Detection of COVID-19 using Deep Learning Techniques and Extraction of the Infected Region using Lung Image Segmentation

Divyang Kapadia

A Thesis
in
The Concordia Institute
for
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Applied Science (Electrical and Computer Engineering) at
Concordia University
Montréal, Québec, Canada

March 2023

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By:	**Divyang Kapadia**
Entitled:	**Detection of COVID-19 using Deep Learning Techniques and Extraction of the Infected Region using Lung Image Segmentation**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Rabin Raut	Chair
Dr. Chun-Yi Su (MIAE)	External Examiner
Dr. Wei-Ping Zhu	Examiner
Dr. M. N. S. Swamy	Thesis Supervisor

Approved by
Dr. Abdel R. Sebak, Associate Chair, Graduate Studies, ECE

April 3, 2023
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Detection of COVID-19 using Deep Learning Techniques and Extraction of the Infected Region using Lung Image Segmentation

Divyang Kapadia

The world at present is still suffering from corona virus disease of 2019 (COVID-19) pandemic. COVID-19 is caused by severe acute respiratory syndrome corona virus 2 (SARS-CoV-2), which damages human lungs and results in pneumonia. Pneumonia is a disease in which the lungs become filled with fluid and inflame, leading to breathing difficulties. Sometimes, breathing problems become very severe, requiring proper treatment, including oxygen or a ventilator.

Though RT-PCR tests are the commonly used method to detect COVID-19 virus, radiological tests are often used by doctors to diagnose the disease based on severity level and risk factors. This thesis concentrates on two major issues, automatic detection of the COVID-19 infection using deep learning techniques and the determination of the severity level of the infection to help reduce the manual tasks and burden of the radiologist.

In the first part, different deep neural network architectures including convolutional neural network-based ResNet50, DarkNet19, GoogLeNet, and VGG16 methods along with a self-attention based vision transformer (ViT) approach called COViT-CT are implemented to detect COVID-19 computed tomography (CT)-Scan images. The performance of the various architectures are compared using various metrics, such as accuracy, precision, recall (sensitivity), specificity, F1-score and AUC, as well the confusion matrix, and the best architecture with the highest accuracy, which is COViT-CT, is selected for automatic COVID-19 detection.

In the second part, if the CT-Scan image is COVID-19 positive during the first part, then the image segmentation method is used to extract the COVID-19 infected region from the Lung CT-Scan images. The infected region is useful in determining the severity level of the COVID-19 infection, which helps in the diagnosis of the disease.

All the experiments are performed using the SARS-CoV-2 CT-Scan dataset. It is shown that the self-attention based COViT-CT method provides the best performance on the test sets of the above-mentioned dataset.

Acknowledgments

It has been a great privilege to work with Dr. M.N.S. Swamy, my research supervisor, who allowed me to pursue my research in the field of deep learning and image processing. I can never thank him enough for his guidance and support, valuable critiques, constant assistance, and enthusiastic encouragement during my research. It has been a great learning experience, and I am thrilled to have this opportunity to work under his direction.

I want to express my deep gratitude to Dr. Hassan Rivaz for the course Medical Image Processing, Dr. Habib Benali for Biological signal processing, Dr. Dongyu Qiu for Probability and Stochastic Processes, and Dr. Wei-Ping Zhu for the courses on Digital Signal Processing and 2-D Image Processing. Their knowledge and experience help me a lot during my research.

I am thankful to my parents, Chetnaben and Nareshkumar, and my younger brother Vinus for their exceptional support and encouragement during my research work. I am deeply indebted to my wife, Hinal, who sacrificed her family time and allowed me to work on my thesis. I would also like to thank Mr. Kalpeshkumar Ranipa and all my friends, lab mates and colleagues for editing help, late-night feedback sessions, and moral support. Not only this thesis but every successful step of my life was possible due to constant support from all my family members and friends.

Finally, I thank God for his blessings.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ANFIS	Adaptive Neuro-Fuzzy Inference Systems
ANN	Artificial Neural Network
AXIRs	Automated X-ray Imaging Radiography Systems
CNN	Convolutional Neural Network
COVID-19	Corona Virus Disease of 2019
CSA	Cuckoo Search Algorithm
CT	Computed Tomography
CV	Chan-Vese
DeCoVNet	3D Deep Convolutional Neural Network
ELM	Extreme Learning Machine
FCN	Fully Convolutional Network
FN	False Negatives
FP	False Positives
GELU	Gaussian Error Linear Unit
GKD	Gaussian Kernel Dropout
HOG	Histogram of Oriented Gradients
KISEG	Key and Intermediate Frame of Segmentation
LBP	Local Binary Pattern

LS	Level-Set
LSTM	Long Short-term Memory
MADE	Memetic Adaptive Differential Evolution
MLP	Multilayer Perceptron
MODE	Multi-Objective Differential Evolution
MRF	Markov Random Field
MSA	Multi-Headed Self Attention
NLP	Natural Language Processing
ReLU	REctified Linear Unit
ResNet	Residual Neural Network
ROC	Receiver Operating Characteristic
ROI	Region of Interest
SARS-CoV-2	Severe Acute Respiratory Syndrome Corona Virus 2
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machines
TB	Tuberculosis
TN	True Negatives
TP	True Positives
VGG	Visual Geometry Group
ViT	Vision Transformer
WHO	World Health Organization
YOLO	You Only Look Once

List of Symbols

C	Number of channels of input image
E_{pos}	Positional embedding vectors
K	Key matrix
N	Total number of patches
P	Patch size
Q	Query matrix
S	Stride
$SA_i(Z)$	Self-Attention output generated by the i^{th} head
V	Value matrix
W	Weight parameters
W^K	Key weight matrix
W^Q	Query weight matrix
W^V	Value weight matrix
X_{class}	Learnable class vector
$\mathrm{erf}(x)$	Gaussian error function of x
$\mathrm{U_{msa}}$	Multi-head Self-Attention (MSA) weight matrix
$\mathrm{ReLU(z)}$	REctified Linear Unit
$\tanh(z)$	Hyperbolic tangent function
b	Bias parameter

k^*	Optimal threshold value for Otsu thresholding method
η	Measure of separability to evaluate the “goodness” of the threshold
γ	Positive constant gamma for the power-law transformation
μ	Mean value
μ_0	Mean of class C_0
μ_1	Mean of class C_1
μ_T	Total class mean
ω_0	Probability of occurrence for class C_0
ω_1	Probability of occurrence for class C_1
σ	Standard deviation
$\sigma(z)$	Sigmoid function
σ_T^2	Total class variance
σ_B^2	Between-class variance

Chapter 1

Introduction

1.1 Overview

The first case of COVID-19 was reported in December-2019, and after that on 30th January 2020, the World Health Organization (WHO) declared COVID-19 as an international public health emergency [1]. During that time, researchers as well as medical industries around the world have been searching for different methods and new modalities to diagnose COVID-19. Due to the lack of COVID-19 testing kits around the world, many COVID-19 patients lost their lives. To overcome the issue of lack of testing kits, it is required to find alternative methods based on image modalities and radiological techniques. Moreover, currently, Artificial Intelligence (AI) research community is mainly focusing on the development of deep learning techniques based on different medical imaging modalities for diagnosing COVID-19. Here, in this work, different deep learning models are implemented and compared for automatic detection of COVID-19 from CT-Scan images.

After detection of COVID-19, it is also necessary to take proper medication. For that purpose, it is required to identify the severity level of the infection. Traditionally, an experienced doctor generally examines the medical images (such as CT-Scan or X-Ray images) to identify the severity level of the diseases from the infected regions, and then, based on it, the doctor suggests possible treatments for reducing the infection of the disease. Due to a large number of COVID-19 patients, it is required to develop an automatic method to obtain the severity level, if the patient is detected as COVID-19 positive, to reduce the diagnostic burden of the doctors. In this project, the image segmentation method is used to identify the infected region and the severity level of the infection from the lung CT-Scan image, only if it is classified as a COVID-19 positive using the deep learning method.

1.2 Literature Review

In this section, a brief summary of the different methods that have been developed by AI research community for automatic detection of COVID-19 using various medical imaging modalities is first given. It is then followed by a discussion of the image segmentation methods applied on COVID-19 images to obtain the infected region of the human lung.

1.2.1 Automatic COVID-19 Detection using Deep Learning Methods

During the last two years, a large number of deep learning algorithms have been developed on various types of medical image modalities for automatic detection of COVID-19. The deep CNN models including Residual Neural Network (ResNet)18, ResNet50, ResNet101, Visual Geometry Group (VGG)16, and VGG19 were implemented on chest X-Ray images for automatic COVID-19 detection by Ismael et al. in [2]. The above mentioned pretrained CNN models were used for the feature extraction. Furthermore, for the classification of the extracted features, the support vector machines (SVM) classifier was used. Ozturk et al. [3] have implemented the DarkNet model with 17 convolutional layers and gradually increased the number of filters for each layer. They developed the deep learning model for automatic COVID-19 detection from the X-Ray images with binary classification, namely, COVID-19 and No-Findings, as well as multi-class classification in terms of COVID-19, No-Findings and Pneumonia. The CNN models such as VGG19, MobileNet v2, Inception, Xception, Inception ResNet v2 have been described briefly in [4]. All these models are implemented on X-Ray images with 2-class and 3-class classifications. According to [4], VGG19 and MobileNet v2, are the two best models from the point of view of classification accuracy over the rest of the CNN models discussed in the article. Sethy and Behera [5] have tested 13 pre-trained CNN models including AlexNet, VGG16, VGG19, GoogleNet, ResNet18, ResNet50, ResNet101, Inception v3, Inception-ResNet v2, DenseNet201, XceptionNet, MobileNet v2 and ShuffleNet along with SVM classifier. The method in [5] classifies the input X-Ray image into 3 classes as healthy people, pneumonia patients and COVID-19 patients. Out of these 13 models, ResNet50 plus SVM gives the best performance yielding the highest accuracy. Automated X-ray imaging radiography systems (AXIRs) were introduced by S.H. Yoo et al. [6] in July 2020. Out of the four AXIRs, AXIR1 is for classification of X-Ray images into abnormal/normal, while AXIR2 is for classification as Tuberculosis (TB)/non-TB. If the input image is classified as abnormal through AXIR1, then AXIR2 will be used to determine whether it is TB or non-TB. Furthermore, AXIR3 and AXIR4 are used to classify the image into COVID-19 or non-COVID-19, with or without TB. The authors have used a two-dimensional CNN algorithm with PyTorch frame for the purpose of training as well as testing, and the whole architecture is based on pre-trained CNN model ResNet18 with ImageNet dataset. Panwar et al. [7] have proposed an alternative fast screening method known as nCOVnet, which is based on deep learning neural network. This method analyses the input X-Ray images of the lungs and classifies them into Positive or Negative for COVID-19. Furthermore, it uses a pre-trained CNN model VGG16 and is trained on the ImageNet dataset. Albahli [8] has developed a deep convolutional model with 152 hidden layers, which is derived from ResNet-152. The project mainly focuses on distinguishing COVID-19 from other lung diseases using deep learning methods from X-Ray images of the lungs. These images are classified into eight major classes based on various chest diseases. In November 2021, Jain and Medal [9] introduced the deep learning-based CORO-NET architecture for automatic detection of COVID-19 from X-Ray images and classifies them into three classes namely pneumonia patients, COVID-19 patients and normal subjects.

Singh et al. [10] have implemented a Multi-Objective Differential Evolution (MODE) based CNN for automatic detection of COVID-19 from the chest CT-Scan images. They compared the MODE model with CNN, Adaptive Neuro-Fuzzy Inference Systems (ANFIS), and Artificial Neural Network (ANN) models. A weakly-supervised framework from 3D chest CT volumes for COVID-19 classification and lesion localization was developed by Wnag et al. [11]. The lung portion was first segmented through a pre-trained U-Net in their method. Then, the 3D lung region was fed into a 3D deep neural network to determine the COVID-19 infection probability. The method is known as 3D deep convolutional neural network (DeCoVNet) to detect COVID-19. Ahuja et al. [12] have tested four different pre-trained CNN models such as ResNet18, ResNet50, ResNet101, and SqueezeNet for automatic detection of COVID-19 from lung CT-Scan images into 2 classes, namely, COVID positive and COVID negative. They achieved the best performance using ResNet18 with highest classification accuracy for training and validation.

Recently, attention based technique, Vision Transformer (ViT) [13] has been shown to achieve remarkable performance compared to CNN. ViT, based on a self-attention, was implemented by Shome et al. [14] for COVID-19 detection from chest X-Ray images. They classify the images into 3 classes, Normal, COVID-19 and Pneumonia. Krishnan and Krishnan et al. [15] implemented Vision Transformer (ViT)-based COVID-19 detection using X-Ray images. As per their research, ViT-B/32 (ViT baseline model with 32×32 input patch size) performed well compared to other CNN based approaches such as Inception V3, DenseNet, and WideResNet101.

Till now most of the researchers have focused on either X-Ray or CT-Scan imaging modalities for COVID-19 diagnostic purpose. Generally, chest X-Ray or CT-Scan images are used to cure lung diseases, and it has been always a debatable topic among researchers concerning which medical imaging modality is best suitable for the diagnostic of lung diseases. Both imaging modalities, X-Ray and CT-Scan have their own merits and demerits. Compared to CT-Scan images, X-Ray images are less expensive and readily available but have less visibility. According to the work of Aditya Borakati et al. [16], CT-Scan has substantially improved the diagnostic performance over the chest X-Ray for COVID-19. Moreover, it is very difficult to identify the infection levels from the chest X-Ray due to poor visibility of the X-Ray images. Furthermore, Kunwei Li et al. [17] have also concluded that the visual quantitative analysis of CT-Scan images provides good efficiency along with high diagnostic ability. Hence, in this research work, CT-Scan images are preferred over the chest X-Ray images.

1.2.2 Lung Image Segmentation Methods for Different Imaging Modalities

For a proper analysis of the Lung medical images, the image segmentation result is beneficial in retrieving the area of interest from given radiological images. A. Mansoor et al. [18] have explained the capabilities and performance of the current lung image segmentation approaches along with challenges and future trends. They divided the currently used segmentation methods for the lung CT images into five significant classes, namely, thresholding-based, region-based, shape-based, neighboring anatomy-guided, and machine learning-based methods. Amin et al. [19] have proposed a 3-D semantic segmentation and classification approach for COVID-19 diagnosis using CT-Scan images. They used the Gabor filter and thresholding for image pre-processing to improve the image quality. Then, to separate the lung region, the marker-based watershed segmentation approach along with thresholding was used. In order to get COVID-19 infected region, an encoder/decoder-based deep learning model was used in the segmentation phase. Finally, SVM, Ensemble Tree, and Extreme Learning Machine (ELM) classifiers were used in the last phase to classify the segmented images into COVID-19 and non-COVID-19, based on the extracted statistical and geometrical features. A three-stage segmentation framework known as Key and Intermediate frame of Segmentation (KISEG) was introduced by Xiaohong Liu et al. [20]. This approach is useful to enhance the performance on serial CT image segmentation with multi-level acceleration. Moreover, for data augmentation, they proposed Gaussian Kernel Dropout (GKD). Satapathy et al. [21] have introduced Cuckoo Search Algorithm (CSA), which is based on Kapur [22] or Otsu [23] image thresholding method and Level-Set (LS) [24] or Chan-Vese (CV) [25] image segmentation method to extract COVID-19 infected region from the input CT-Scan slices. Finally, they compared the results obtained by all possible combinations of thresholding and image segmentation methods, namely, Kapur+LS, Otsu+LS, Kapur+glsCV, and Otsu+CV. Rajinikanth et al. [26] have proposed a firefly-Algorithm for COVID-19 infected region from lung CT-Scan images, which are based on Shannon entropy and Markov Random Field (MRF). They used the firefly algorithm along with Shannon entropy for image pre-processing to enhance the pneumonia lesion. Then, during image post-processing, they used the MRF segmentation method to extract the lesion with reasonable accuracy. Valizadeh and Shariatee [27] have discussed various semantic segmentation methods, also known as pixel-level classification of medical images for COVID-19 detection. The study was divided into two parts: traditional image segmentation methods and deep neural network-based image segmentation methods. They covered most of the traditional image segmentation approaches including Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), Local Binary Pattern (LBP), SURF method, Harris Corner Detection, K-Mean algorithm, SVM, MRF, etc.

along with a comparison of their performance. Furthermore, recent deep neural network-based approaches such as ANN, Fully Convolutional Network (FCN), pyramid methods, and various CNN models were also described with their advantages and disadvantages related to COVID-19 diagnostic purpose. Harmony-search-optimization [28] and Otsu thresholding-[23]based COVID-19 detection method from lung CT-scan images has been introduced by Rajinikanth et al. [29] in April, 2020. This approach used a threshold filter to separate the lung region and then used harmony-search-optimization and otsu thresholding for image enhancement. Finally, they implemented the watershed image segmentation method to obtain the COVID-19 infected region and the severity level of COVID-19 infection. N Khehrah et al. [30] have implemented the fully automatic framework for nodule detection using image segmentation from lung CT-Scan images. The method is divided into two phases; the first phase is lung segmentation, in which the histogram-based thresholding along with morphological operations and connected components analysis is used to extract the lung portion. The second phase is nodule detection, which comprises separation of the inner structure followed by statistical and shape-based feature extraction, and SVM classifier to separate the lung nodules from other lung components.

1.3 Motivation

The world is still struggling against COVID-19 and its different variants even after two years of the first COVID-19 case. As of December-2021, a total of 275,790,573 COVID-19 cases have been reported over 224 countries and out of this, 5,376,566 people have died due to COVID-19 during the past two years [31]. COVID-19 testing kit, which is also known as RT-PCR, is not available universally. To overcome the issue of the lack of COVID-19 testing kit, it is necessary to develop COVID-19 diagnosis methods, which are easily available over the whole world, and at the same time that are fast, efficient, and based on medical imaging modalities.

Nowadays, AI is very popular for solving complex problems with automation along with its subsets of machine learning and deep learning-based models. Deep learning models are useful for enhancing the image features that are not visible in the original medical images and then classifying the images according to the extracted features. Many research works have already been published on the automatic classification of the images into COVID-19 positive and negative classes based on deep learning models.

Traditionally, the infected region from the various medical imaging modalities is identified by experienced doctors, and then further treatment suggested accordingly. However, this process is very time-consuming and becomes tedious if there are very large number of patients. The main aim of this study is to compare the different deep learning models for the detection of COVID-19 CT-Scan images and then obtain the severity level of the infection from the lung CT-Scan images based on image segmentation methods to reduce the processing time and diagnostic burden of the physicians.

1.4 Objectives of the Thesis

Most of the research hitherto is on either deep learning-based COVID-19 classifications or segmentation of COVID-19 images. In this thesis, we propose to combine these two tasks, i.e., if a patient is identified to be COVID-19 positive, then the image segmentation method is applied on the CT-Scan images to find the severity level of the infection for further diagnosis. The main objective of this research work is to include automation for both the identification of the disease and its severity level. This should save a lot of time for the doctors.

The goal of this research work is to propose the use of a vision transformer (ViT) for COVID-19 screening using CT-Scan images instead of using CNN methods. Hence, we implement the self-attention-based ViT approach on the SARS-CoV-2 CT-Scan dataset

[32] and call it COViT-CT. Moreover, we also implement various CNN methods, including ResNet50, DarkNet19, GoogLeNet, and VGG16, to compare the results obtained by the attention-based COViT-CT method with CNN methods. After comparing the results through standard comparison metrics, we choose the best method for lung CT-Scan image classification. Finally, to achieve the main goal, we implement the lung CT-Scan segmentation model to identify the infected lung region and severity of the diseases of positive COVID-19 patients. This will help the doctor to provide proper medications and further treatment to the patients.

1.5 Thesis Outline

The rest of the thesis is organized as follows.

Chapter 2 describes artificial neural network architecture along with convolutional neural network architecture with different layers. In this chapter, the transfer learning approach which uses pre-trained CNN models is described for COVID-19 detection from lung CT-Scan images. It provides a detailed description of the implementation of various CNN networks, namely, ResNet50, DarkNet19, GoogLeNet, and VGG16.

Chapter 3 presents the vision transformer, a self-attention-based deep learning technique in detail. This chapter discloses the brief information on the self-attention network. It provides a description of the implementation of the COViT-CT network along with various stages such as patch and position embedding, and transformer encoder.

Chapter 4 discusses the image dataset used in this work, the experimental setup, and the environments. The results obtained by various pre-trained CNN models and the COViT-CT method are also presented. All the results are compared using the standard performance evaluation metrics. Furthermore, the results obtained by the COViT-CT method are compared with the recent state-of-art works in this chapter.

Chapter 5, describes the image segmentation process for COVID-19 positive lung CT-Scan images to obtain the severity level of the disease for further diagnosis. This chapter describes the detailed steps and mathematical formulas for different stages of the overall segmentation task, including image enhancement, image pre-processing, image thresholding, and finally, image segmentation using the watershed image segmentation method. It also describes as how to compute the severity level of the disease after extracting the infected region from the COVID-19 positive CT-Scan images. Moreover, the output of segmentation of the COVID-19 positive lung CT-Scan images are also presented in this chapter.

Chapter 6 concludes the thesis by summarizing the results obtained and discussing possible future directions related to this research work.

Chapter 2

Deep Learning Models for COVID-19 Detection

2.1 Deep Learning and Artificial Neural Networks

Artificial intelligence (AI) breaks the ceilings of all cutting-edge technologies, and involves the technique that enables machines to mimic human intelligence using various logic and rules. Machine learning is a subset of AI, which uses statistical approaches to analyze the data to improve the performance. The term “Machine Learning” was first introduced by Samuel in 1959 during the design of an algorithm for the game of checkers [33]. Moreover, deep learning is a subset of machine learning and one of the many applications of artificial neural networks (ANNs). Over two decades, deep learning has been a booming field in medical imaging among the different techniques based on AI. The term “Deep Learning” was first introduced by Rina Dechter to the machine learning community in 1986 [34], and it was the first time mentioned for Artificial Neural Networks by Aizenberg et al. in 2000 [35]. Deep learning is generally focused on creating large neural network models with multiple layers that are capable of making accurate data-driven decisions [36]. It is very beneficial for complex and very large datasets. The term “Deep” in “Deep Learning” represents multiple layers between the input and output, and each layer produces the output with its input from the previous layer. These multiple hidden layers help identify and extract valuable features and patterns from a large complex dataset to produce the final data-driven decisions.

Artificial neural networks, also known as neural networks, are biologically inspired algorithms, modeled based on neurons of brains and nervous systems and designed to simulate the way in which the human brain processes the information [37], [38]. Hence, it is required to understand how human brain works to model the ANN. Single neuron structure was drawn by a Spanish neuroscientist, Santiago Ramón y Cajal, around 1900 is shown in Figure 2.1 [39]. The figure shows that a typical neuron comprises three functionally distinct parts: a large number of dendrites, the soma, and the axon. The dendrites are responsible for collecting the signals from other neurons and hence act as an input device. The collected neurons are then transmitted to the soma, which acts as a central processing unit and is responsible for performing a nonlinear processing step with a threshold function to determine whether the total input arriving at the soma has a high enough potential to trigger an output. If this threshold is passed, then the output signal is generated, and then it passes to the other neurons through the axon, which acts as an output device.

