIaIQobChMIxuaF1eqW8gIV1-F3Ch3OuwKhEAAAYASAAEgI6WPD BwE& cf chl captcha tk =pmd 0985337f3255677663f83f9a092187756d179d0f-1628061763-0-gqNtZGzNA02jcnBszQb6#event-16>
- Abaas, A., Abdelsamea, M. M., Gaber, M. M., “Classification of COVID-19 in Chest X-Ray images using DeTraC deep convolutional neural network”, Applied Intelligence (2021) 51:854-864, September 2020.
- Pahani, A. H., Rafiei, A., Rezaee, A., “FCOD: Fast COVID-19 Detector based on Deep Learning Techniques”, Informatics in Medicine Unlocked 22 (2021)100506, December 2020. [Online]. Available: <https://doi.org/10.1016/j.imu.2020.100506>
- Krizhevsky, A., Sutskever, I., Hinton, G. E., “ImageNet Classification with Deep Convolutional Neural Networks”, Advances in Neural Information Processing Systems 25 (NIPS 2012), 2012. [Online]. Available: <https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html>
- Olimov, B., Karshiev, S., Jang, E., Din, S., Paul, A., Kim, J., “Weight initialization based-rectified linear unit activation function to improve the performance of a convolutional neural network model”, Concurrency and Computation: Practice and Experience, Vol. 33, Issue 22, December 2020. [Online]. Available: <https://doi.org/10.1002/cpe.6143>
- Prijono, B., “Student Notes: Convolutional Neural Networks (CNN) Introduction”, Indoml. March 2018. [Online]. Available: <https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/>
- Ouchicha, C., Ammor, O., Meknassi, M., “CVDNet: A novel deep learning architecture for detection of corona virus (COVID-19) from chest x-ray images”,

Chaos, Solitons and Fractals 140 (2020) 110245, September 2020.[Online].

Available: <https://doi.org/10.1016/j.chaos.2020.110245>

Ozyurt, F., “Automatic Detection of COVID-19 Disease by Using Transfer Learning of Light Weight Deep Learning Model”, *Traitement du Signal*, Vol. 38, No. 1, February 2021, pp. 147-153.

Hamdalla, F. K., The IQ-OTH/NCCD lung cancer dataset, Kaggle. [Online].

Available: <https://www.kaggle.com/datasets/hamdallak/the-iqothnccd-lung-cancer-dataset/discussion>

Brownlee, J., “Weight Initialization for Deep Learning Neural Networks”, *Machine Learning Mastery*, February 2018. [Online]. Available:

<https://machinelearningmastery.com/weight-initialization-for-deep-learning-neural-networks/>

Koushik, J., “Understanding Convolutional Neural Networks”, *ArXiv e-prints, Other Statistics (stat.OT)*, May 2016. [Online]. Available:

<https://arxiv.org/abs/1605.09081v1>

Nayak, J., Nail, B., Dinesh, P., Vakula, K., Byomakesha, P., Pelusi, D., “Significance of deep learning for COVID-19: state-of-the-art review”, *Research on Biomedical Engineering* (2021), March 2021. [Online]. Available:

<https://doi.org/10.1007/s42600-021-00135-6>

Wu, J., “Introduction to Convolutional Neural Networks”, *National Key Lab for Novel Software Technology*, May 2017. [Online]. Available:

<https://cs.nju.edu.cn/wujx/paper/CNN.pdf>

Yi, J., Kang, H. K., Kwon, J-H., Kim, K-S., Park, M. H., Seong, Kim, Y. K. D. W., Ahn, B., Ha, K., Lee, J., Hah, Z., Bang, W-C., “Technology trends and applications of deep learning in ultrasonography: image quality enhancement, diagnostic support, and improving workflow efficiency”, *Ultrasonography* 2021, 40(1): 7-22. [Online].

Available: <https://doi.org/10.14366/usg.20102>

Dev, K., Khowaja, S. A., Bist, A. S., Saini, V., Bhatia, S., “Triage of potential COVID-19 patients from chest X-ray images using hierarchical convolutional networks”, *Neural Computing and Applications*, 2021. [Online]. Available:

<https://link.springer.com/article/10.1007/s00521-020-05641-9#citeas>

O'Shea, K., Nash, R., "An Introduction to Convolutional Neural Networks", ArXiv e-prints, Neural and Evolutionary Computing (cs.NE), December 2015. [Online].

Available: <https://arxiv.org/abs/1511.08458>

Guruchan, M K., "Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network", UpGrad, December 2020. [Online]. Available:

<https://www.upgrad.com/blog/basic-cnn-architecture/>

Hilmizen, N., Bustamam, A., Sarwinda, D., "The multimodal Deep Learning for Diagnosing COVID-19 Pneumonia form Chest CT-Scan and X-Ray Images", 2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), 2020.

Viradiya, P., "Kaggle". [Online]. Available:

<https://www.kaggle.com/datasets/preetviradiya/covid19-radiography-dataset>

Khartik, R., Menaka, R., Hariharan, M., "Learning distinctive filters for COVID-19 detection from chest X-ray using shuffled residual CNN", Applied Soft Computing Journal 99 (2021) 106744, 2021. [Online]. Available:

<https://www.sciencedirect.com/science/article/pii/S1568494620306827>

Albawi, S., Mohammed, T. A, Al-Zawi, S., "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6. [Online]. Available: <https://ieeexplore.ieee.org/document/8308186>

Yadav, S., "Weight Initialization Techniques in Neural Networks", Towards Data Science, November 2018. [Online]. Available:

<https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78>

Wang, S.-H., Nayak, D. R., Guttery, D. S., Zhang, X., Zhang, Y.-D., "COVID-19 classification by CCSHNet with Deep fusion using transfer learning and discriminant correlation analysis", Information fusion, Vol 68, February 2021, pp. 131-148.