2.2 Mathematical Model of Artificial Neural Networks

In 1958, Frank Rosenblatt [40] developed the very first artificial neural network called the perceptron, which was a probabilistic model for information storage and organization in the human brain. Figure 2.2 [41] describes the mathematical model of an artificial neural network inspired by the function of a single neuron. Artificial neuron takes several inputs $x_0, x_1, ..., x_i$, processes them by summing them together, and finally applies an activation function to obtain the output signal y that can be passed to the next hidden layer of the network. As shown in Figure 2.2, summation of all the inputs is carried out in a weighted

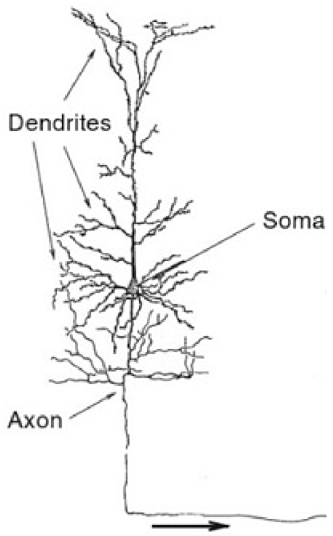


Figure 2.1: Structure of neurons by Ramón y Cajal (adapted from [39])

way, i.e., all the inputs are scaled up or scaled down based on the specific weight assigned to that particular input. Along with these weight parameters, another parameter, the neuron's bias, is also trained and used for the summation process. All the inputs are summed up and scaled using the weight parameters, and the final result is assigned to variable z. The overall process can be expressed as

$$z = X \cdot W + b \tag{2.1}$$

where X is a row vector consisting of the input signals, W denotes a column vector containing weight parameters and b is a bias parameter. Hence, the dot product between these two vectors can be denoted by:

$$\begin{aligned} X \cdot W &= \begin{pmatrix} x_0 & x_1 & \cdots & x_i \end{pmatrix} \cdot \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_i \end{pmatrix} \\ &= \sum_i x_i w_i \\ &= x_0 w_0 + x_1 w_1 + \ldots x_i w_i \end{aligned} \tag{2.2}$$

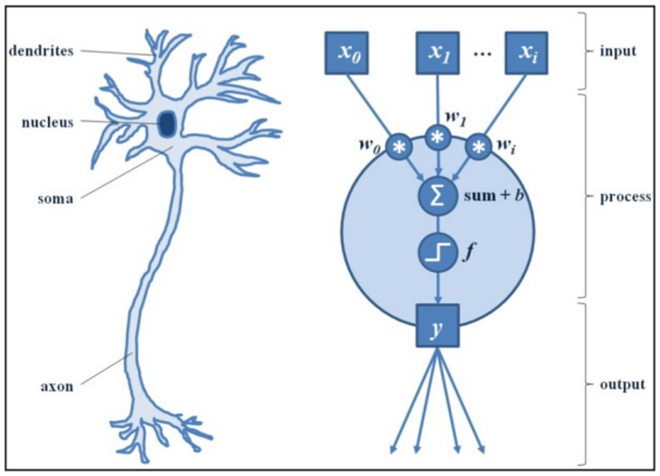


Figure 2.2: Comparison between biological neuron with artificial neural network (adapted from [41])

Finally, in order to get the neuron's output, the activation function f is applied to output z. As per the biological function of the neurons, the soma acts as a central processing unit. By comparing biological neurons with the mathematical model of an artificial neuron, the activation function would be a binary function, having an output 1 if z is greater than or equal to a threshold value t, otherwise, the output is 0. Mathematically, it can be expressed as the step function

$$y = f(z) = \begin{cases} 0 & \text{if } z < t \\ 1 & \text{if } z \geq t \end{cases} \tag{2.3}$$

Furthermore, for more advanced properties, other activation functions are also used instead of step function, such as, the sigmoid function $\sigma(z)$, the hyperbolic tangent $\tanh(z)$, the REctified Linear Unit (ReLU) $\mathrm{ReLU(z)}$. All above mentioned activation functions can be expressed mathematically as [38]

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.4}$$

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.5}$$

$$\mathrm{ReLU}(z) = \max(0, z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases} \tag{2.6}$$

We have discussed the mathematical model of a simple artificial neuron, through which the received input signal would be processed, and then generated output signal would be forwarded to other neurons to build the network, which is known as the artificial neural network. Deep ANN contains multiple layers, including one input layer, one output layer, and multiple hidden layers between the input and output layers. Each layer consists of many neurons, each receiving the same input as the others and having the same activation function. Moreover, as discussed, each neuron first sums all the inputs with their specific weights along with bias values. Figure 2.3 represents the network with multiple hidden layers.

To understand the mathematical model of the ANN, let us consider a network with 3 layers and 2 inputs (x_1 and x_2) that produces the one final output y_H, as shown in Figure 2.4 [41]. In this network, the input layer, i.e., layer one, consists of three neurons, A, B, and C; the hidden layer, i.e., layer 2, has four neurons, D, E, F, and G; and the output layer 3 contains only one neuron H. Hidden layer 2 takes the three inputs from the previous layer

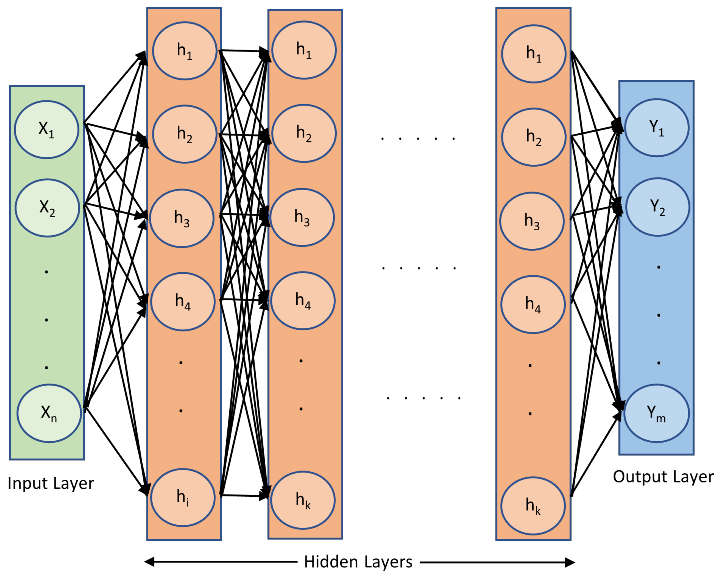


Figure 2.3: Artificial neural network with multiple hidden layers

and generates four outputs that are passed to the next layer. Layer 2 of this network is known as a fully connected layer or dense layer as each neuron of the layer is connected with every neuron of the preceding layer.

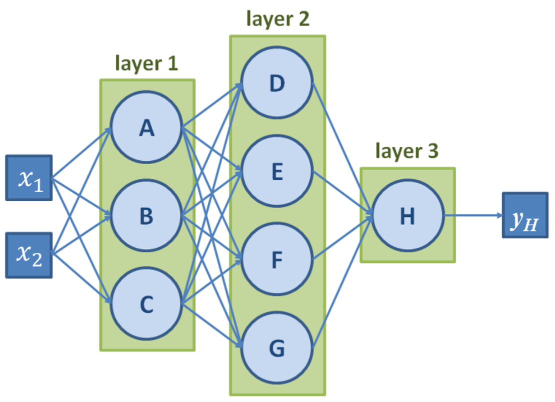


Figure 2.4: Artificial neural network with three layers (adapted from [41])

Similar to the mathematical model of single neuron, the output produced by the first layer is given by:

$$\begin{aligned} z_A &= x_1 \cdot w_{a1} + x_2 \cdot w_{a2} + b_A \\ z_B &= x_1 \cdot w_{b1} + x_2 \cdot w_{b2} + b_B \\ z_C &= x_1 \cdot w_{c1} + x_2 \cdot w_{c2} + b_C \end{aligned} \tag{2.7}$$

Equation (2.7) can be rewritten as

$$z = x \cdot W + b \tag{2.8}$$

where input array $x = \begin{pmatrix} x_1 & x_2 \end{pmatrix}$, weight parameters $W = \begin{pmatrix} w_{a1} & w_{b1} & w_{c1} \\ w_{a2} & w_{b2} & w_{c2} \end{pmatrix}$, bias b= $\begin{pmatrix} b_A & b_B & b_C \end{pmatrix}$, and output $z = \begin{pmatrix} z_A & z_B & z_C \end{pmatrix}$.

Hence, the final output of the first layer after applying activation function is given by $y = f(z) = \begin{pmatrix} f(z_A) f(z_B) f(z_C) \end{pmatrix}$, which is passed on to the second layer as an input vector, and so on until the last layer.

2.3 Convolutional Neural Network

Convolutional neural network (CNN), also known as the ConvNet, is the type of ANN generally used for the analysis of visual image datasets to extract particular features from the images and classify them according to the extracted features. CNN are widely used in computer vision, image and video recognition, speech recognition, image classification, natural language processing, medical image analysis, and bioinformatics.

2.3.1 CNN Architecture

Figure 2.5 illustrates the basic architecture of a convolutional neural network. CNN architecture consists of two main parts, namely, feature extraction and classification. In a CNN, a convolution tool is used to recognize the various features from the input image for further analysis. Finally, a fully connected layer classifies the image based on the extracted features. Convolutional layers and pooling layers are generally responsible for feature extraction. In some CNN algorithms, batch normalization layers are also used along with the previously mentioned layers.

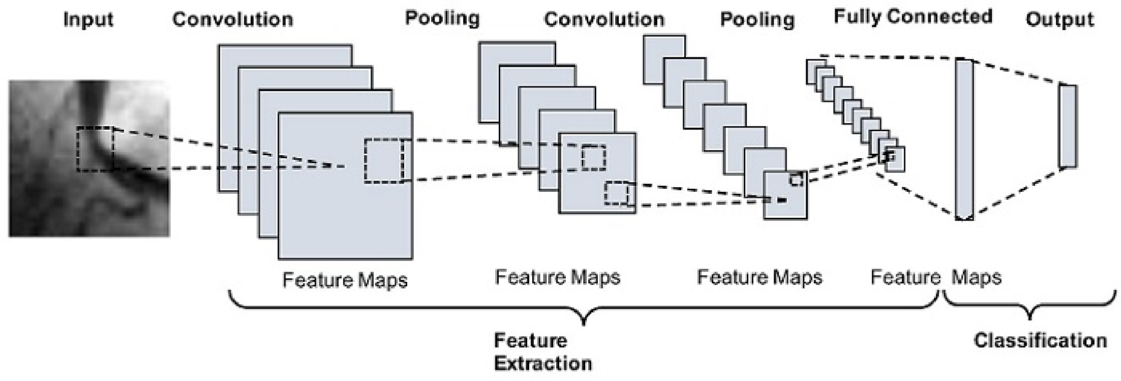


Figure 2.5: CNN architecture with various layers (adapted from [42])

2.3.2 Convolutional Layers

In CNN, the convolutional layer is mainly responsible for extracting the essential features from the input images. Despite its colloquial name, convolution, this layer theoretically executes the sliding dot product or cross-correlation operation between the image and a kernel. A kernel is a matrix consisting of learnable values called weights and this matrix is slid across the 2-D input image, and element-wise multiplied by the corresponding values of the input image and then all these products added to obtain a single output. Each of these convolution kernels acts as a distinct filter, generating an output feature map that represents different characteristics, such as kernels for detecting top edges, bottom edges, diagonal lines, etc. The final output image produced through convolution operation contains the essential features of the input image, and is known as a feature map. The total number of feature maps produced during this layer depends on the number of kernels; for example, if n kernels are applied to the input image, it will produce a total of n feature maps.

To understand the convolutional operation in the neural network, let's take a 5×5 input image matrix and an edge detection kernel as shown in figure 2.6. Now, to calculate the convolution for the pixel located at $(2, 2)$ i.e., the second row and the second column (which is highlighted by the red color box) of the input image, the element-wise multiplication with the filter kernel is carried out followed by the summation, i.e., $(1 \times -1 + 0 \times 0 + 2 \times 1 + 0 \times -1 + 1 \times 0 + 0 \times 1 + 0 \times -1 + 0 \times 0 + 1 \times 1)$ to generate the output value 2. Then, we slide the kernel to the right and repeat the convolution operation. We repeat this whole process, sliding the kernel to the right and downwards until it reaches the lower-right corner of the image.

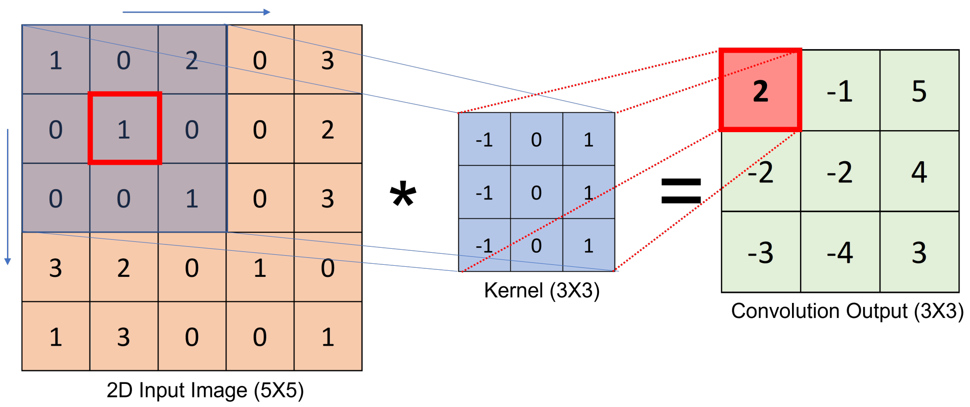


Figure 2.6: Convolution operation

The central issue of the convolution operation is that it does not involve the pixels located at the border of the image, as the size of a kernel is generally less than the size of the image. To solve this issue, padding is used, through which extra pixels are added around the boundary of the image. There are two different types of padding available in deep learning: Valid padding and Same padding. Valid padding means no padding. However, the same padding produces the output feature map with the same size as the input image size. In the same padding, zeros are appended to the outer frame of the image and the size of the padding depends on the kernel size. If kernel size is $k \times k$, where k is odd, then the padding size is $(k-1)/2$.

Figure 2.7 illustrates the same padding operation carried out on the 5×5 image shown in Figure 2.6 for the kernel size 3×3. Hence, the size of the padding is $(3-1)/2 = 1$. Hence, an extra row and a column of zeros are added on all the sides of the image, as shown in Figure 2.7, which effectively increases the input image size from 5×5 to 7×7.

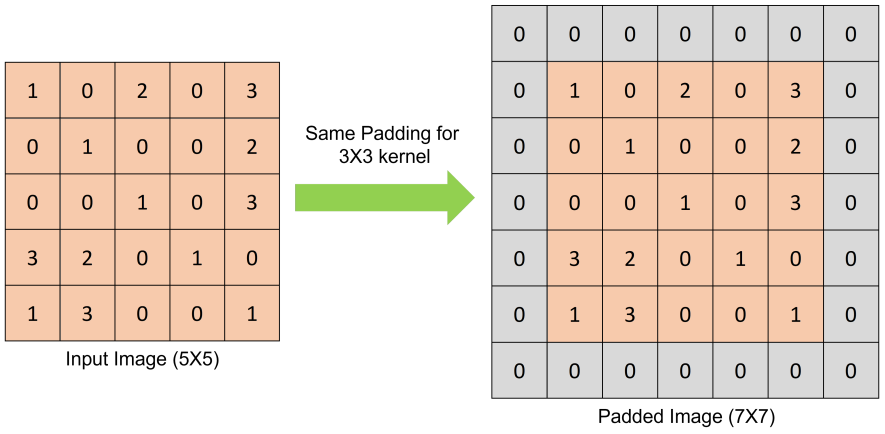


Figure 2.7: Same padding

Another convolution parameter is the stride that is used to reduce the size of the output feature map. It denotes the number of pixels by which the filter slides over the input image. Figure 2.8 represents the convolution example with stride value S= 2 which produces the output image of size 3×3 from input image 5×5 with a padding of one. Here, the filter skips one pixel every time while traversing the input image, as shown in the figure. The output feature map is generated through the convolutional layer by the dot product between the input image and filter W_0, considering bias value $b_0 = 0$.

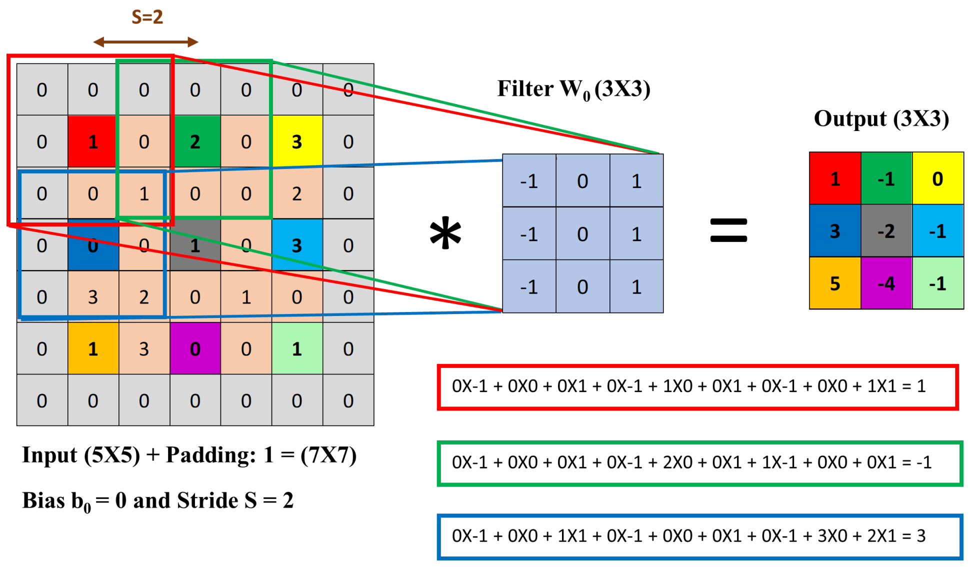

S=2
Filter W_0 (3X3)
Output (3X3)
Input (5X5) + Padding: 1 = (7X7)
Bias b_0 = 0 and Stride S = 2
0X-1 + 0X0 + 0X1 + 0X-1 + 1X0 + 0X1 + 0X-1 + 0X0 + 1X1 = 1
0X-1 + 0X0 + 0X1 + 0X-1 + 2X0 + 0X1 + 1X-1 + 0X0 + 0X1 = -1
0X-1 + 0X0 + 1X1 + 0X-1 + 0X0 + 0X1 + 0X-1 + 3X0 + 2X1 = 3

Figure 2.8: Convolution operation between image and kernel with stride 2 and padding 1

2.3.3 Pooling Layers

The primary purpose of the pooling layer is to reduce the size of the convolved feature map. In most cases, the pooling layer is generally used after one or more convolutional layers. Generally, in CNN, two types of pooling operations are used, these are max pooling and average pooling. Max pooling selects the pixel with the highest intensity value from the predefined sized image sections. However, average pooling calculates the mean of all the elements of the pooling region. Figure 2.9 illustrates the output generated through the two types of pooling operations. The selection of pooling methods depends on the image datasets. Max pooling is essential to extract the sharpest features from the feature map. On the other hand, in average pooling, all values of the pooling region are considered. Max pooling fails when most of the elements in the pooling region have the highest intensity value, and average pooling fails when too many zero elements are present in the pooling region [43].

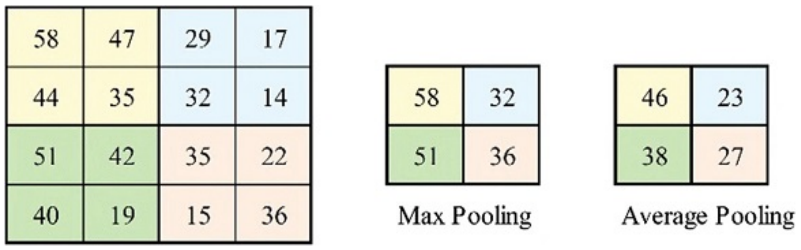


Figure 2.9: Pooling layer: Max pooling and average pooling (adapted from [44])

2.3.4 Batch Normalization

Batch Normalization, also known as Batch Norm, is an optional layer and is often placed after one or more convolutional layers in CNN architecture. To understand the working of the Batch Norm layer, it is required first to discuss normalization, which is a pre-processing technique used for input-data standardization. Data normalization helps to set the range of the input data into the appropriate range. There are two different normalization methods. In the first method, input data is scaled to a range from 0 to 1 using the formula:

$$x_{\text{normalized}} = \frac{x - \mu}{x_{\max} - x_{\min}} \tag{2.9}$$

where x is the input data-set, μ represents the mean value of the given data-set, and $x_{\max}$ and $x_{\min}$ denote the maximum and the minimum values, respectively, from the data-set.

The second method is generally force the overall data-set to have a mean value 0 and a standard deviation value 1 through the following formula:

$$x_{\text{normalized}} = \frac{x - \mu}{\sigma} \tag{2.10}$$

where x is the input data-set, μ is the mean of the input data-set, and σ is the standard deviation of the data-set.

Now, Batch Normalization is very similar to the normalization used to normalize batches of data inside the neural network itself. In short, batch normalization is used between the layers of an artificial neural network instead of the raw data sets, which helps to increase the overall training speed and use higher learning rates [45].

Let us consider the m inputs over a mini-batch $B = \{x_1 \cdots x_m\}$. Then the output of batch normalization is given by,

$$y_i = \gamma \cdot \frac{x_i - \mu_B}{\sigma_B} + \beta \tag{2.11}$$

where γ and β are learning parameters of the Batch Norm responsible for scaling and shifting of normalized value respectively. The mean μ_B and the standard deviation σ_B of the input data of the mini-batch B can be calculated using

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.12}$$

$$\sigma_B = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 + \epsilon} \tag{2.13}$$

where ϵ is a small constant used to avoid division by zero.

2.3.5 Classification Layers

After feature extraction, it is required to classify the extracted features of the processed image. Generally, classification layers of CNN comprise a flatten layer, several fully connected (FC) layers, a dropout layer (optional), and finally, an activation function. The flatten layer is responsible for flattening the output of the convolutional or pooling layer by converting the 2-D feature matrix into a 1-D array. This single long feature vector is then connected to the final classification model, known as fully connected layers or FC layers, where the mathematical operations occur. Sometimes, to reduce the model's size and increase the computational speed, some neurons are dropped from the network based on the probability and statistics using a dropout layer. Finally, the most important parameter of classification layers is the activation functions such as ReLU, Softmax, hyperbolic tangent (tanh), and Sigmoid functions. Usually, softmax and sigmoid functions are used for binary classification, and the softmax activation function is preferred for multi-class classification.

Based on the CNN concept discussed in Section 2.3.1, there are various CNN architectures. Out of these architectures, we implement four CNN models, namely, ResNet50, DarkNet19, GoogLeNet, and VGG16, (which are discussed briefly in the following section), using a transfer learning approach.

2.4 Implementation of Various Neural Network Architecture

Figure 2.10 demonstrates the flow of the overall process of COVID-19 screening from the CT-Scan image dataset. The figure shows various pre-trained neural network models, namely, ResNet50, DarkNet19, GoogLeNet, and VGG16 are implemented on lung CT-Scan image dataset for automatic classification of the input images into two classes, namely, COVID and Non-COVID. If the input image is classified as COVID-19 positive, the infected region is identified using the lung image segmentation technique for further analysis.

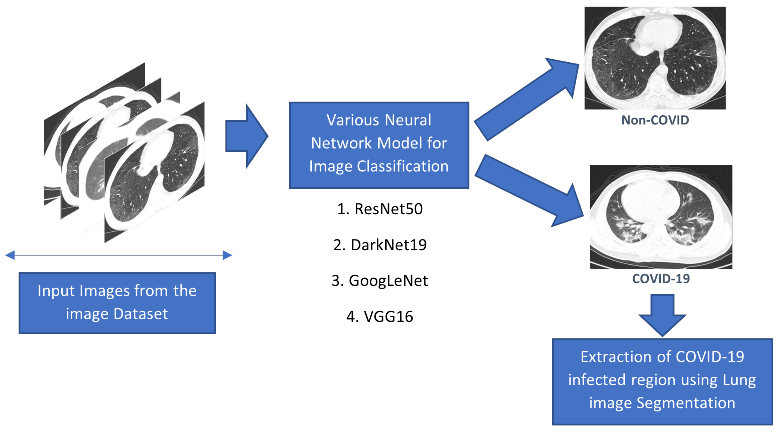

Input Images from the image Dataset
Various Neural Network Model for Image Classification
1. ResNet50
2. DarkNet19
3. GoogLeNet
4. VGG16
Non-COVID
COVID-19
Extraction of COVID-19 infected region using Lung image Segmentation

Figure 2.10: Process flow of COVID-19 screening methodology

2.4.1 Transfer Learning

Transfer learning is a deep learning technique in which a model trained for one task (a pre-trained model) is applied to a different task. In the present work, various pre-trained deep neural network models ResNet50, DarkNet19, GoogleNet, and VGG16 are implemented using a transfer learning approach in which the last three layers of the pre-trained model are fine-tuned based on the classes of an input image dataset, as shown in Figure 2.11. In transfer learning, image features are extracted using pre-trained CNN models, and layers responsible for final image classification are replaced by new layers to categorize images based on new required classes.

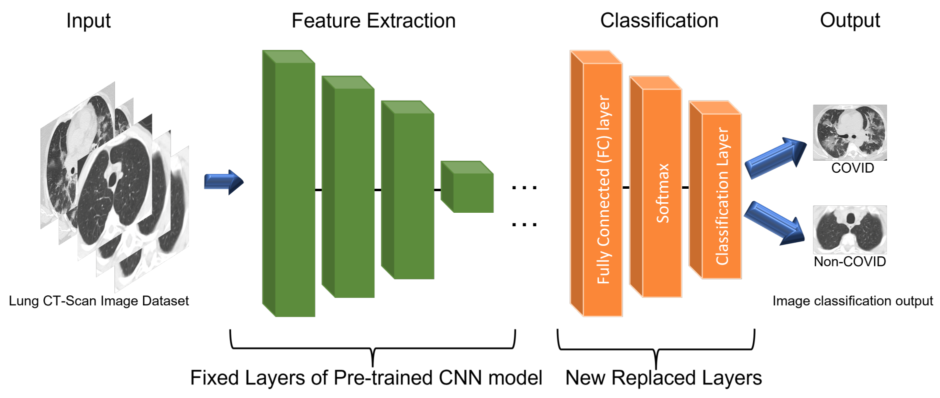

Input
Feature Extraction
Classification
Output
Lung CT-Scan Image Dataset
Fully Connected (FC) layer
Softmax
Classification Layer
COVID
Non-COVID
Image classification output
Fixed Layers of Pre-trained CNN model
New Replaced Layers

Figure 2.11: Transfer Learning mechanism

2.4.2 ResNet50

ResNet [46] was originally introduced by Microsoft Corporation, which is based on a residual learning framework. ResNet was designed to solve two major deep neural network problems known as the vanishing gradient problem and degradation problem. In the vanishing gradient problem, the derivative of loss functions will become smaller and smaller as we go backward with every layer during the training of the deep neural network. If there are a large number of hidden layers, then the vanishing gradient problem causes unstable network behavior and slows down the training speed of the network. Another problem with increasing the depth of the network is the degradation problem, that is, as the network depth increases, accuracy gets saturated and then degrades rapidly as it becomes difficult for the layers to propagate the information from the shallow layers, and due to that, the information is lost [46]. As shown in Figure 2.12, ResNet comprises residual connections, also known as skip connections, which ensures that the next layer of the network will perform at least as good as the previous layer, not worse than that.

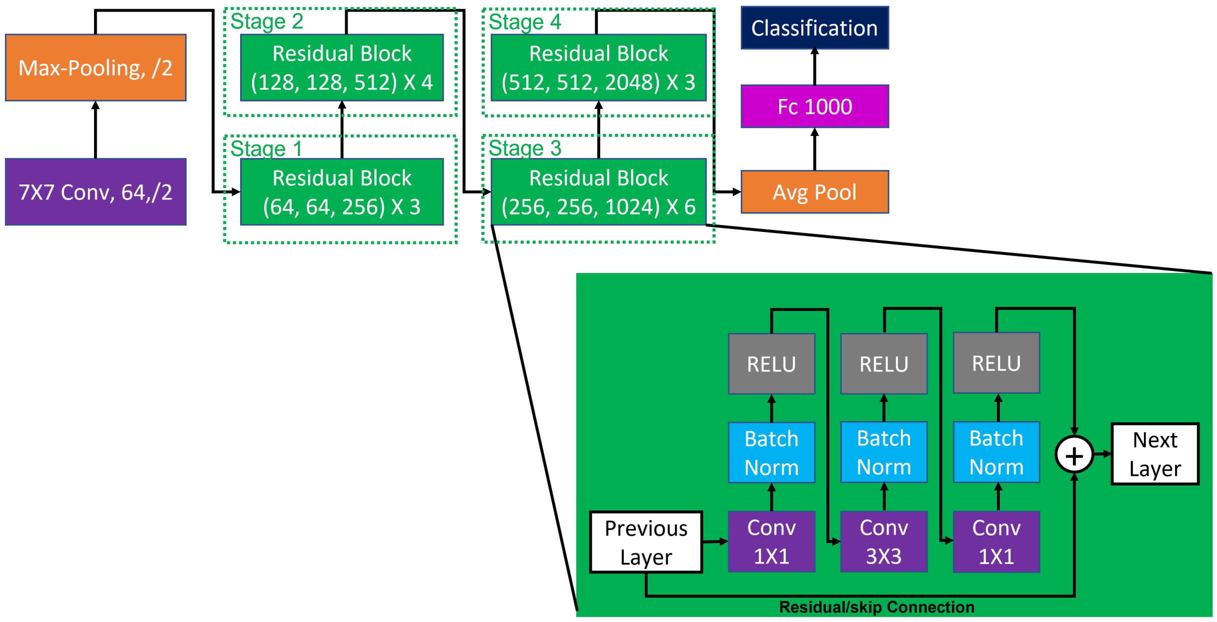


Figure 2.12: ResNet50 pre-trained model

ResNet is originally trained on ImageNet 2012 dataset [47] and classifies the input images into 1000 different categories. Kaiming He et al. introduced five different ResNet models based on the number of layers of the residual network, namely, ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152 [46]. The architecture of the ResNet50 model, which is 50-layer residual network architecture with a total of 177 layers, is shown in Figure 2.12.

In the ResNet50 architecture, the initial layer consists of a convolution layer using a kernel of size 7×7 with stride $S = 2$, and the number of filters is 64, followed by a max-pooling layer with stride $S = 2$. This is followed by four stages comprising a total of 16

residual blocks. Each residual block contains three convolution layers with kernels of size 1×1, 3×3, and 1×1, respectively. Hence, there are 16×3 convolution layers belonging to the residual blocks plus the first convolutional layer for a total of 49 convolution layers in the ResNet50 architecture. Stage 1 of the network comprises three residual blocks with 64, 64, and 256 convolutional filters in each residual block. The second stage has four residual blocks having 128, 128, and 512 convolutional filters in each residual block. Stage 3 and stage 4 contain six and three residual blocks, respectively. In stage 3, there are 256, 256, and 1024 convolutional filters in each residual block, whereas in stage 4, the number of convolutional filters is 512, 512, and 2048 in each residual block. Finally, the classification layer includes average pooling, a fully connected layer with 1000 nodes, and a softmax function. Hence, in the ResNet50 architecture, the total number of residual network layers is 49 (convolution layers) $+1$ (classification layer) $= 50$ layers.

In the present research work, ResNet50 pre-trained network is loaded into MATLAB and then the last three layers “fc1000”, “fc1000_softmax”, and “ClassificationLayer_fc1000” are replaced by “new_fc”, “new_softmax”, and “new_classoutput” layers, respectively, as per the transfer learning approach discussed in Section 2.4.1. Input image size for ResNet50 is 224×224, and hence, all the images of the dataset are resized to that size during pre-processing.

2.4.3 DarkNet19

DarkNet19, based on You Only Look Once (YOLO) V2 object detector [48], is a convolutional neural network with 19 convolutional layers and 64 total layers. The pre-trained version of DarkNet19 was trained on more than one million images of the ImageNet dataset [47] and classified the images into 1000 different categories, same as ResNet50. The detailed information of different layers of the DarkNet19 network is described in Table 2.1.

As discussed in the Transfer Learning approach, the DarkNet19 pre-trained network is

Table 2.1: Different layers of the DarkNet19 pre-trained network [48]

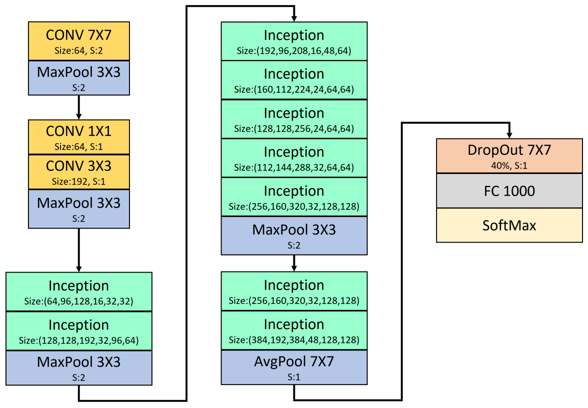
Type	**Filters**	**Size/Stride**	**Output**
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

loaded into MATLAB and the last three layers are replaced by new layers before performing the classification task. Input image size for DarkNet19 is 256×256, and hence, all the dataset images are resized to that size during pre-processing.

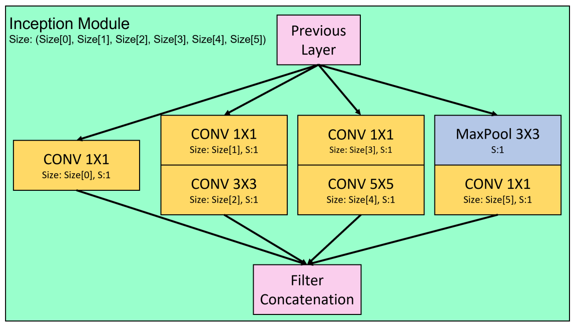
2.4.4 GoogLeNet

GoogLeNet, based on stacked inception modules, was introduced by Szegedy et al. in 2015 [49]. The network is 22 layers deep when considering only layers with parameters, and 27 layers deep if the pooling layers are also taken into consideration. The pre-trained network is trained using the ImageNet [47] or Places365 [50] datasets, which classify the input images into 1000 object categories or 365 various place categories, respectively.

The overall architecture of the network is represented in Figure 2.13(a) which includes the inception module of different size convolution layers. The inception module comprises a total of four streams including a 1×1 convolution layer; a 1×1 convolution layer followed by a 3×3 convolution layer; a 1×1 convolution layer followed by a 5×5 convolution layer; and finally a 3×3 max-pooling layer followed by a 1×1 convolution layer as shown in Figure 2.13(b). The output obtained by each four streams are concatenated and transferred to the next layer.


CONV 7X7
Size:64, S:2
MaxPool 3X3
S:2
CONV 1X1
Size:64, S:1
CONV 3X3
Size:192, S:1
MaxPool 3X3
S:2
Inception
Size:(64,96,128,16,32,32)
Inception
Size:(128,128,192,32,96,64)
MaxPool 3X3
S:2
Inception
Size:(192,96,208,16,48,64)
Inception
Size:(160,112,224,24,64,64)
Inception
Size:(128,128,256,24,64,64)
Inception
Size:(112,144,288,32,64,64)
Inception
Size:(256,160,320,32,128,128)
MaxPool 3X3
S:2
Inception
Size:(256,160,320,32,128,128)
Inception
Size:(384,192,384,48,128,128)
AvgPool 7X7
S:1
DropOut 7X7
40%, S:1
FC 1000
SoftMax

(a)



(b)

Figure 2.13: (a) GoogLeNet pre-trained model (b) Inception module block

In the present work, the pre-trained GoogLeNet network is loaded into MATLAB, and the last three layers of the pre-trained network are replaced with the new layers for the classification of input images as per requirements. The input image size for this network is 224×224.

2.4.5 VGG16

VGG (Visual Geometry Group) was first introduced by Simonyan and Zisserman from Oxford University [51]. VGG is a very deep convolutional network in which the depth of the network is increased using very small convolutional filters of size 3×3. VGG16 which is 16 layers deep comprises 13 convolutional layers and three fully-connected layers. The total number of layers of the VGG16 is 41.

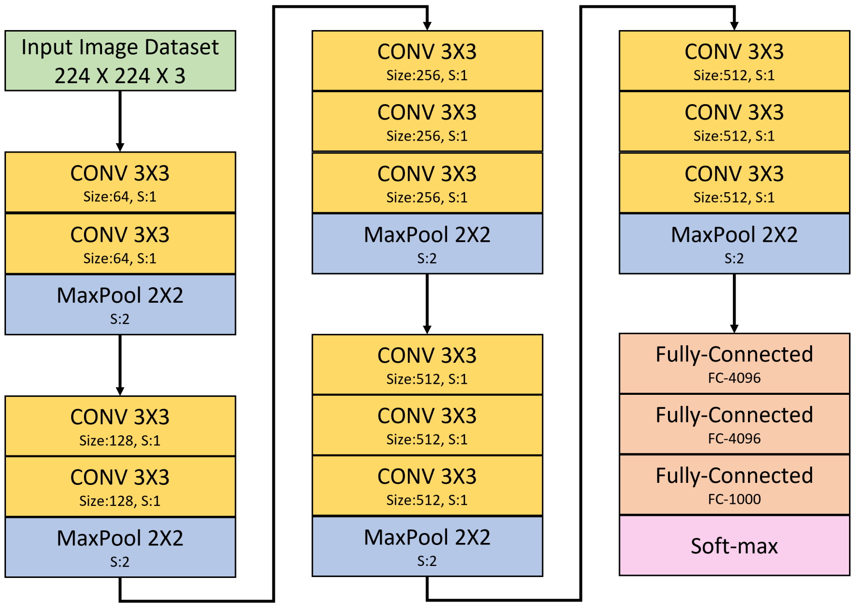


Figure 2.14: VGG16 pre-trained model

The network architectures of VGG16 is demonstrated in Figure 2.14. As shown in the figure, the network comprises 3×3 convolutional layers with stride $S = 1$ and 2×2 max-pooling layers with stride $S = 2$. The network was trained on the ImageNet dataset [47] and classified the images into 1000 different categories with input image size 224×224.

Here, we use a pre-trained network and change the last three layers of the pre-trained network to classify the input lung CT-Scan images into two classes, namely, COVID and Non-COVID, as discussed in Section 2.4.1.

2.5 Summary

In this chapter, the basic concept of the biological neural network, the mathematical model of the artificial neural network, and the architecture of the convolutional neural network with various layers of the network have been discussed in detail. Moreover, the overall process flow of the COVID-19 screening methodology and the transfer learning approach have been presented. Finally, the structures of various CNN models ResNet50, DarkNet19, GoogLeNet, and VGG16 have been presented. The results obtained using these CNN models will be presented and compared in Chapter 4. In the next chapter, a self-attention-based deep learning approach will be discussed in detail for the COVID-19 screening.

Chapter 3

Vision Transformer for COVID-19 Detection

3.1 Introduction

Transformer, a self-attention-based deep learning approach, was introduced initially for natural language processing (NLP) at Google Brain by Vaswani et al. [52]. Inspired by the success of transformers in NLP, Dosovitskiy et al. [13] introduced the transformer method for image classification, known as vision transformer (ViT). In this research work, we propose to use a transformer-based approach instead of using CNN models to improve the computational efficiency and accuracy. In this chapter, ViT is discussed in detail along with an implementation of ViT-based approach for COVID-19 CT-Scan image classification. Since we employ ViT for detection of COVID-19 from CT-Scan images, we call this method as COViT-CT.

3.2 What is Self-Attention?

The attention mechanism was first proposed by Graves et al. with a content-based attention mechanism for neural turing machines [53]. After that, Cheng et al. introduced the self-attention concept with a modified long short-term memory (LSTM) unit [54]. Self-attention or intra-attention is an attention method in which the inputs interact with one another (hence known as self) to compute a representation of the input sequence based on the obtained attention value. Hence, it calculates the attention of all other inputs with respect to one input.

The self-attention mechanism is demonstrated in Figure 3.1. Consider n input sequences $X_1, X_2, \cdots X_n$, each sequence being of size $1 \times d_e$, where d_e is the embedding size. Figure 3.1 represents the steps for computing the self-attention score for the current input X_1. The first step is to obtain the embedding query vectors $q_1, q_2, \cdots, q_n$, key vectors $k_1, k_2, \cdots, k_n$ and value vectors $v_1, v_2, \cdots, v_n$ by multiplying the corresponding input sequences with the query weight matrix W^Q, the key weight matrix W^K and the value weight matrix W^V. Here, the size of trainable weight matrices W^Q, W^K, and W^V are $d_e \times d_q$, $d_e \times d_k$, and $d_e \times d_v$, respectively. Hence, multiplication of the weight matrices with the input sequences will produce the embedding query vectors of the size $1 \times d_q$, the key vectors with the size $1 \times d_k$, and the value vectors with the size $1 \times d_v$. Since all the input sequences have the same embedding size d_e, the query vectors, key vectors, and value vectors are all also of the same size, i.e., $d_q = d_k = d_v$. The next step is to compute the dot product between the query vector (q_1) of the current input (X_1) with the key vectors ($k_1, k_2, \cdots, k_n$) of all the inputs, and then the product obtained is divided by the square root of the dimensions of the key vector ($\sqrt{d_k}$) to produce the output $a_{11}, a_{12}, \cdots, a_{1n}$. Here, we scale the output to prevent the final dot product value from being too large. Hence, this attention mechanism is also known as scaled dot product attention. For small values of d_k, the dimensions of key and query vectors are also small. In this case, the dot product without scaling gives similar results as the dot product with scaling. On the other hand, larger values of d_k result in large magnitudes of the dot products, thus pushing the softmax function into the region where it has small gradients, resulting in the vanishing-gradient problem [52]. To counteract this issue, the dot product is divided by the square root of d_k. Then, the softmax function is applied on dot product outputs to obtain the outputs $\hat{a}_{11}, \hat{a}_{12}, \cdots, \hat{a}_{1n}$. Finally, the softmax outputs are multiplied by the corresponding embedding value vectors $v_1, v_2, \cdots, v_n$ and then added to produce the final attention vector A_1 of size $1 \times d_v$. The overall process of calculating the self-attention score of the input X_1 can be written as

$$\hat{a}_{1i} = softmax(a_{1i}) = softmax(\frac{q_1 \cdot k_i}{\sqrt{d_k}}) \tag{3.1}$$

$$A_1 = Attention(q_1, K, V) = \sum_{i=1}^{n} \hat{a}_{1i} v_i \tag{3.2}$$

where q_1 is the embedding query vector corresponding to input X_1, and K and V respectively represent the key and the value matrices made by the embedding key vectors and value vectors corresponding to all the inputs.

The self-attention mechanism discussed above can also be explained in terms of matrices. Let's consider the input matrix X made by the n input embedding vectors $X_1, X_2, \cdots, X_n$. The size of each input vector is $1 \times d_e$, hence, the size of the input matrix is $n \times d_e$. We can generate the query matrix Q, the key matrix K, and the value matrix V by multiplying the input matrix X (size:$n \times d_e$) with the trainable weight matrices W^Q (size:$d_e \times d_q$), W^K (size:$d_e \times d_k$), and W^V (size:$d_e \times d_v$), respectively. The size of Q, K, and V matrices are $n \times d_q$, $n \times d_k$, and $n \times d_v$, respectively, where, $d_q = d_k = d_v$.

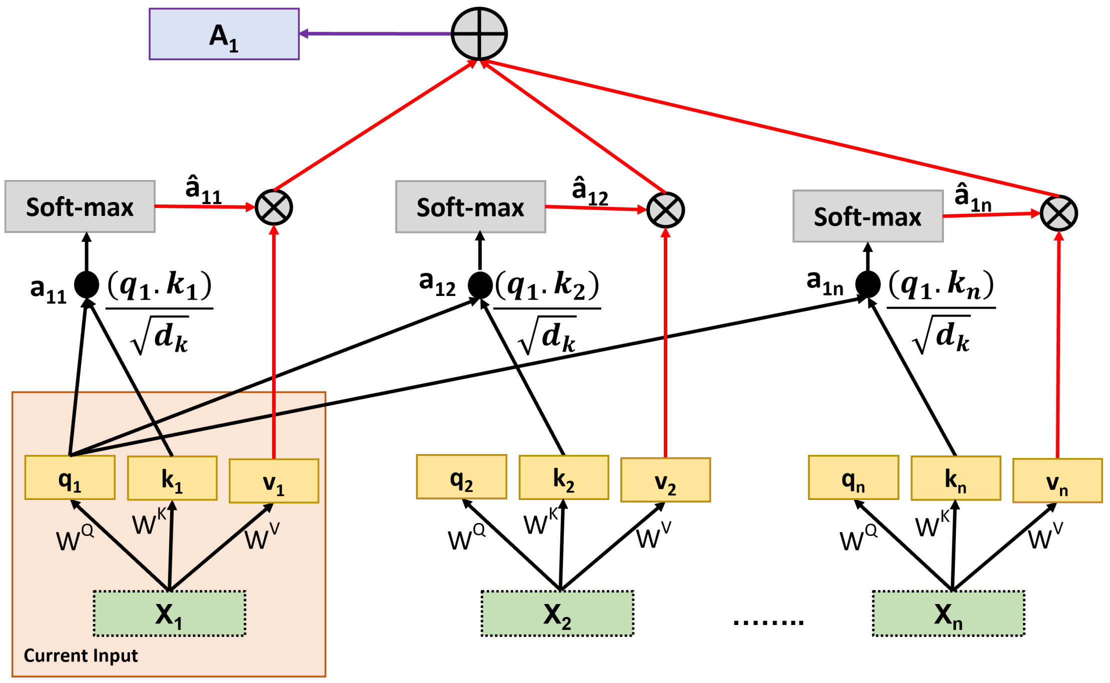


Figure 3.1: Calculation of the self-attention score for the input sequence X$_1$

$$
\begin{aligned}
Q &= X \times W^Q \\
K &= X \times W^K \\
V &= X \times W^V
\end{aligned}
\tag{3.3}
$$

From the above analysis, we can say that the query matrix Q comprises a total of n embedding query vectors $q_1, q_2, \cdots, q_n$, where the size of each vector is $1 \times d_q$. Similarly, matrices K and V contain the total of n embedding key vectors and value vectors, respectively. The next step is to calculate the attention score by multiplying the Q matrix with the transpose of the K matrix as shown in Figure 3.2. The size of the attention score matrix is $n \times n$. The attention score so obtained is scaled by the factor $\frac{1}{\sqrt{d_k}}$ and then the softmax function is applied. Finally, the obtained attention matrix is multiplied by the value matrix V of size $n \times d_v$ to generate the self-attention output matrix of size $n \times d_v$. The overall process can be expressed by the equation

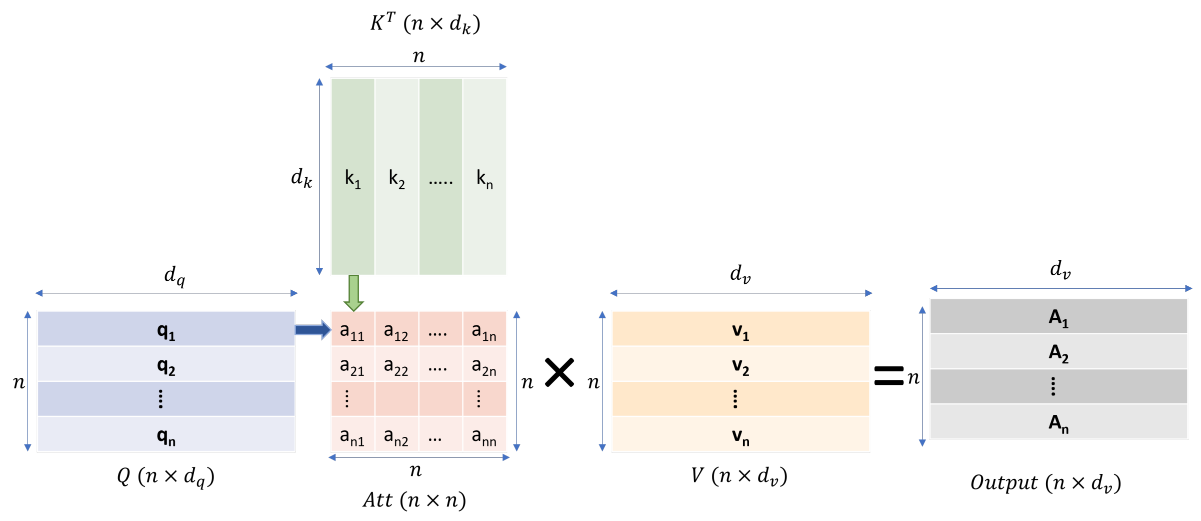


Figure 3.2: The self-attention mechanism as matrix multiplications

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.4}$$

where Q, K, and V represent the query, key, and value vectors, respectively, and d_k denotes the dimension of the key vector.