[Online]. Available:

<https://www.sciencedirect.com/science/article/pii/S1568494620306827>

Jin, W., Dong, S., Dong, C., Ye, X., "Hybrid ensemble model for differential diagnosis between COVID-19 and common viral pneumonia by chest X-Ray

radiograph”, Computer in Biology and Medicine 131 (2021) 104252, February 2021. [Online]. Available: <https://doi.org/10.1016/j.combiomed.2021.104252>

Zhang, X., Zhou, X., Lin, M., Sun, J., “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices”, Computer Vision and Pattern Recognition (cs.CV), December 2017. [Online]. Available: <https://arxiv.org/abs/1707.01083>

## APPENDICES

### Appendix A: Binary Classification Code

#### Loading Data

```
%TRAINING DATA

categories = {'COVID', 'Normal'};

rootfolder = 'trainData';

imds = imageDatastore(fullfile(rootfolder, categories), ...
    'labelSource', 'foldernames');

%TEST DATA

rootfolder = 'testData';

imds_test = imageDatastore(fullfile(rootfolder, categories), ...
    'labelSource', 'foldernames');
```

#### Network Architecture

```
%% NETWORK ARCHITECTURE

conv1 = convolution2dLayer(11, 96, 'Stride', 4);

conv1.Weights = gpuArray(single(randn([11 11 3 96])*0.01));

Layers = [
    imageInputLayer([227 227 3])
    conv1;
    reluLayer();
    crossChannelNormalizationLayer(5);
    maxPooling2dLayer(3, 'Stride', 2);
```

```

groupedConvolution2dLayer(5,128,2,'Stride',1,'Padding',2);
reluLayer();
crossChannelNormalizationLayer(5);
maxPooling2dLayer(3,'Stride',2);
convolution2dLayer(3,384,'Padding',1,'Stride',1);
reluLayer();
groupedConvolution2dLayer(3,192,2,'Stride',1,'Padding',1);
reluLayer();
groupedConvolution2dLayer(3,128,2,'Stride',1,'Padding',1);
reluLayer();
maxPooling2dLayer(3,'Stride',2);
fullyConnectedLayer(4096,'BiasLearnRateFactor',20);
reluLayer();
dropoutLayer(0.5)
fullyConnectedLayer(4096,'BiasLearnRateFactor',20);
reluLayer();
dropoutLayer(0.5);
fullyConnectedLayer(2,'BiasLearnRateFactor',20)
softmaxLayer;
classificationLayer()]

```

## Training

```

% TRAINING
opts = trainingOptions('sgdm', ...
    'MiniBatchSize',20, ...
    'MaxEpochs',15, ...
    'InitialLearnRate',1e-3, ...

```

```
'ValidationData',imds_test, ...
'ValidationFrequency',3, ...
'ValidationPatience',Inf, ...
'Verbose',true, ...
'Plots','training-progress');

[lexnet,info] = trainNetwork(imds,Layers,opts);

%% Testing Data

labels = classify(lexnet, imds_test);

ii = randi(35);
im = imread(imds_test.Files{ii});
imshow(im);

if labels(ii) == imds_test.Labels(ii)
    colorText = 'g';
else
    colorText = 'r';
end

title(char(labels(ii)), 'Color', colorText);

%Computing Accuracy

YPred = classify(lexnet,imds_test);
YTest = imds_test.Labels;

accuracy = sum(YPred == YTest)/numel(YTest);
```



## Test Network

```
confMat = confusionmat(imds_test.Labels, labels);
confMat = confMat./sum(confMat,2);
mean(diag(confMat))
figure,plotconfusion(imds_test.Labels, labels)
```

## Plot ROC

```
[predVal,scores] = classify(lexnet,imds_test);
[X,Y,T,AUC,OPTCOOPT,SUBY,SUBYNAMES] =
perfcurve(imds_test.Labels,scores(:,1),'COVID');
figure,plot(X,Y)
xlabel('False Positive Rate')
ylabel('True Positive Rate')
title('ROC for COVID and Normal Classification')
```

## Appendix B: Code for Resizing Images in the Dataset

Read Folder

```
inputFolder =
'C:\Users\DELL\Documents\MATLAB\DeepLearning\Medical
Radiography Images\COVID-19_Radiography_Dataset\COVID';
outputFolder =
'C:\Users\DELL\Documents\MATLAB\DeepLearning\Medical
Radiography Images\Resized\3ch\trainData\COVID';
filenames = dir(fullfile(inputFolder, '*.png'));
numimages = numel(filenames);
```

Resize and save

```
for n = 1:numimages
    f = fullfile(inputFolder,filenames(n).name);
    fimg = imread(f);
    fimgr = cat(3,fimg,fimg,fimg);
    reimg = imresize(fimgr,[227 227]);
    fullOutputFileName =
fullfile(outputFolder,filenames(n).name);
    imwrite(reimg,fullOutputFileName);
end
```