The original transformer architecture [52] consists of an encoder as well as a decoder. However, in most of the applications, only one block is used based on the task. The encoder is responsible for determining how various patches of the input information relate to one another. The decoder is used to predict the next token in a sequence based on the contextual information provided by the encoder. Both the encoder and the decoder are implemented using the self-attention method. ViT is implemented using just the encoder part of the original encoder-decoder-based transformer architecture to extract the important image features for classification. In the following section, self-attention-based COViT-CT approach is thoroughly explained.

3.3 Self-Attention Based COViT-CT Approach

The COViT-CT method is inspired by the Vision Transformer (ViT) approach [13]. In this method, a lung CT-Scan image data-set is used to classify the input images into two classes, namely, COVID and Non-COVID. Figure 3.3 represents the overview of the COViT-CT network. The ViT architecture is divided into three stages. The first stage is patch and potion embedding to convert 2D image patches into 1D vectors, the second stage consists of the transformer encoder which is responsible for the feature extraction and the last stage is the classification head that classifies the input image.

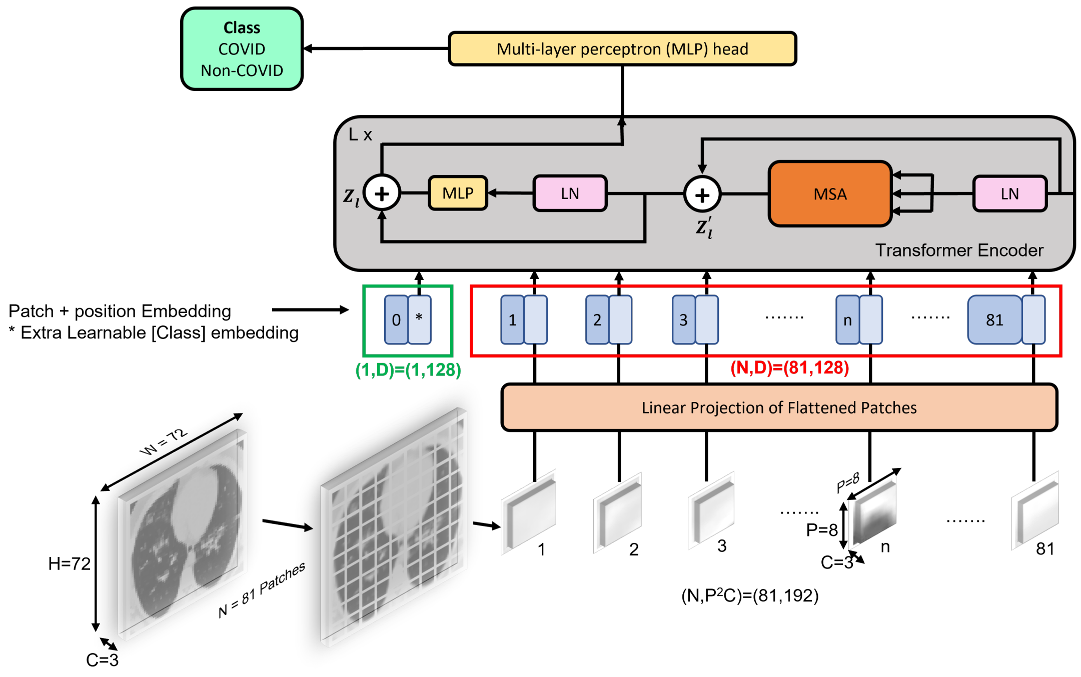


Figure 3.3: Model overview of the COViT-CT approach

3.3.1 Patch and Position Embedding

Generally, the transformer accepts a 1D sequence as an input. Hence, it is required to convert the 2D images into 1D sequences. For that, the input image X having a size $H \times W \times C$ is split into a total of N square patches $X_P^1, X_P^2, \cdots, X_P^N$, where H, W, and

C represent the height, width, and channels of the input image. The size of each patch is $P \times P \times C$, where P is a predefined parameter that denotes the patch size and C is the number of channels associated with each patch. The total number of image patches can be determined by the equation $N = HW/P^2$. Then, all the patches are flattened, resulting in N vectors, each having the size $1 \times P^2C$. The flattened patches are then multiplied with a trainable embedding matrix of size $P^2C \times D$, which linearly projects each flat patch to a dimension D. The dimension D is constant in the architecture, and this step will create a total of N embedded patches of size $1 \times D$.

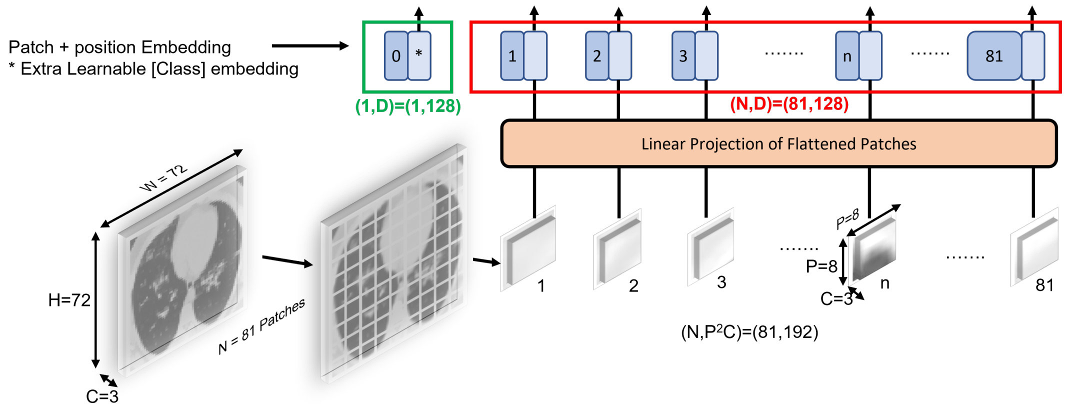

Patch + position Embedding
* Extra Learnable [Class] embedding
0
*
1
2
3
n
81
(1,D)=(1,128)
(N,D)=(81,128)
Linear Projection of Flattened Patches
W = 72
H=72
C=3
N = 81 Patches
P=8
P=8
C=3
(N,P²C)=(81,192)

Figure 3.4: Patch and position embedding

After the patch embedding, the next step is to apply the position embedding to the sequence of embedding. Positional embedding is used to add the spatial representation of each patch within the sequence. Here, a 1-dimensional positional embedding approach is used, wherein the sequence of patches are considered in the raster order (i.e., left to right, up to down). Along with the position encoding, an extra learnable class token of size $1 \times D$ is prepended to the sequence of patch embedding. The first box with an asterisk mark in Figure 3.4 indicates the learnable class vector X_{class} to store the label corresponding to a specific class. The position encoding is essential, in view of the fact that if we change the position of the patches for the model without position encoding, then the transformer will detect both of the images as the same image. The output of the patch and position embedding is given by

$$Z_0 = [X_{class}; X_P^1 E; X_P^2 E; \cdots; X_P^N E] + E_{pos}, \qquad E \in \mathbb{R}^{(P^2 \cdot C) \times D}, E_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{3.5}$$

where X_{class} denotes an extra learnable class token, $X_P^1, X_P^2, \cdots, X_P^N$ represent flatten patches each having the size $1 \times P^2C$, E is a trainable embedding matrix of size $P^2C \times D$, and E_{pos} represents $(N+1)$ positional embedding vectors, each having the size of $1 \times D$.

3.3.2 Transformer Encoder

As discussed in the previous section, the input of the transformer encoder is flattened image patches with position information. The stack of L transformer encoders are responsible for extracting the image features for image classification. The transformer encoder consists of alternative blocks of Multi-Headed Self Attention (MSA) and Multilayer Perceptron (MLP) layers in which layer normalization (LN) is applied before every block, as shown in Figure 3.5. It also consists of residual connections after every block [55].

MSA is implemented using multiple layers of self-attention also known as scaled dot-product attention modules in parallel. Figure 3.6 represents the multiheaded self-attention mechanism with h heads. Before computing the self-attention score, it is required to calculate the query, key, and value matrices through a linear transformation using different learnable weight matrices (i.e., W^Q, W^K, W^V) for each head. Then, self-attention or scaled dot-product attention is computed for each head independently using Equation (3.4). As discussed in Section 3.2, the size of the output matrix generated using a self-attention mechanism applied on $(N+1)$ patches is $(N+1) \times d_v$. The output produced by each self-attention block is concatenated to generate the final output matrix of size $(N+1) \times d_v \cdot h$, where $(N+1)$ represents the number of input sequences plus learnable class token, d_v is

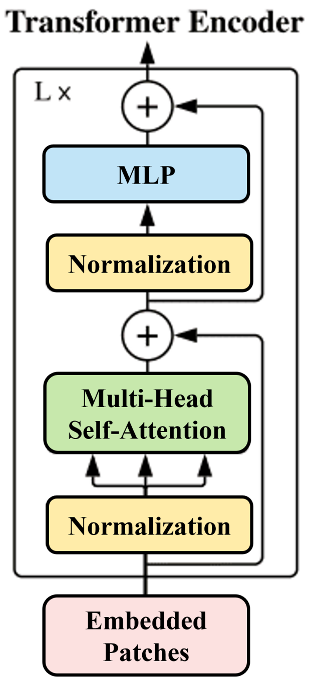


Figure 3.5: Vision transformer encoder (adapted from [13])

the dimension of the value vector and h is the number of heads. Finally, the concatenated output is again linearly projected using the MSA weight matrix ($\mathrm{U}_{\mathrm{msa}}$) of size $h \cdot d_v \times d_o$, where d_o is output dimension. Hence, the size of MSA output matrix is $(N+1) \times d_o$. For h heads the equation of the MSA is given by

$$MSA(Z) = [SA_1(Z); SA_2(z); \cdots; SA_h(Z)]\mathrm{U}_{\mathrm{msa}} \tag{3.6}$$

where $SA_i(Z)$ denotes the self-attention output generated by the i^{th} head.

After the MSA block, there is a 2-layer MLP block (hidden and output). MLP is a feed-forward neural network in which every layer is a fully connected layer. In the transformer encoder, layer normalization is applied before the MLP and MSA blocks help to improve the training time and overall performance of the network [55]. Layer normalization is similar to batch normalization as discussed in Section 2.3.4; however, it calculates the mean and standard deviation over all the hidden units in the same layer instead of mini-batch

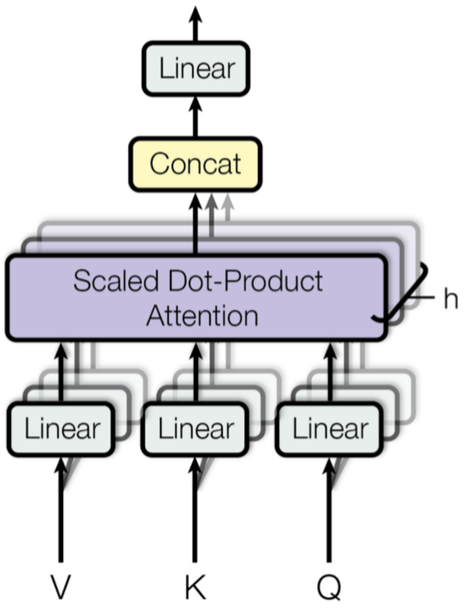


Figure 3.6: Multiheaded self-attention (adapted from [52])

[56]. The residual connections are also applied after the MLP and MSA blocks to solve the vanishing gradients problem in very deep architecture. The output $Z_l^{'}$ obtained after the MSA block is given by Equation (3.7) and the output of the transformer encoder is given by Equation (3.8).

$$Z_l^{'} = MSA(LN(Z_{l-1})) + Z_{l-1}, \qquad l = 1 \ldots L \tag{3.7}$$

$$Z_l = MLP(LN(Z_l^{'})) + Z_l^{'}, \qquad l = 1 \ldots L \tag{3.8}$$

3.3.3 Classification Head

The last component of the vision transformer is the classification head that contains a 2-layer MLP as shown in Figure 3.7. The output of the transformer head is fed to the MLP head for the final classification. The MLP head uses the Gaussian Error Linear Unit (GELU) as the activation function, which is defined as [57]

$$\begin{aligned}\mathrm{GELU}(x) = xP(X \leq x) &= x \cdot \frac{1}{2}[1 + \mathrm{erf}(x/\sqrt{2})] \\ &\approx 0.5x(1 + \tanh[\sqrt{2/\pi}(x + 0.044715x^3)]) \\ &\approx x\sigma(1.702x)\end{aligned} \tag{3.9}$$

where $\mathrm{erf}(x)$ represents the Gaussian error function of x given by $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt$, and $\sigma(x)$ is the sigmoid function of x given by Equation (2.4).

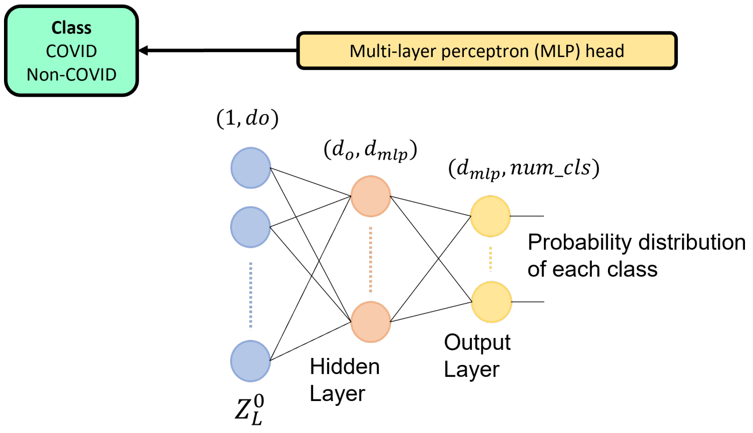


Figure 3.7: Classification head of the COViT-CT network

As discussed in Section 3.3.2, the size of the output of the transformer encoder is $(N+1) \times d_o$. From this output matrix, only the first representation of the class token is used for the classification with size $1 \times d_o$. Here, a 2-layer MLP is used for pre-training with weight matrices of size $d_o \times d_{mlp}$ and $d_{mlp} \times num\_cls$ for the hidden layer and the output layer, respectively, where num_cls represents the number of classes used for classification. Hence, the final output of the network is a vector of size $1 \times num\_cls$, which comprises the probabilities associated with each of the num_cls classes. The final classification output y is given by

$$y = MLP(LN(Z_L^0)) \tag{3.10}$$

where Z_L^0 represents the first output vector representing the class token.

3.4 Implementation of Self-Attention Based COViT-CT Approach

As discussed in Section 3.3.1, the first step is to convert the input color image into serialized data. As shown in Figure 3.3, the input lung CT-Scan image is resized to $72 \times 72 \times 3$, where height $H = 72$, width $W = 72$, and channel $C = 3$ due to the red, green, and blue color channels. We consider the patch size to be $P = 8$, and hence, the total number of patches is $N = HW/P^2 = (72 \times 72)/8^2 = 81$. So, we have a total of 81 image patches each having the size $8 \times 8 \times 3$ as the input to the COViT-CT network. Then, each patch is converted into a 1-D sequence of size $1 \times P^2C = 1 \times 192$. Since we take the embedding dimension $D = 128$, each flattened patch is then multiplied with a trainable embedding matrix of size $P^2C \times D = 192 \times 128$ to generate a total of 81 patches of size 1×128. During the next step, the positional information is attached to all the patches along with an extra learnable class token, which results in a matrix of size $(N + 1) \times D = 82 \times 128$. We can say that the input of the transformer encoder is a total of 82 patches each having the size 1×128. During the implementation of the transformer encoder, we have set $L = 8$, i.e., 8 stacked transformer encoders each comprising $h = 6$ MSA heads. The transformer encoder will generate the image features having the size $82 \times d_o$, where $d_o = d_v \cdot h = 128 \times 6 = 768$. The first representation of the class token having the size of 1×768 is then used for the classification using MLP. So, the input image is classified into two classes, namely, COVID and Non-COVID based on the 768 extracted features.

3.5 Summary

In this chapter, the mechanism of the self-attention method is described in detail. Further, the implementation of the proposed COViT-CT network is explained briefly along with the patch and position embedding, the transformer encoder, and the classification head. In the next chapter, the results obtained through various CNN models as described in Chapter 2, as well as with the self-attention-based deep learning approach COViT-CT, will be discussed and compared using standard performance evaluation metrics.

Chapter 4

Performance of the Proposed COViT-CT Method and Comparison with that of Other Deep Learning Techniques

This chapter presents the experimental setup for implementing various deep learning approaches for COVID-19 screening, and the results are analyzed and compared based on standard comparison metrics. The chapter is organized as follows. First, the description of the COVID-19 CT-Scan dataset used for the experiments is presented. Then, the system environment used for the experiments is discussed, along with the standard performance evaluation metrics used to compare the results. Then, the results obtained by the various CNN models and the self-attention-based COViT-CT approach are presented. Finally, the results obtained through various implemented techniques are compared with the recent state-of-art works.

4.1 Dataset

In this work, a publicly available COVID-19 CT-Scan image dataset known as the SARS-CoV-2 CT-scan dataset provided by Soares et al. [32] has been used for analysis. This dataset contains 2481 images comprising 1252 COVID-19 positive and 1229 COVID-19 negative images. The whole dataset is split into training, testing, and validation in the ratio $60\% : 20\% : 20\%$. Table 4.1 describes the detailed information regarding splitting the dataset into training, testing, and validation. Moreover, Figure 4.1 represents sample images from both classes. In this work, the models are evaluated through a statistical analysis based on accuracy, precision, recall, F1-Score, and area under the curve (AUC). Here, the transfer learning approach, discussed in Section 2.4.1, is used for image classification through various CNN models.

Table 4.1: Training-Validation-Testing data splitting

Data Split	**COVID**	**Non-COVID**	**Cumulative**
Traning	752	737	1489
Validation	250	246	496
Testing	250	246	496
Total			2481

4.2 Experimental Set-up and Performance Evaluation Metrics

All the experiments were performed on a system with an Intel(R) Core(TM) i7-11390H processor having a maximum clock speed of 3.4 GHz, 16 GB RAM, and a Windows 11 operating system. In addition, all the four CNN methods described in Section 2.4, namely, ResNet50, DarkNet19, GoogLeNet and VGG16, are implemented using a MATLAB R2021a platform, and the self-attention-based technique COViT-CT discussed in

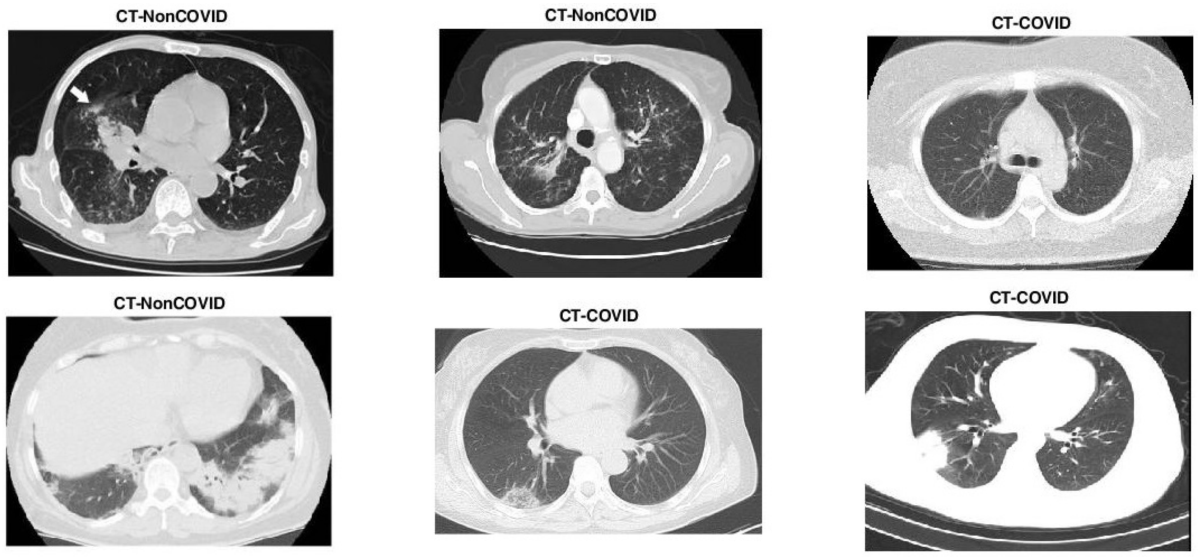


Figure 4.1: Sample CT-Scan images from the dataset which contains the images from both the classes

Chapter 3 is implemented using Keras, which is an open-source Python library for creating deep learning applications.

The performance of all these methods, including the four CNN models and the COViT-CT method, are evaluated based on six evaluation metrics, which are accuracy, precision, recall (sensitivity), specificity, F1-Score, and AUC. Moreover, the corresponding confusion matrix for each model is also represented.

The confusion matrix gives an output in the form of a matrix that describes the complete performance of the model. We used the softmax layer as a binary classifier for all four CNN models that classifies the images into two classes, namely, CT-COVID and CT-NonCOVID. The confusion matrix is used to compare the predicted classes of our classifier with the actual classes. Based on this, there are four important terms, namely, true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In this case, all the cases with the prediction CT-COVID which are actually COVID-19 positive, are considered as TP. The cases with the classifier prediction CT-NonCOVID and their actual class is also CT-NonCOVID are considered as TN. However, if the predicted class is CT-COVID and they are actually CT-NonCOVID are FP cases and also known as type-I error. Moreover, type-II error or FN contains all the cases which actually COVID-19 positive, but classified as CT-NonCOVID.

Accuracy(ACC) of the model represents the correctly classified observations over the total number of observations and is given by

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Precision(PC), also known as a positive predictive value, is the ratio that indicates the positive responses that are correctly classified. It is used to determine how precise or accurate the model is. Precision is defined as

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

Recall is also known as sensitivity(SN), or the true positive rate is a measure of the probability, representing the actual positive observations that are classified as positive. Recall is given by

$$Sensitivity(Recall) = \frac{TP}{TP + FN} \tag{4.3}$$

Specificity(SP), also known as the true negative rate, indicates how often the actual negative observations will be classified as negative. The equation of specificity is

$$Specificity = \frac{TN}{TN + FP} \tag{4.4}$$

F1-Score is represented as the harmonic mean of the precision and recall, which varies from 0 to 1. This metric is used to evaluate the preciseness and robustness of the classifier. The higher value of the F1-Score means our model performance is also good. F1-Score is given by

$$\frac{1}{F1\text{-}Score} = \left(\frac{1}{2}\right)\left[\frac{1}{Precision} + \frac{1}{Recall}\right]$$

Or

$$F1\text{-}Score = 2 \times \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{4.5}$$

The above equation can be further simplified by substituting the values of $Precision$ and $Recall$ from the equations (4.2) and (4.3), respectively,

$$F1\text{-}Score = \frac{2TP}{2TP + FP + FN} \tag{4.6}$$

AUC is generally used to evaluate the performance of binary classifiers. It calculates the area under the ROC curve. ROC is the curve with the false positive rate on the x-Axis and the true positive rate on the y-Axis. As discussed above, the true positive rate is also known as recall or sensitivity and is given by Equation (4.3). Moreover, the false positive rate is defined as

$$FalsePositiveRate = \frac{FP}{TN + FP} \tag{4.7}$$

Here, values of both the axes vary from 0 to 1. Both the true positive rate and the false positive rate are computed for different threshold values to plot the AUC curve. The range of AUC is $[0, 1]$, and higher the value of AUC, better is the performance of the model.

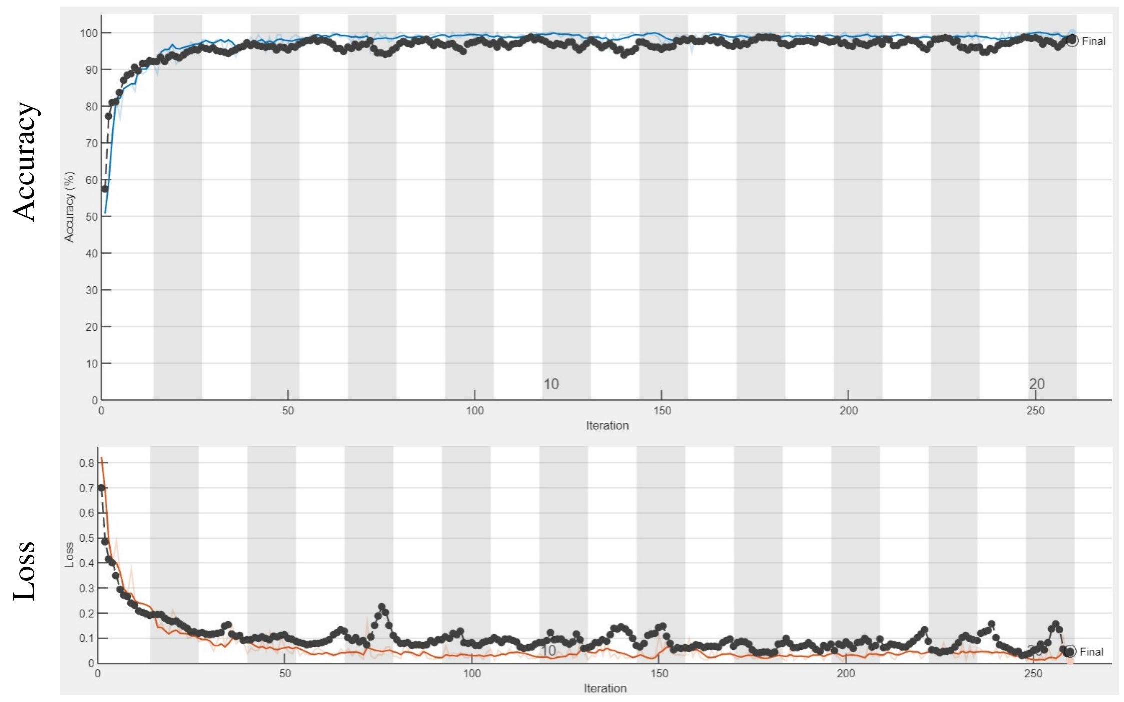
4.3 Performance Results using the Four CNN Methods ResNet50, DarkNet19, GoogLeNet and VGG16

As discussed in Section 4.2, all the four CNN methods are implemented using MATLAB R2021a. In this experiment, we use “Adam” as the optimizer [58] with an initial learning rate value of 10^{-4}, maximum epoch size 20, and minimum batch size 115 for all the four CNN models. This will result in 13 iterations per epoch as we have 1489 training images and 115 images per batch, i.e., $1489 \div 115 = 12.95 \approx 13$. As there are 20 epochs, the total number of iterations is $20 \times 13 = 260$. Other training options used in this work are data shuffling before each epoch and an L2-regularization with a weight decay value of 0.005 to overcome the problem of over-fitting by forcing the weights to be small, but not exactly zero. Moreover, we assign a validation dataset as described in Table 4.1 and set the validation frequency to 1.

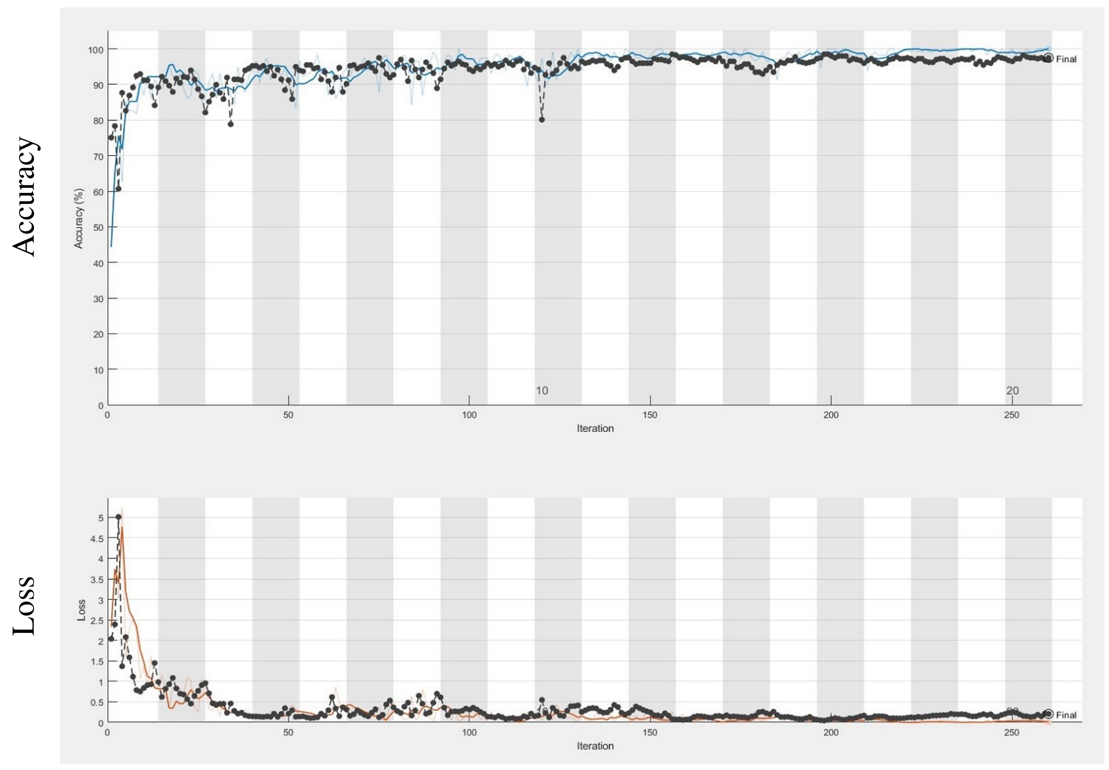
Figure 4.2 represents the training progress chart of the four implemented CNN methods, namely, ResNet50, DarkNet19, GoogLeNet and VGG16. For each technique, the training progress comprises the accuracy vs. iteration and loss vs. iteration graphs for training (blue and red color lines indicate the accuracy and loss, respectively) as well as validation (black dotted line indicates the validation data).

Figure 4.3 represents the confusion matrices of the four CNN methods. It is seen that ResNet50 gives the best results with an efficiency of 98.6%. Moreover, the ROC curves of all the methods are represented in Figure 4.4 to compare their performance based on the AUC parameter. It is observed that ResNet50 has the highest value of AUC, which is 0.9992, followed by DarkNet19 with an AUC value of 0.9981, and VGG16 with an AUC value of 0.9975. However, the performance of GoogLeNet is the worst based on the AUC value, which is 0.9856.

Moreover, Figure 4.5 represents the activated features from the layer “activation_2_relu",

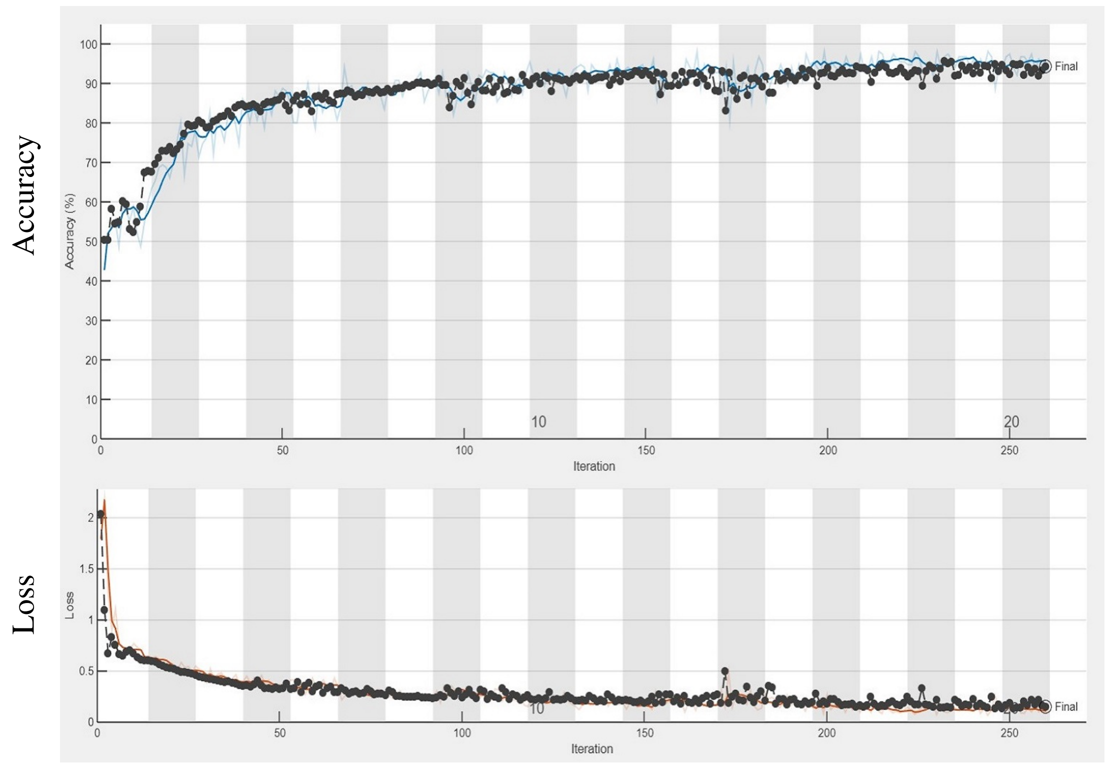


(a) ResNet50

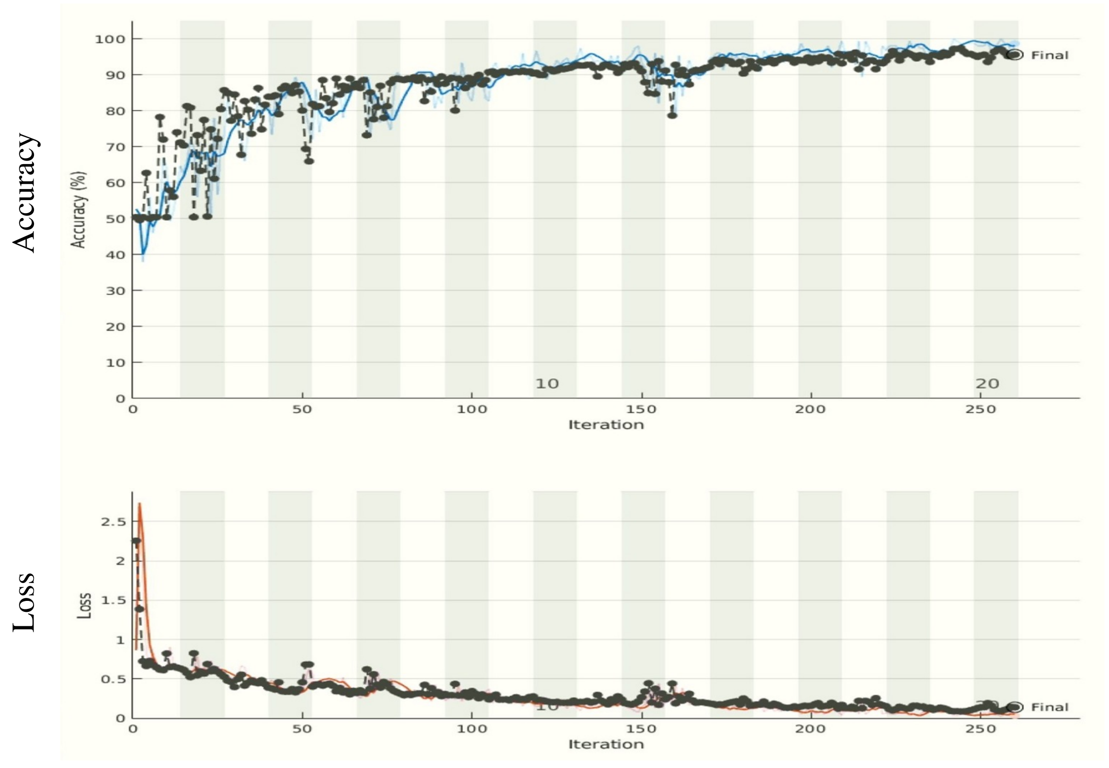


(b) DarkNet19

Figure 4.2: Training progress of the various CNN methods



(c) GoogLeNet



(d) VGG16

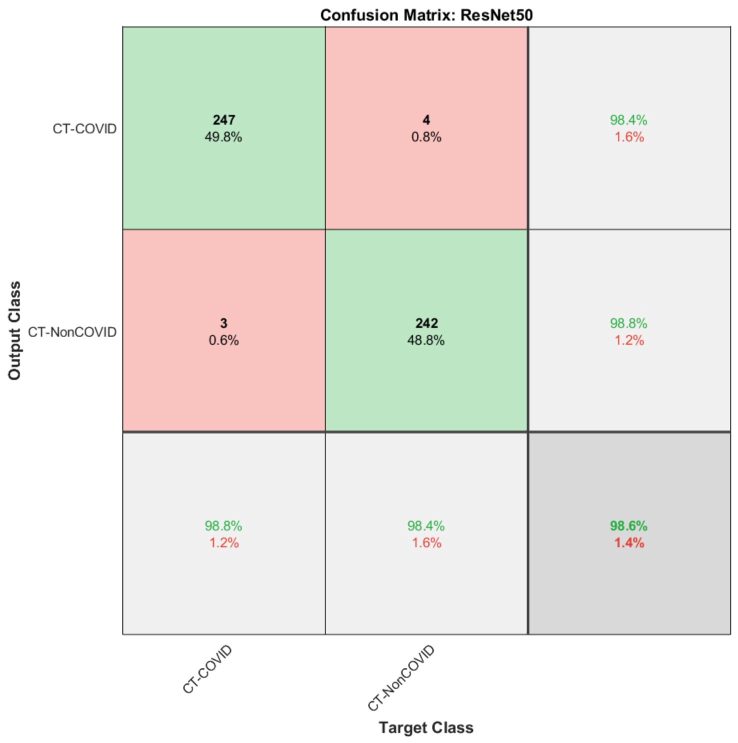
Figure 4.2: Training progress of the various CNN methods (cont.)

TP: 247

TN: 242

FP: 4

FN: 3



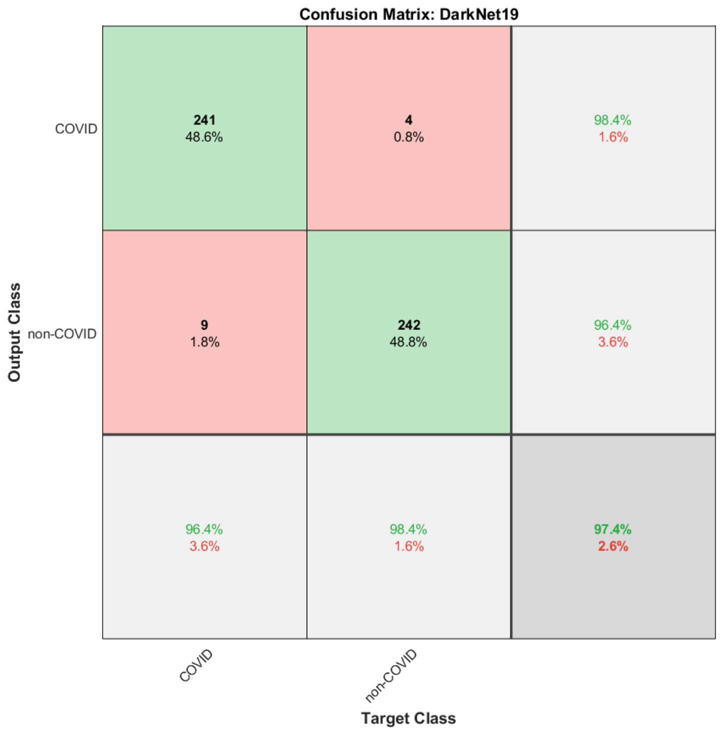
(a) ResNet50

TP: 241

TN: 242

FP: 4

FN: 9



(b) DarkNet19

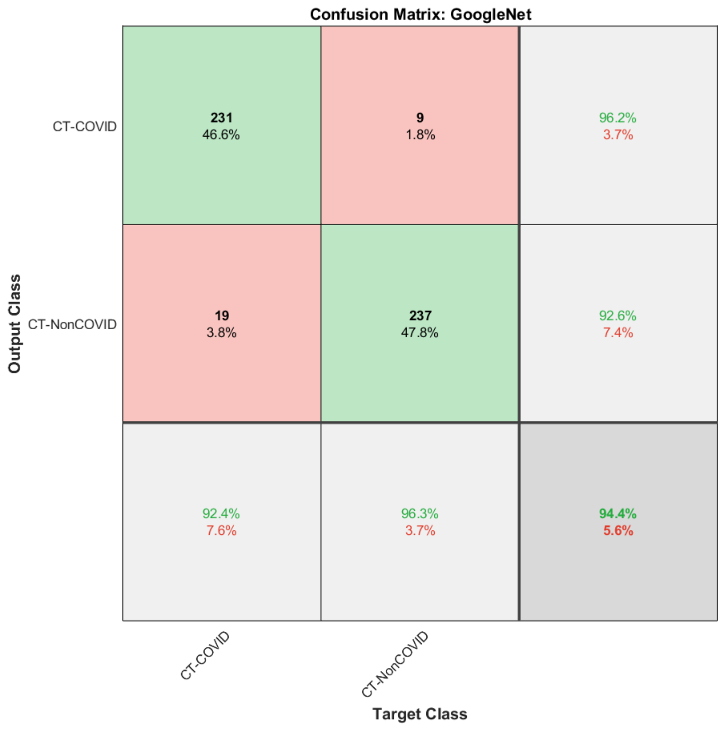
Figure 4.3: Confusion matrices of the various CNN methods

TP: 231

TN: 237

FP: 9

FN: 19



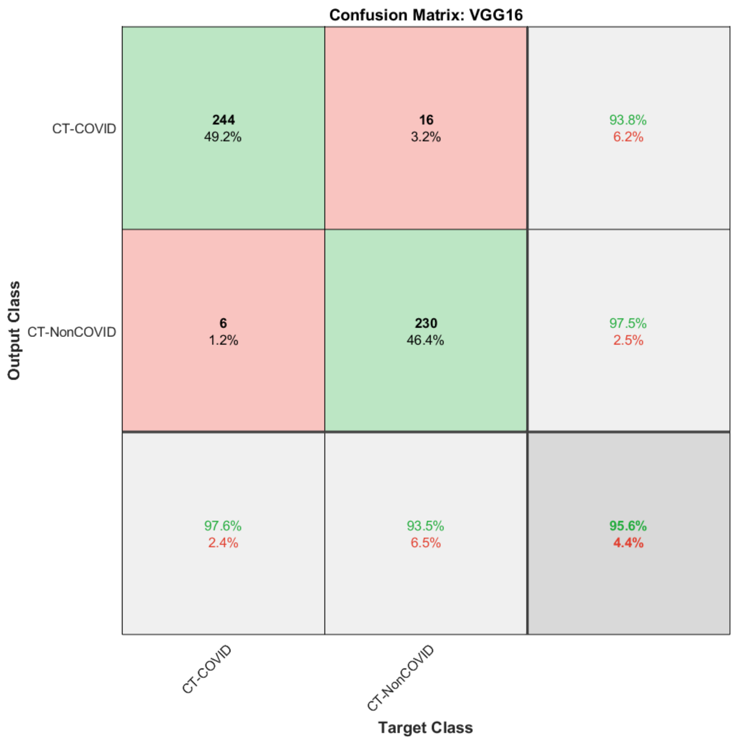
(c) GoogLeNet

TP: 244

TN: 230

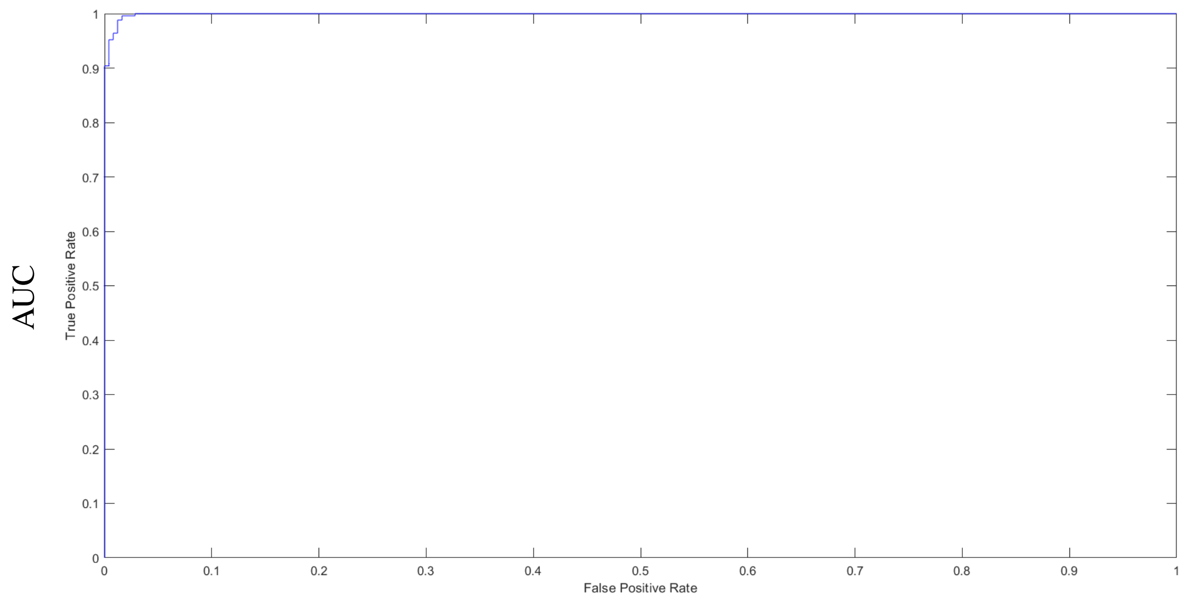
FP: 16

FN: 6

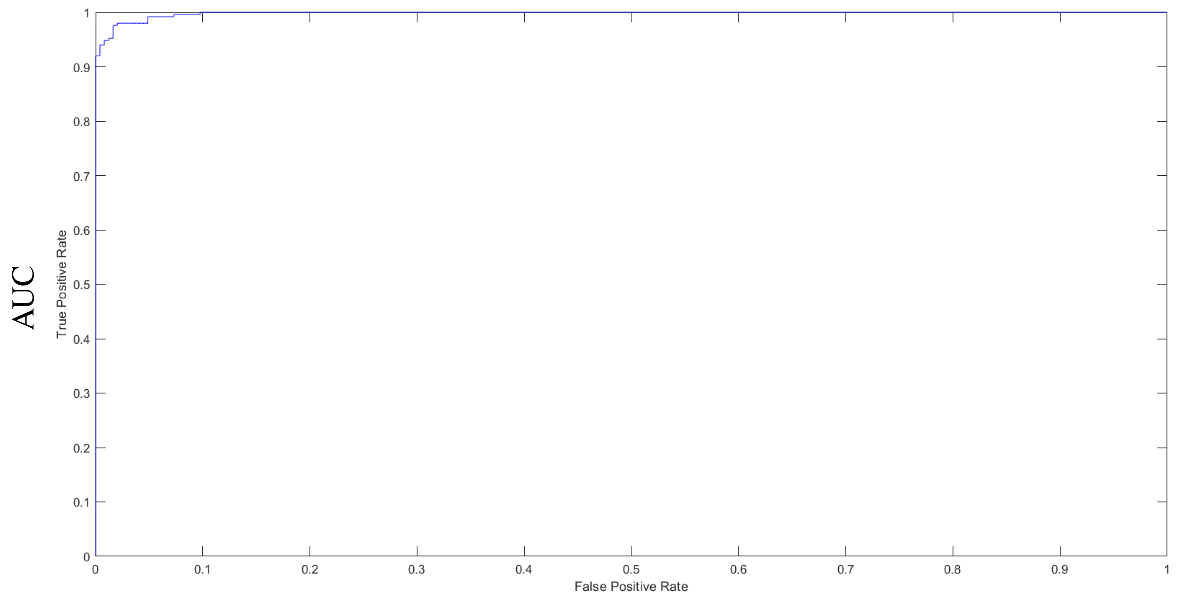


(d) VGG16

Figure 4.3: Confusion matrices of the various CNN methods (cont.)

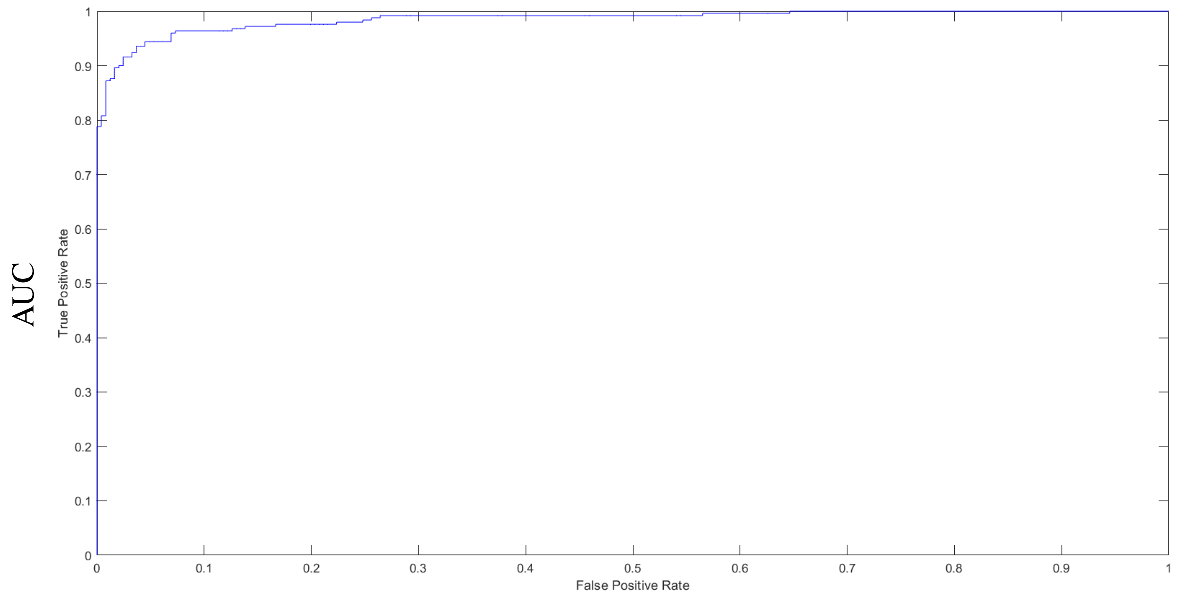


(a) ResNet50

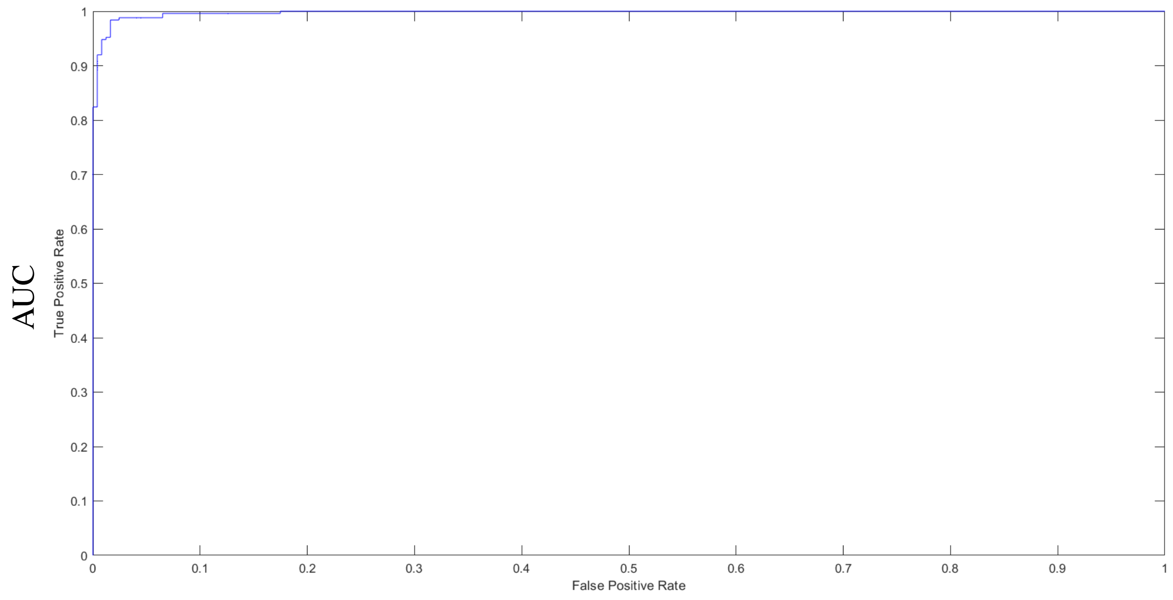


(b) DarkNet19

Figure 4.4: ROC curves for implemented CNN methods



(c) GoogLeNet



(d) VGG16

Figure 4.4: ROC curves for implemented CNN methods (cont.)

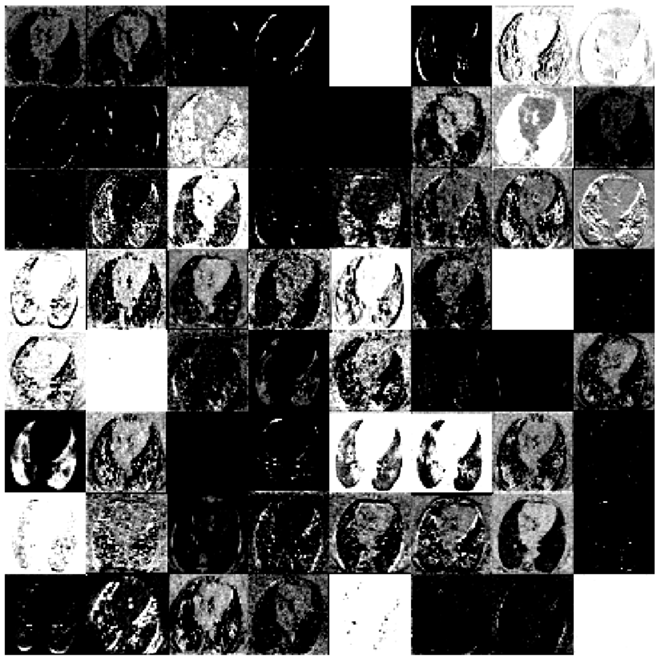


Figure 4.5: Activated features of activation_2_relu layer of ResNet50

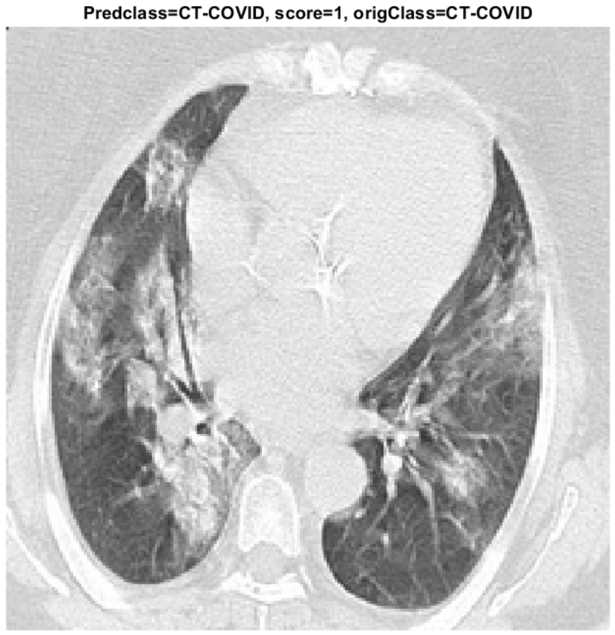
which is the second linear rectified unit layer of ResNet50. These image features are extracted from a random image Figure 4.6(a) of the testing image data set. In Figure 4.5, dark colors represent the less activated features, whereas highly activated features are with bright colors. As discussed in Section 2.4.2, the second RELU layer of ResNet50 contains 64 various filters; hence, we have 64 various activated features. Similarly, we can visualize the various activated features from different layers of all the implemented CNN networks. The total number of features depends on the total number of filters used in the corresponding layer. Figure 4.6 represents two random images from the testing image data set with the predicted class and the original class of these random images along with the prediction score. Figure 4.6(a) represents the true positive case for which the predicted class and original class both are CT-COVID. In this case, the score vector, i.e., the output of the softmax layer is $[1, 5.0066 \times 10^{-7}]$ which indicates the probability score 1 for the CT-COVID as the predicted class and the probability score $5.0066 \times 10^{-7} \approx 0$ for the CT-NonCOVID as the predicted class. Hence, the image is classified as CT-COVID. However, Figure 4.6(b) represents the false negative case (type-II error) for which the original CT-COVID image is classified as CT-NonCOVID. Here, the probability score by the softmax layer is [CT-COVID, CT-NonCOVID] $= [0.3910, 0.6090]$. Hence, the image is classified as CT-NonCOVID as the probability score for that class is higher than that of CT-COVID.

4.4 Performance Results using the Proposed Self-Attention Based COViT-CT Approach

The self-attention-based COViT-CT method is implemented using the Keras library in Python. The ViT for COVID-19 detection from CT-Scan images used the 'Adam' optimizer [58] having a learning rate and weight decay values of 0.0002. Here, we set the batch size to 150 and total epochs to 700. The input image is resized to 72×72. As discussed in Section 3.4, the input image is first divided into 81 patches with the size of each patch being $8 \times 8 \times 3$. The sample input image from the training dataset and its patches are presented in Figure 4.7(a) and Figure 4.7(b), respectively. In this work, we set the projection dimension D to be 128 during patch and position embedding. Hence, all the 81 input patches are projected into a vector of size 128. Moreover, we employ six multi-head attention units and eight transformer blocks. The training process chart, i.e., 'Accuracy Vs. Epoch' and 'Loss Vs. Epoch' chart, along with the confusion matrix and ROC curve of the results obtained using the above-mentioned parameters are presented in Figure 4.8 and Figure 4.9, respectively.

4.5 Comparison of Results

In this section, we will compare the results of the COViT-CT approach discussed in Section 4.4 with not only the four different CNN methods as discussed in Section 4.3, but also

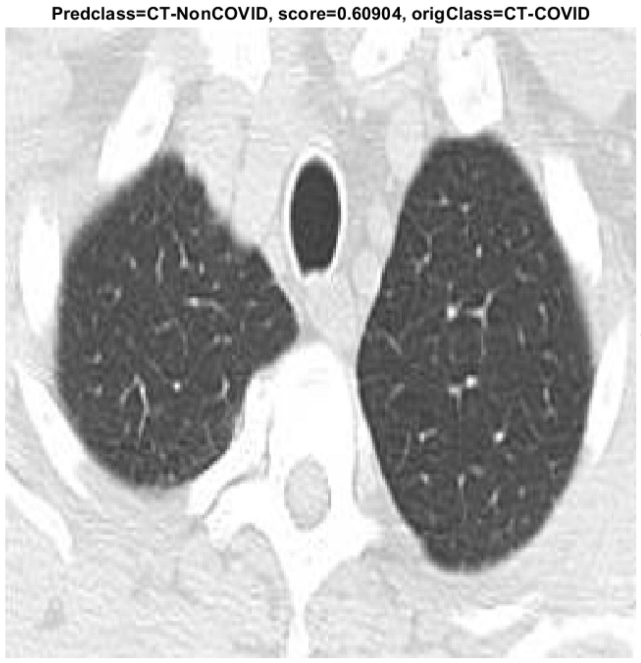


Predicted Class: CT-COVID

Score: $[1, 5.0066 \times 10^{-7}]$

Original Class: CT-COVID

(a) True positive case



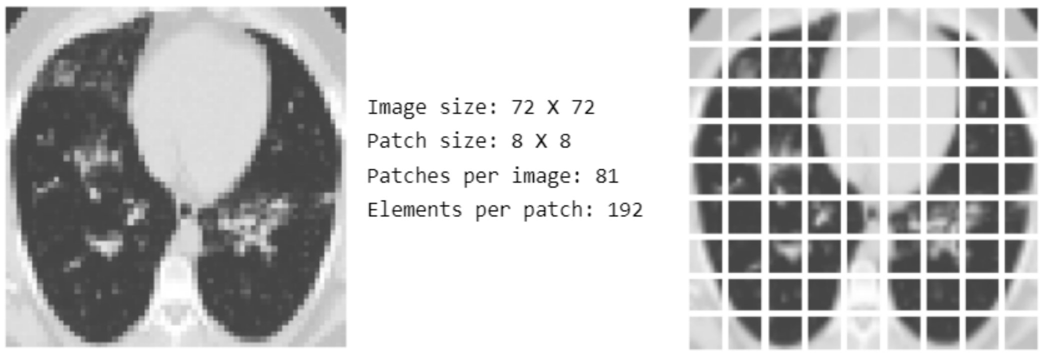
Predicted Class: CT-NonCOVID

Score: $[0.3910, 0.6090]$

Original Class: CT-COVID

(b) False negative case (Type-II error)

Figure 4.6: Random output images with predicted class, prediction score, and original class from the testing image dataset



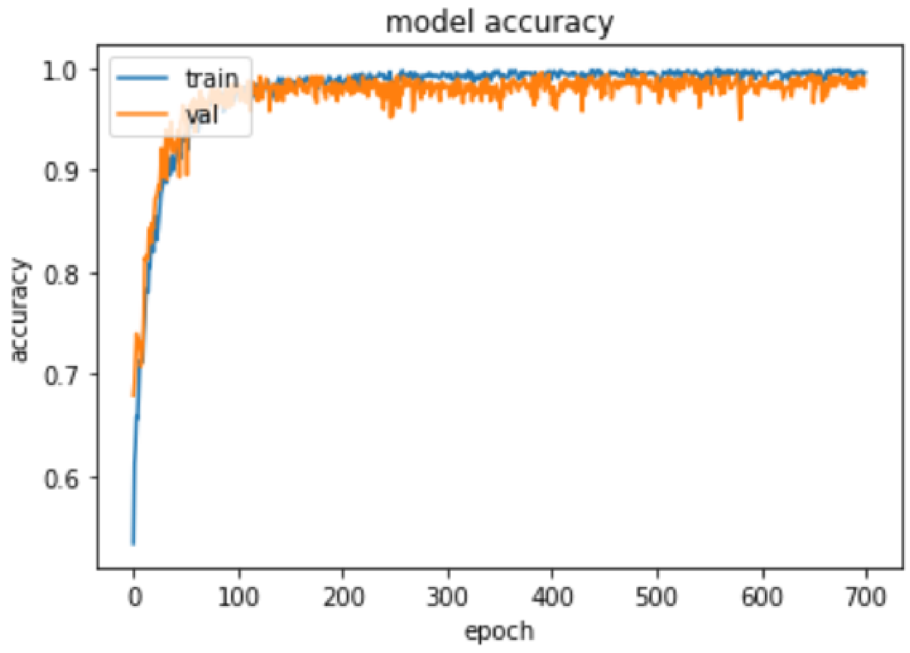
(a) (b)

Figure 4.7: (a) Sample image from the training image dataset (b) Input image divided into 81 patches

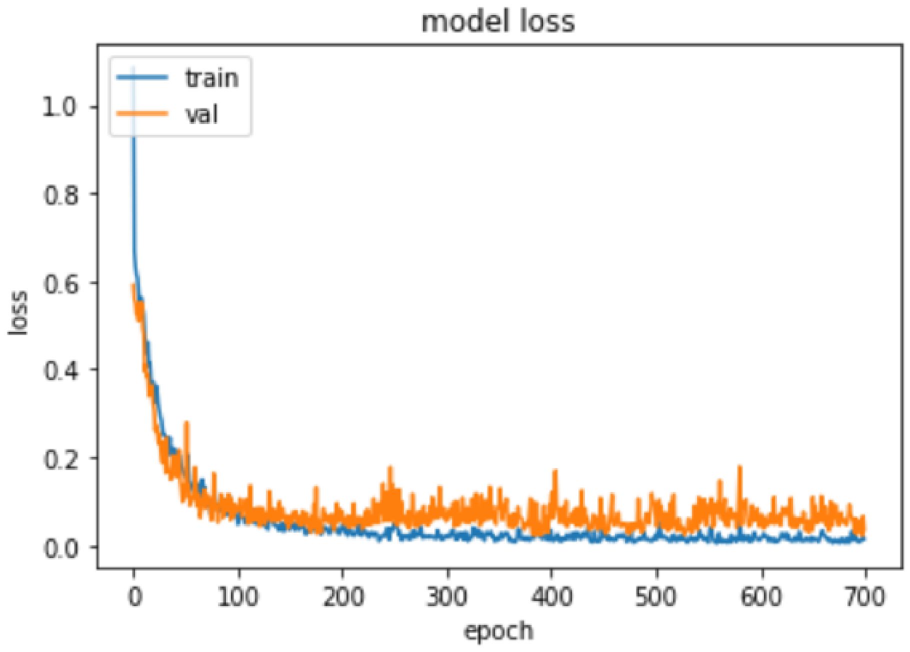
with other recent state-of-the-art methods. All the comparisons are based on the performance evaluation metrics as discussed in Section 4.2.

The results obtained by classification via the four implemented CNN methods, namely, VGG16, GoogLeNet, ResNet50 and DarkNet19, and the proposed COViT-CT method are shown in red fonts in Table 4.2. From the analysis of these entries, it is observed that the best results are attained by the self-attention-based COViT-CT model with a precision of 99.62%, sensitivity (recall) of 98.86%, specificity of 99.57%, F1-score of 99.24%, and validation accuracy of 99.20%. If we consider only the four CNN models, then ResNet50 gives the best results for all the metrics. Moreover, it can be observed by comparing the confusion matrices presented in Figure 4.3 and Figure 4.9(a) that the COViT-CT method achieves the smallest misclassification error followed by ResNet50. From Figure 4.2 and Figure 4.8, it can be seen from the accuracy/loss Vs. epoch plot that the validation line closely follows the training line for the best performing models, namely, COViT-CT and ResNet50.

We now compare the performance of the COViT-CT approach with that of the various state-of-the-art architectures, which have been implemented using the same SARS-CoV-2 CT-scan dataset. The results for these architectures are shown in black fonts in Table 4.2,

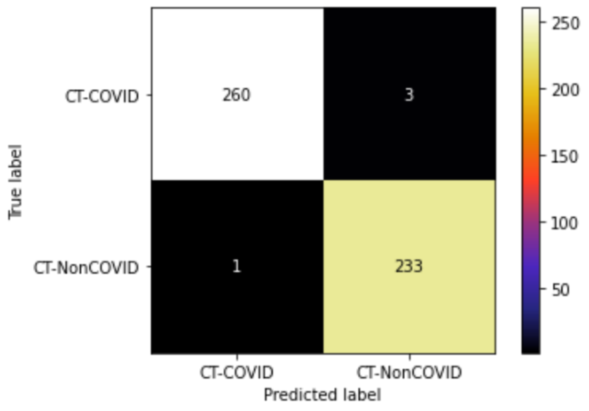


(a) Accuracy Vs. Epoch

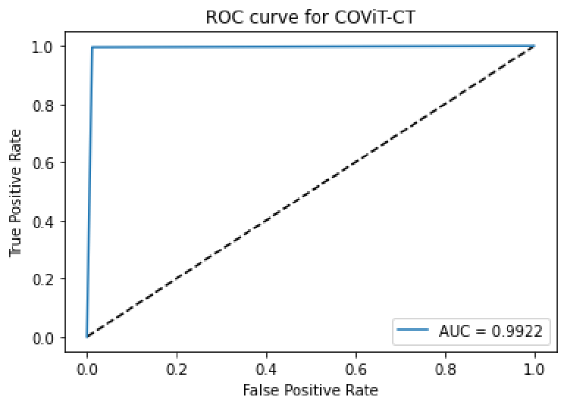


(b) Loss Vs. Epoch

Figure 4.8: Training progress chart of the COViT-CT approach



(a)



(b)

Figure 4.9: (a) Confusion matrix for the COViT-CT approach (b) ROC curve for the CoViT-CT approach

Table 4.2: Comparison of the performance of the various implemented deep learning approaches

Metrics / Methods	Accuracy	Precision	Recall (Sensitivity)	Specificity	F1-Score	AUC
VGG16	95.56%	93.85%	97.60%	93.50%	95.69%	0.9975
GoogLeNet	94.35%	96.25%	92.40%	96.34%	94.29%	0.9856
ResNet50	98.59%	98.41%	98.80%	98.37%	98.60%	0.9992
DarkNet19	97.38%	98.37%	96.40%	98.37%	97.37%	0.9981
Decision Tree [32]	79.44%	76.81%	83.13%	-	79.84%	0.7951
DBM [59]	97.23%	98.14%	97.68%	-	97.89%	0.9771
MADE-DBM [59]	98.37%	98.74%	98.87%	-	98.14%	0.9832
AdaBoost [32]	95.16%	93.63%	96.71%	-	95.14%	0.9519
AlexNet [32]	93.75%	94.98%	92.28%	-	93.61%	0.9368
VGG16 [32]	94.96%	94.02%	95.43%	-	94.97%	0.9496
VGG19 [60]	95.00%	95.30%	94.04%	-	94.67%	-
GoogLeNet [32]	91.73%	90.20%	93.50%	-	91.82%	0.9179
Dragonfly algorithm (DA) [61]	98.39%	98.21%	97.78%	-	98.00%	0.9952
ResNet50 [62]	98.35%	98.02%	98.80%	-	98.41%	0.9994
ResNet [32]	94.96%	93.00%	97.15%	-	95.03%	0.9498
ResNet101 [62]	96.71%	96.43%	97.20%	-	96.81%	0.9944
ResNet18 [62]	97.12%	96.83%	97.60%	-	97.21%	0.9973
EfficientNet [63]	98.99%	99.20%	98.80%	-	-	-
xDNN [32]	97.38%	99.16%	95.53%	-	97.31%	0.9736
COViT-CT	99.20%	99.62%	98.86%	99.57%	99.24%	0.9922

and the metric values given for these methods are taken from the respective articles for which the references are indicated in the parentheses. It is seen from Table 4.2 that the proposed self-attention-based COViT-CT method has the highest accuracy of 99.20% amongst all the reported works in the literature. Moreover, the precision and F1-score of the COViT-CT method are also the highest amongst all the state-of-the-art techniques, with the Recall (or sensitivity) score almost the same as the highest score obtained by the MADE-DBM method [59] in which a Memetic Adaptive Differential Evolution (MADE) algorithm is used to tune the hyperparameters of the deep bidirectional long short-term memory network with a mixture density model.

4.6 Summary

In this chapter, we have obtained the performance results for the proposed COViT-CT approach and compared it with that of the various state-of-the-art techniques using the standard comparison metrics. From the analysis of the results, we can conclude that the implemented self-attention-based COViT-CT is the best approach with the highest accuracy of 99.20%, highest precision of 99.62% and highest F1-score of 99.24%. Hence, we choose this method for lung CT-Scan image classification. Based on the classification results, we will decide whether to apply image segmentation or not. If the lung CT-Scan image classified as COVID-19 positive, then the image segmentation method is applied in order to identify the severity level of the disease. The image segmentation approach used in this thesis will be discussed in the next chapter.

Chapter 5

Extraction of COVID-19 Infected Region

In the previous chapters, we discussed various neural network methods for classifying the input lung CT-Scan images into COVID and Non-COVID. After analyzing the results obtained by the various methods in Chapter 4, we selected the COViT-CT method for image classification as it is the best method with the highest accuracy. If the lung CT-Scan image is classified as COVID-19 positive, then the infected lung portion is extracted for further diagnosis. In this chapter, image segmentation-based approach for extracting the infected lung region is discussed in detail. Finally, the severity level of the infection is determined based on the infected region, which helps to reduce the processing time and burden of experienced doctors.

5.1 Introduction

Image segmentation is a process of dividing the digital image into multiple segments or image regions. There is no single standard algorithm for image segmentation. We can select the appropriate segmentation method based on the types of images and applications [64]. In this work, we implement an image segmentation-based approach for extracting the COVID-19 infected region from lung CT-Scan images.

The overall flow diagram of the image segmentation-based approach used to detect the infected COVID-19 region is illustrated in Figure 5.1. As shown in the figure, the method is divided into three main parts, image pre-processing, image segmentation, and computation of the severity level of the infection. The main goal of image pre-processing is to obtain the region of interest (ROI) for image segmentation using three steps, which are (i) removing unnecessary background portions based on user-defined shape, (ii) applying image enhancement through the power-law transformation technique [65], and (iii) finally, applying the Otsu thresholding approach [23] to obtain the ROI, i.e., lung portion.

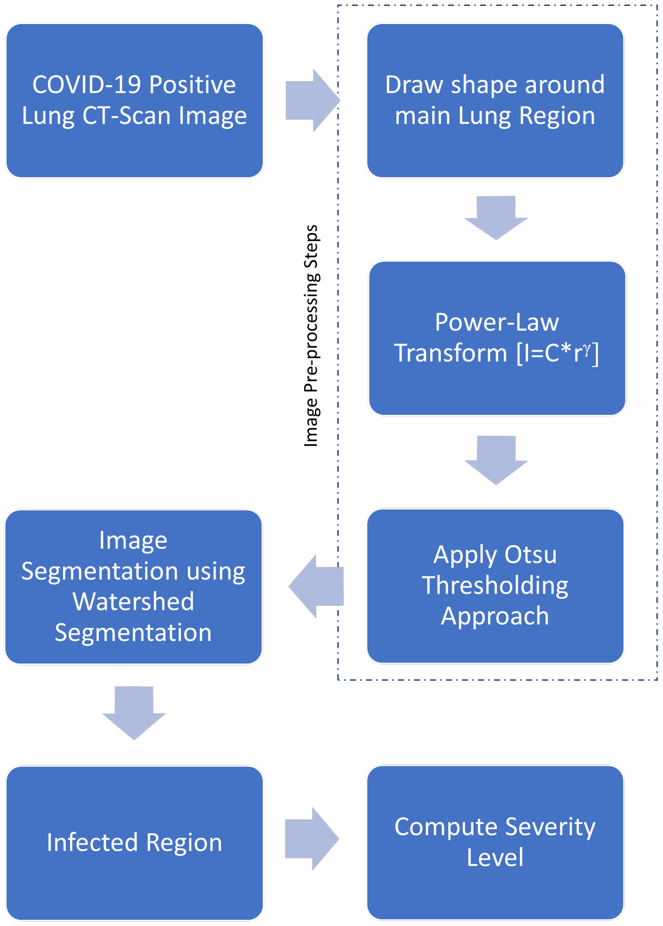

COVID-19 Positive Lung CT-Scan Image
Draw shape around main Lung Region
Image Pre-processing Steps
Power-Law Transform [I=C*r$^{\gamma}$]
Apply Otsu Thresholding Approach
Image Segmentation using Watershed Segmentation
Infected Region
Compute Severity Level

Figure 5.1: Flow chart of extraction of COVID-19 infected region

After extracting the ROI, the watershed image segmentation approach [66], [67] is applied to the pre-processed lung portion to identify the infected region. Finally, the severity level of the infection is computed based on the pixel ratio of the infected region to the extracted lung portion (ROI).

5.2 Image Pre-Processing

The main objective of image pre-processing is to remove unnecessary background and get proper lung portion (ROI) for the segmentation purpose. The first step in image pre-processing is to choose the area by drawing a close shape around the lung portion on the CT-Scan image. The final cropped image obtained using this step is then enhanced using the image enhancement technique.

5.2.1 Image Enhancement

Image enhancement is the second step of image pre-processing. In the proposed work, the intensity transformation method, namely, the power-law transformation, also known as Gamma transformation, is used for image enhancement. Here, the power-law transformation is used to increase the brightness of the high-intensity pixels and is given by [65]

$$I = c \cdot r^{\gamma} \tag{5.1}$$

where I is the output enhanced image, c and γ are positive constants, and r represents the pixel values of the input image. The value of γ is decided based on the brightness of the input image. For example, $\gamma < 1$ is more sensitive to changes in dark pixels, whereas $\gamma > 1$ is more sensitive to changes in bright pixels, as shown in Figure 5.2. In the present work, the values of γ and c are chosen to be 1.5 and 3, respectively.

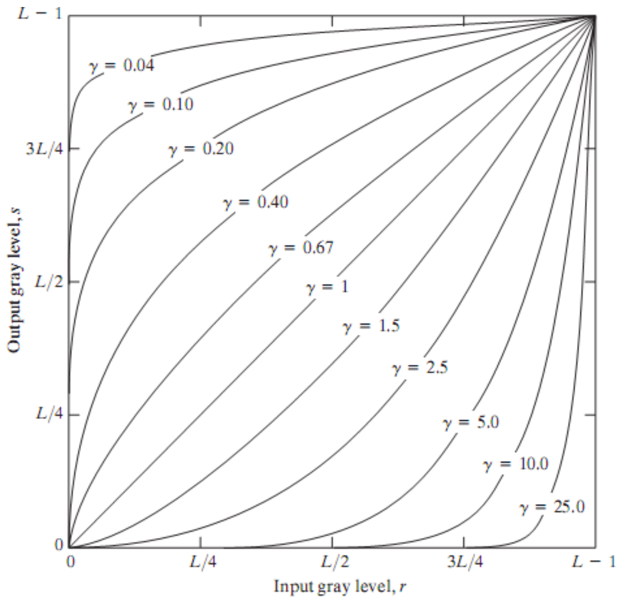


Figure 5.2: Plot of the power-law transformation $(I = c \cdot r^{\gamma})$ for the various values of γ with $c = 1$ (adapted from [65])

5.2.2 Otsu Thresholding Method

In digital image processing, various approaches are used to extract the main object from the image. In our present case, the third stage after the image enhancement consists of a thresholding filter that is used to extract the lung portion and remove the unnecessary background from the input lung CT-Scan image. In this thesis, we employ the statistical-based Otsu thresholding approach which was first introduced by Nobuyuki Otsu in 1979 [23].

Consider an input gray-scale image I having L gray levels represented by $[1, 2, 3, \cdots, L]$ and let the number of pixels having a gray-scale intensity value i be denoted by n_i. Then, the total number of pixels for the input gray-scale image is given by $N = n_1 + n_2 + n_3 + \cdots + n_L$. The probability distribution of the input image is given by,

$$p_i = n_i/N, \qquad p_i \geq 0, \quad \sum_{i=1}^{L} p_i = 1 \tag{5.2}$$

where p_i denotes the probability of the pixel having the gray-scale intensity value i. Practically, the frequency distribution of the pixels, i.e., the number of pixels with gray-scale intensity value i, can be obtained using the histogram plot of the image as shown in Figure 5.3. Here, the image pixels are classified into two classes: C_0 and C_1, and both classes are separated by the threshold value k. Hence, the distribution of classes C_0 and C_1 with respect to pixels gray level value is given by $[1, 2, 3, \cdots, k]$ and $[k+1, k+2, \cdots, L]$, respectively.

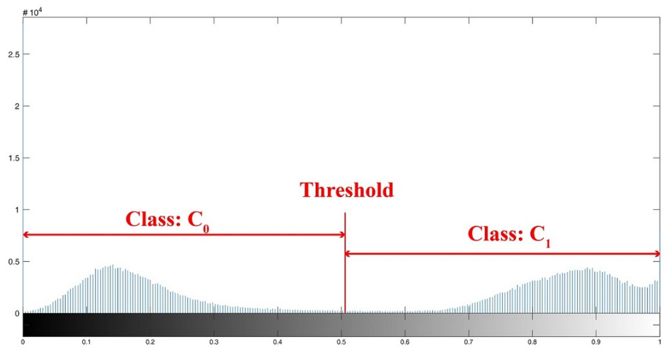


Figure 5.3: Normalized histogram plot of the input image

Then, the probabilities of occurrence for the two classes C_0 and C_1 are, respectively, given by

$$\mathsf{P}(C_0) = \sum_{i=1}^{k} p_i = \omega_0(k) \tag{5.3}$$

and

$$\mathsf{P}(C_1) = \sum_{i=k+1}^{L} p_i = \omega_1(k) = 1 - \omega_0(k) \tag{5.4}$$

since

$$\mathsf{P}(C_0) + \mathsf{P}(C_1) = 1 \tag{5.5}$$

The mean values $\mu_0(k)$ and $\mu_1(k)$ for the two classes C_0 and C_1 are, respectively, given by

$$\mu_0(k) = \sum_{i=1}^{k} i\mathsf{P}(i \mid C_0) = \frac{1}{\omega_0(k)} \sum_{i=1}^{k} ip_i \tag{5.6a}$$

and

$$\mu_1(k) = \sum_{i=k+1}^{L} i\mathsf{P}(i \mid C_1) = \frac{1}{\omega_1(k)} \sum_{i=k+1}^{L} ip_i \tag{5.6b}$$

The total mean level μ_T of the image depends on the probability of occurrence of the two classes and the mean values of these classes, and is given by

$$\mu_T = \omega_0(k)\mu_0(k) + \omega_1(k)\mu_1(k) = \sum_{i=1}^{L} ip_i \tag{5.7}$$

Now, in order to evaluate the "goodness" of the threshold at level k, the normalized measure η, introduced by Otsu [23], given by

$$\eta = \frac{\sigma_B^2}{\sigma_T^2} \tag{5.8}$$

is used, where σ_T^2 is the total variance representing the intensity variance of all the pixels in the given image, and given by

$$\sigma_T^2 = \sum_{i=1}^{L} (i - \mu_T)^2 p_i \tag{5.9}$$

and σ_B^2 is the between-class variance defined as

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 \tag{5.10}$$

Equation (5.10) can be further simplified as

$$\sigma_B^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2 \tag{5.11}$$

The above equation can be further reduced to

$$\sigma_B^2 = \frac{\omega_0(k)}{1-\omega_0(k)}[\mu_T - \mu_0(k)]^2 \tag{5.12}$$

The above equation for σ_B^2 is more efficient computationally to evaluate the “goodness" of the threshold, since we need to calculate only two parameters, namely, $\omega_0(k)$ and $\mu_0(k)$ for various values of k. The total mean μ_T needs to be computed only once. Further, the total variance σ_T^2 is a constant and is independent of the threshold value k. Hence, the measure of separability η depends only on the value of σ_B^2, which needs the calculation of only $\omega_0(k)$ and $\mu_0(k)$ for various values of k. From Equation (5.10) it is clear that a larger value of σ_B^2 indicates more distance between the two class means μ_0 and μ_1. From the above analysis, it is seen that the optimal threshold value k^* can be obtained by maximizing the value of η, which is equivalent to maximizing the value of σ_B^2:

$$\sigma_B^2(k^*) = \max_{1\leq k<L} \sigma_B^2(k) \tag{5.13}$$

After applying the Otsu thresholding method, the output of thresholding is inverted. Then, the morphological operation [68] is applied to the inverted image to remove irrelevant artifacts from the image and to obtain the binary lung mask. Finally, ROI can be obtained by multiplying the binary lung mask with the enhanced image obtained by the Gamma transformation as discussed in Section 5.2.1. The next step after pre-processing is image segmentation.

5.3 Watershed Segmentation Method

In this work, a region-based image segmentation method known as the watershed image transformation technique is used for image segmentation purposes. The watershed transformation was originally proposed by Digabel and Lantuéjoul in 1978 [66] and then further improved by Beucher and Lantuéjoul [67] in 1979. Moreover, in 1990, Meyer and Beucher proposed a marker-controlled segmentation approach to solving the problem of over-segmentation [69].

The overall idea of the watershed segmentation method is based on the geography of landscape relief flooded by water [65]. Any gray-scale image can be visualized as a three-dimensional image with two spatial coordinates and the intensity value of the pixels. In this topographic view of the image, we have three types of points. First, points belonging to local minimum, i.e., points M_1 and M_2 in Figure 5.4. Second, all the points lie between the points $W_1 - M_1$, $W_2 - M_1$, $W_2 - M_2$, and $W_3 - M_2$. If a drop of water were placed in any of these regions, it would flow towards a corresponding local minimum point M_1 or M_2. A group of all these points makes a catchment basin or watershed of that local minimum. Third, the points W_1, W_2, and W_3 at which the drop of water has an equal probability of falling to more than one local minimum point. For example, if a drop of water is placed at point W_2, it can either flow towards point M_1 or M_2. The set of all these points is known as watershed lines or divide lines which make a boundary around the segmented region. The main objective of the watershed segmentation method is to find the watershed lines.

In the present work, the distance transformed-based watershed segmentation method as described in [70] is implemented using MATLAB. First, we convert the input gray-scale extracted ROI into a binary image. Then, we compute the Euclidean distance transform (EDT) [71] on the complement of the binary image. After that, we take the complement of the distance transformed image so that light pixels represent high elevations and dark pixels represent low elevations for the watershed transform. Finally, we apply the watershed

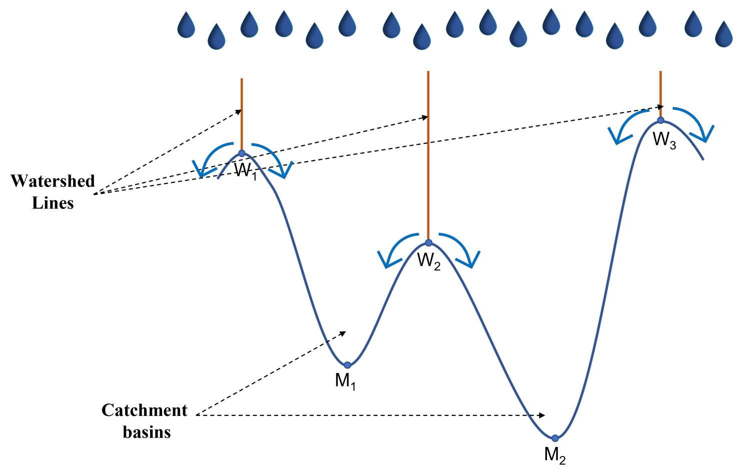


Figure 5.4: Geographical concept of the watershed segmentation method

transform approach to the complement of the distance transformed image. The output of the watershed segmentation extracts the COVID-19 infected lung portion. The final results obtained using watershed transformation are represented in Section 5.5.

5.4 Computation of Severity Level

Currently, RT-PCR testing, which is known as a reverse-transcriptase polymerase chain reaction, has been a widely used laboratory testing method to detect the COVID-19 virus [72]. After testing, if the patient is COVID-19 positive, further treatment is totally based on the severity level of the infection. To check the severity level, doctors use medical imaging methods such as X-Ray or CT-Scan of the lung and diagnose the disease according to that, including medical treatment, medicines, and dosage level of drugs. In this work, the infected region is extracted, and the severity level is computed based on the pixel density of the infected region. This computer-based method helps doctors to reduce their burden and save lots of time. The severity level of the disease is computed based on the equation [29]

$$\text{Infection Rate } (\%) = \frac{\text{Pixel Density of Infected Section}}{\text{Pixel Density of Lung Portion}} \times 100 \tag{5.14}$$

5.5 Results Obtained by Image Segmentation

The flow chart of image segmentation method used in this project is shown in Figure 5.1. In this section, the results obtained during the different stages of the segmentation process are represented. The first stage of the segmentation method is image pre-processing comprising extraction of ROI, image enhancement, and then applying the Otsu thresholding. Here, we consider one sample CT-Scan image, which was classified as a COVID-19 positive using the COViT-CT approach discussed in Chapter 3. Figure 5.5(a) represents the input image with the user-defined shape. This input image has 256 intensity levels (i.e., L=256) represented by $[1, 2, 3, \cdots, 256]$ and the total number of pixels is $N = 210800$. Figure 5.5(b) represents the mask generated by the shape drawn by the user around the input image. Finally, Figure 5.5(c) represents the output of the extracted ROI image after multiplying the input image by the binary mask generated in Figure 5.5(b). After that, the power-law transformation given by Equation (5.1) is applied for image enhancement with $\gamma = 1.5$ and $c = 3$. The final enhanced image is presented in Figure 5.6.

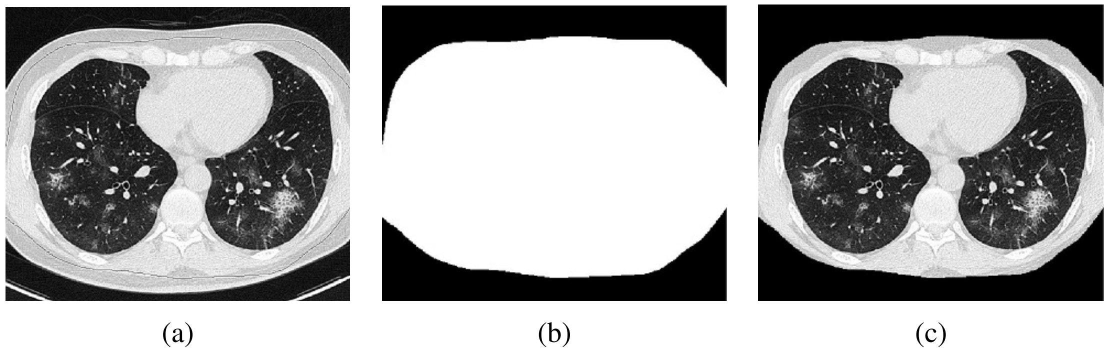

(a) (b) (c)

Figure 5.5: (a) Sample COVID-19 positive image with the user-defined shape (b) Mask generated based on shape drawn by the user (c) Final cropped image, i.e., ROI

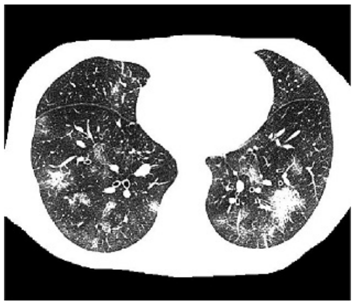


Figure 5.6: Image enhancement through power-law transform with $\gamma = 1.5$ and $c = 3$

Finally, the Otsu thresholding is applied to the enhanced image to produce the binary lung mask. To generate the final lung mask, the output of the Otsu thresholding is inverted, and then, the morphological operations are applied. The generated lung mask is then applied to the enhanced image to remove the background and get the final pre-processed image. Figure 5.7(a) represents the output of the Otsu thresholding method. Moreover, the different statistical values for the input image obtained through the Otsu thresholding are represented in Table 5.1. The table gives the probabilities ω_0 and ω_1, as well as the mean values μ_0 and μ_1 for the two classes, the between-class variance σ_B^2, and the separability measure η for different threshold values around the optimal threshold. For the selected input image, the total mean value $\mu_T = 104.515925$ and the total variance $\sigma_T^2 = 9.633155818 \times 10^3$. For this image, the optimal threshold value is $k^* = 120$, which can be obtained by maximizing the value of σ_B^2. The maximum between-class variance σ_B^2 and the class separability measure η are shown in red fonts in Table 5.1. The inverted image and the extracted lung mask are shown in Figures 5.7(b) and 5.7(c), respectively. The output after the image pre-processing stage is displayed in Figure 5.7(d).

After image pre-processing, the watershed segmentation method described in Section 5.3 is applied to the pre-processed image to extract the infected lung region. Figures 5.8(a) and 5.8(b) represent the binary image and the complement of the binary image obtained from the final pre-processed image, respectively. Figure 5.8(c) is obtained after computing the Euclidean distance transform of the binary image Figure 5.8(b). The complement of that

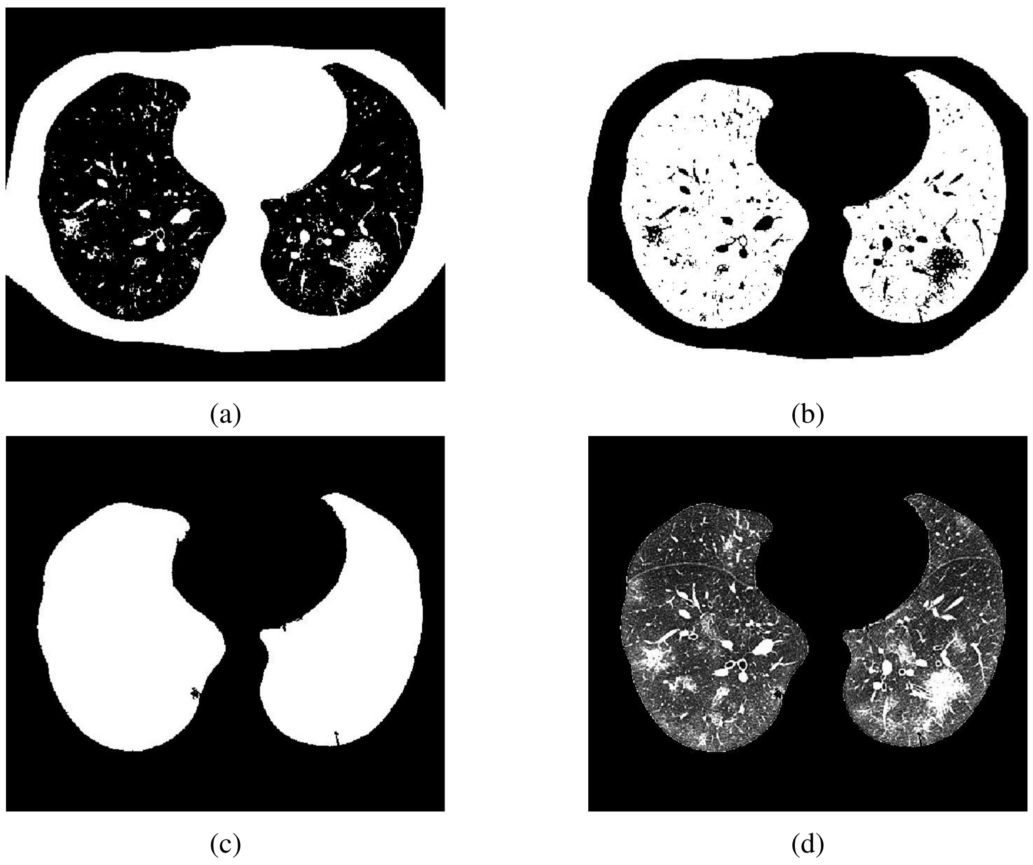


Figure 5.7: (a) Output obtained by the Otsu thresholding (b) Inverted image (c) Extracted lung mask (d) Final output of image pre-processing

Table 5.1: Statistical values for the two classes obtained through the Otsu thresholding method

Threshold	$k = 118$	$k = 119$	$k^* = 120$	$k = 121$	$k = 122$
ω_0	0.584416509	0.584833966	0.585256167	0.585649905	0.586048387
ω_1	0.415583491	0.415166034	0.414743833	0.414350095	0.413951613
μ_0	24.93310605	25.00025145	25.06878384	25.13327933	25.19914359
μ_1	216.4296901	216.5276575	216.6259208	216.7167897	216.8079669
σ_B^2	8906.412906	8906.688355	8906.81866	8906.801491	8906.643343
η	0.924558169	0.924586763	0.92460029	0.924598507	0.92458209
μ_T	104.515925				
σ_T^2	9.633155818×10^3				

distance-transformed image is represented by Figure 5.8(d). The final output of the watershed segmentation algorithm is shown in Figure 5.9, which is obtained by applying the watershed transformation on the distance-transformed image. Figure 5.9(b) represents the segmented portion with various color labels and Figure 5.9(c) represents the infected lung portion.

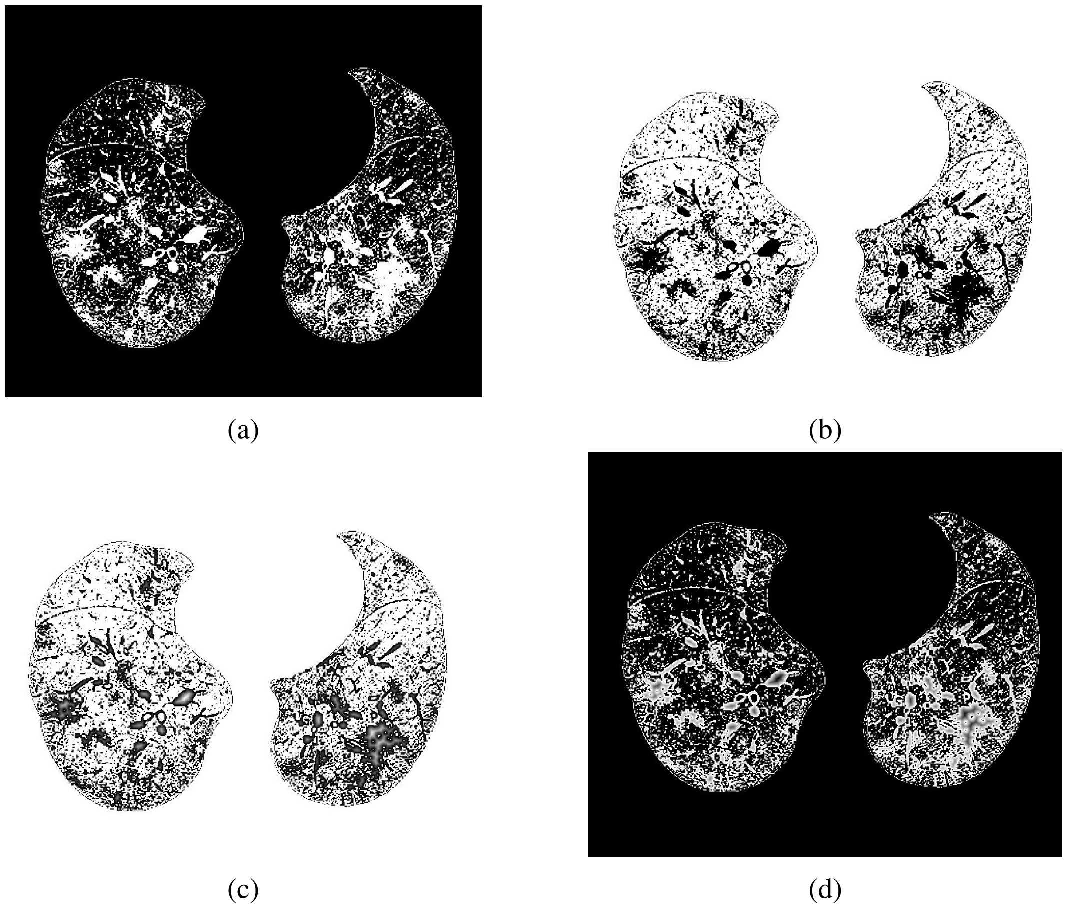


Figure 5.8: (a) Binary image of ROI (b) Complement of binary image (c) Distance transform of the complement of binary image (d) Complement of the distance transform

The final stage of segmentation is to compute the severity level of the disease using Equation (5.14). To calculate the severity level, it is required to calculate the number of pixels in the infected region, i.e., the white portion of Figure 5.10(a) and the total number of pixels in the lung mask in Figure 5.10(b). In our case, the number of pixels in the infected region is 11931, and in the lung mask is 73665. Hence, the percentage of infection rate is

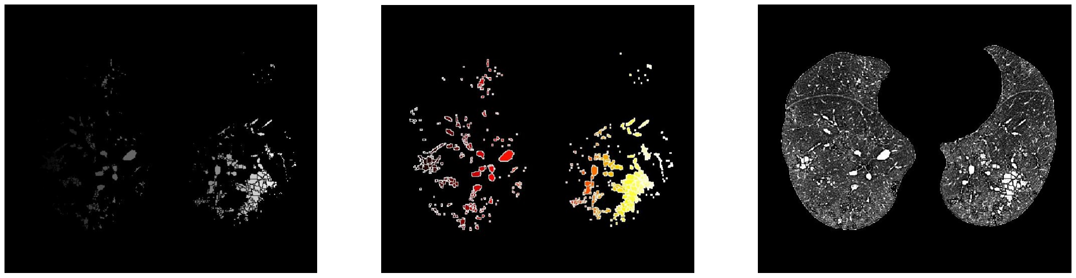

(a) (b) (c)

Figure 5.9: (a) Watershed segmentation output (b) Segmented image with different color-shades (c) Infected lung portion

$(11931 \div 73665) \times 100 = 16.196\%$.

(a) (b)

Figure 5.10: (a) Pixel density of infected segment which is 11931 (b) Pixel density of lung portion which is 73665

5.6 Summary

In this chapter, we have discussed the image segmentation approach for extracting the infected lung portion of positive COVID-19 lung CT-Scan images. Moreover, after detecting the infected region, we computed the severity level of the COVID-19 disease. The overall process of classifying the input lung CT-Scan image, extracting the infected region of COVID-19 positive CT-Scan image, and computing the severity level of the disease will help to reduce the processing time and burden of the doctors.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, four different CNN methods, namely, ResNet50, DarkNet19, GoogLeNet, and VGG16, have been implemented through the transfer learning approach. During this work, the self-attention-based ViT method has also been implemented to classify the lung CT-Scan images into COVID and Non-COVID classes, and we call this as the COViT-CT approach. The overall work includes a series of processes, including image classification, image pre-processing, image segmentation, and computation of the severity level of the disease. By analyzing the results of the implemented five artificial neural network-based image classification techniques and other state-of-the-art works, we conclude that the COViT-CT approach offers the best performance with the highest accuracy of 99.20%, highest precision of 99.62%, and highest F1-score of 99.24%. Furthermore, we have used the semi-automatic image segmentation approach to detect the infected lung portion, only if the image has been classified as COVID-19 positive during image classification through the COViT-CT approach. For image segmentation, the power-law transform technique has been used to enhance the image, which helps to determine an accurate threshold value by the Otsu thresholding method. Subsequently, morphological transformation and watershed segmentation methods have been used to detect the COVID-19 infected region. Finally, the severity of the disease is determined by calculating the infection rate based on the ratio of the pixel density of the infected region with the pixel density of the lung portion. This infection rate can be classified as mild, moderate, or severe based on the percentage value of the infection rate. As the overall technique includes both classifications as well as segmentation and also comprises severity level computation, it is expected to play a significant role in real-time clinical diagnosis to reduce the processing time of diagnosis, and also reduce the burden of the experienced doctors.

6.2 Future Work

It seems that the end of the COVID-19 pandemic is not over yet. It is also possible to classify the dataset into more than two classes such as COVID, Non-COVID, and Pneumonia. We can also classify the dataset into COVID-19 variants such as Alpha, Beta, Delta, and Omicron. The work carried out in this thesis has several possibilities for improvement in future studies. The accuracy can be further improved by implementing other ViT-based approaches such as Transformer iN Transformer (TNT) [73], Pyramid transformer [74], and Swin transformer [75]. We can create hybrid architecture by combining the attention-based transformer method with the CNN method to get the advantages of both methods. The fusion of the transformer and the CNN-based models might improve the performance of the existing models. Further, the work done in this thesis can also be helpful to detect other lung diseases such as lung cancer, blood clots, tuberculosis, and pneumonia.

References

[1] C. Sohrabi, Z. Alsafi, N. O'neill, M. Khan, A. Kerwan, A. Al-Jabir, C. Iosifidis, and R. Agha, "World health organization declares global emergency: A review of the 2019 novel coronavirus (covid-19)," *International journal of surgery*, vol. 76, pp. 71–76, 2020.

[2] A. M. Ismael and A. Şengür, "Deep learning approaches for covid-19 detection based on chest x-ray images," *Expert Systems with Applications*, vol. 164, p. 114054, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420308198

[3] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, and U. Rajendra Acharya, "Automated detection of covid-19 cases using deep neural networks with x-ray images," *Computers in Biology and Medicine*, vol. 121, p. 103792, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482520301621

[4] I. D. Apostolopoulos and T. A. Mpesiana, "Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks," *Physical and Engineering Sciences in Medicine*, vol. 43, no. 2, pp. 635–640, 2020.

[5] P. K. Sethy and S. K. Behera, "Detection of coronavirus disease (covid-19) based on deep features," 2020.

[6] S. H. Yoo, H. Geng, T. L. Chiu, S. K. Yu, D. C. Cho, J. Heo, M. S. Choi, I. H. Choi, C. Cung Van, N. V. Nhung *et al.*, "Deep learning-based decision-tree classifier for covid-19 diagnosis from chest x-ray imaging," *Frontiers in medicine*, vol. 7, p. 427, 2020.

[7] H. Panwar, P. Gupta, M. K. Siddiqui, R. Morales-Menendez, and V. Singh, "Application of deep learning for fast detection of covid-19 in x-rays using ncovnet," *Chaos, Solitons & Fractals*, vol. 138, p. 109944, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S096007792030343X

[8] S. Albahli, "A deep neural network to distinguish covid-19 from other chest diseases using x-ray images," *Current medical imaging*, vol. 17, no. 1, pp. 109–119, 2021.

[9] R. Jain and D. K. Medal, "Coro-net: Cnn architecture to diagnose covid-19 disease using chest x-ray images," in *Machine Vision and Augmented Intelligence—Theory and Applications*. Springer, 2021, pp. 575–587.

[10] D. Singh, V. Kumar, M. Kaur *et al.*, "Classification of covid-19 patients from chest ct images using multi-objective differential evolution–based convolutional neural networks," *European Journal of Clinical Microbiology & Infectious Diseases*, vol. 39, no. 7, pp. 1379–1389, 2020.

[11] X. Wang, X. Deng, Q. Fu, Q. Zhou, J. Feng, H. Ma, W. Liu, and C. Zheng, "A weakly-supervised framework for covid-19 classification and lesion localization from chest ct," *IEEE transactions on medical imaging*, vol. 39, no. 8, pp. 2615–2625, 2020.

[12] S. Ahuja, B. K. Panigrahi, N. Dey, V. Rajinikanth, and T. K. Gandhi, "Deep transfer learning-based automated detection of covid-19 from lung ct scan slices," *Applied Intelligence*, vol. 51, no. 1, pp. 571–585, 2021.

[13] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[14] D. Shome, T. Kar, S. N. Mohanty, P. Tiwari, K. Muhammad, A. AlTameem, Y. Zhang, and A. K. J. Saudagar, "Covid-transformer: Interpretable covid-19 detection using vision transformer for healthcare," *International Journal of Environmental Research and Public Health*, vol. 18, no. 21, p. 11086, 2021.

[15] K. S. Krishnan and K. S. Krishnan, "Vision transformer based covid-19 detection using chest x-rays," in *2021 6th International Conference on Signal Processing, Computing and Control (ISPCC)*. IEEE, 2021, pp. 644–648.

[16] A. Borakati, A. Perera, J. Johnson, and T. Sood, "Diagnostic accuracy of x-ray versus ct in covid-19: a propensity-matched database study," *BMJ open*, vol. 10, no. 11, p. e042946, 2020.

[17] K. Li, Y. Fang, W. Li, C. Pan, P. Qin, Y. Zhong, X. Liu, M. Huang, Y. Liao, and S. Li, "Ct image visual quantitative evaluation and clinical classification of coronavirus disease (covid-19)," *European radiology*, vol. 30, no. 8, pp. 4407–4416, 2020.

[18] A. Mansoor, U. Bagci, B. Foster, Z. Xu, G. Z. Papadakis, L. R. Folio, J. K. Udupa, and D. J. Mollura, "Segmentation and image analysis of abnormal lungs at ct: current approaches, challenges, and future trends," *Radiographics*, vol. 35, no. 4, pp. 1056–1076, 2015.

[19] J. Amin, M. Sharif, M. A. Anjum, Y. Nam, S. Kadry, and D. Taniar, "Diagnosis of covid-19 infection using three-dimensional semantic segmentation and classification of computed tomography images," *Computers, Materials and Continua*, vol. 68, no. 2, pp. 2451–2467, 2021.

[20] X. Liu, K. Wang, K. Wang, T. Chen, K. Zhang, and G. Wang, "Kiseg: A three-stage segmentation framework for multi-level acceleration of chest ct scans from covid-19

patients," in *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer, 2020, pp. 25–34.

[21] S. C. Satapathy, D. J. Hemanth, S. Kadry, G. Manogaran, N. M. Hannon, and V. Rajinikanth, "Segmentation and evaluation of covid-19 lesion from ct scan slices-a study with kapur/otsu function and cuckoo search algorithm," 2020.

[22] J. N. Kapur, P. K. Sahoo, and A. K. Wong, "A new method for gray-level picture thresholding using the entropy of the histogram," *Computer vision, graphics, and image processing*, vol. 29, no. 3, pp. 273–285, 1985.

[23] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[24] C. Li, C. Xu, C. Gui, and M. D. Fox, "Distance regularized level set evolution and its application to image segmentation," *IEEE transactions on image processing*, vol. 19, no. 12, pp. 3243–3254, 2010.

[25] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Transactions on image processing*, vol. 10, no. 2, pp. 266–277, 2001.

[26] V. Rajinikanth, S. Kadry, K. P. Thanaraj, K. Kamalanand, and S. Seo, "Firefly-algorithm supported scheme to detect covid-19 lesion in lung ct scan images using shannon entropy and markov-random-field," *arXiv preprint arXiv:2004.09239*, 2020.

[27] A. Valizadeh and M. Shariatee, "The progress of medical image semantic segmentation methods for application in covid-19 detection," *Computational Intelligence and Neuroscience*, vol. 2021, 2021.

[28] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[29] V. Rajinikanth, N. Dey, A. N. J. Raj, A. E. Hassanien, K. Santosh, and N. Raja, "Harmony-search and otsu based system for coronavirus disease (covid-19) detection using lung ct scan images," *arXiv preprint arXiv:2004.03431*, 2020.

[30] N. Khehrah, M. S. Farid, S. Bilal, and M. H. Khan, "Lung nodule detection in ct images using statistical and shape-based features," *Journal of Imaging*, vol. 6, no. 2, p. 6, 2020.

[31] D. Worldometer, "Covid-19 coronavirus pandemic," *World Health Organization, www.worldometers.info*, 2021.

[32] E. Soares, P. Angelov, S. Biaso, M. H. Froes, and D. K. Abe, "Sars-cov-2 ct-scan dataset: A large dataset of real patients ct scans for sars-cov-2 identification," *medRxiv*, 2020.

[33] A. L. Samuel, “Some studies in machine learning using the game of checkers,” *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[34] R. Dechter, “Learning while searching in constraint-satisfaction problems,” 1986.

[35] I. N. Aizenberg, N. N. Aizenberg, and J. Vandewalle, “Multiple-valued threshold logic and multi-valued neurons,” in *Multi-valued and universal binary neurons*. Springer, 2000, pp. 25–80.

[36] J. D. Kelleher, *Deep learning*. MIT press, 2019.

[37] S. Agatonovic-Kustrin and R. Beresford, “Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research,” *Journal of pharmaceutical and biomedical analysis*, vol. 22, no. 5, pp. 717–727, 2000.

[38] K. L. Du and M. N. S. Swamy, *Neural Networks and Statistical Learning., Second Edition*. Springer-Verlag, London, 2019.

[39] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[40] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain.” *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[41] B. Planche and E. Andres, *Hands-On Computer Vision with TensorFlow 2: Leverage deep learning to create powerful image processing apps with TensorFlow 2.0 and Keras*. Packt Publishing Ltd, 2019.

[42] E. Ovalle-Magallanes, J. G. Avina-Cervantes, I. Cruz-Aceves, and J. Ruiz-Pinales, “Transfer learning for stenosis detection in x-ray coronary angiography,” *Mathematics*, vol. 8, no. 9, p. 1510, 2020.

[43] D. Yu, H. Wang, P. Chen, and Z. Wei, “Mixed pooling for convolutional neural networks,” in *International conference on rough sets and knowledge technology*. Springer, 2014, pp. 364–375.

[44] N. Akhtar and U. Ragavendran, “Interpretation of intelligence in cnn-pooling processes: a methodological survey,” *Neural computing and applications*, vol. 32, no. 3, pp. 879–898, 2020.

[45] S. Ioffe, “Batch renormalization: Towards reducing minibatch dependence in batch-normalized models,” *arXiv preprint arXiv:1702.03275*, 2017.

[46] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[47] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[48] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[49] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[50] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva, "Places: An image database for deep scene understanding," *arXiv preprint arXiv:1610.02055*, 2016.

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

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[53] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[54] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," *arXiv preprint arXiv:1601.06733*, 2016.

[55] Q. Wang, B. Li, T. Xiao, J. Zhu, C. Li, D. F. Wong, and L. S. Chao, "Learning deep transformer models for machine translation," *arXiv preprint arXiv:1906.01787*, 2019.

[56] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[57] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[59] Y. Pathak, P. K. Shukla, and K. Arya, "Deep bidirectional classification model for covid-19 disease infected patients," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 4, pp. 1234–1241, 2020.

[60] H. Panwar, P. Gupta, M. K. Siddiqui, R. Morales-Menendez, P. Bhardwaj, and V. Singh, "A deep learning and grad-cam based color visualization approach for fast detection of covid-19 cases using chest x-ray and ct-scan images," *Chaos, Solitons & Fractals*, vol. 140, p. 110190, 2020.

[61] S. Sen, S. Saha, S. Chatterjee, S. Mirjalili, and R. Sarkar, “A bi-stage feature selection approach for covid-19 prediction using chest ct images,” *Applied Intelligence*, vol. 51, no. 12, pp. 8985–9000, 2021.

[62] T. Kaur and T. K. Gandhi, “Classifier fusion for detection of covid-19 from ct scans,” *Circuits, systems, and signal processing*, pp. 1–18, 2022.

[63] P. Silva, E. Luz, G. Silva, G. Moreira, R. Silva, D. Lucio, and D. Menotti, “Covid-19 detection in ct images with deep learning: A voting-based scheme and cross-datasets analysis,” *Informatics in medicine unlocked*, vol. 20, p. 100427, 2020.

[64] F. Y. Shih, *Image Segmentation*. Boston, MA: Springer US, 2009, pp. 1389–1395. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_1011

[65] R. C. Gonzalez, *Digital image processing*. Pearson education india, 2009.

[66] H. Digabel and C. Lantuéjoul, “Iterative algorithms,” in *Proc. 2nd European Symp. Quantitative Analysis of Microstructures in Material Science, Biology and Medicine*, vol. 19, no. 7. Riederer Verlag, 1978, p. 8.

[67] S. Beucher and C. Lantuéjoul, “Use of watersheds in contour detection,” in *Proceedings of the International Workshop on Image Processing*. CCETT, 1979.

[68] P. Soille *et al.*, *Morphological image analysis: principles and applications*. Springer, 1999, vol. 2, no. 3.

[69] F. Meyer and S. Beucher, “Morphological segmentation,” *Journal of visual communication and image representation*, vol. 1, no. 1, pp. 21–46, 1990.

[70] Q. Chen, X. Yang, and E. M. Petriu, “Watershed segmentation for binary images with different distance transforms,” in *Proceedings of the 3rd IEEE international workshop on haptic, audio and visual environments and their applications*, vol. 2, 2004, pp. 111–116.

[71] C. R. Maurer, R. Qi, and V. Raghavan, “A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 265–270, 2003.

[72] Y. Li, L. Yao, J. Li, L. Chen, Y. Song, Z. Cai, and C. Yang, “Stability issues of rt-pcr testing of sars-cov-2 for hospitalized patients clinically diagnosed with covid-19,” *Journal of medical virology*, vol. 92, no. 7, pp. 903–908, 2020.

[73] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, “Transformer in transformer,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 15 908–15 919, 2021.

[74] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 568–578.

[75] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.