COVID-19 Classification in Chest CT Images Using Deep Convolution Neural Networks

by

Malek Abdurhman Elgadi

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

COVID-19 Classification in Chest CT Images Using Deep Convolution Neural Networks

by

Malek Abdurhman Elgadi

Supervisory Committee

Dr. Fayez Gebali, Co-Supervisor
(Department of Electrical and Computer Engineering)

Dr. Haytham El Miligi, Co-Supervisor
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Fayez Gebali, Co-Supervisor
(Department of Electrical and Computer Engineering)

Dr. Haytham El Miligi, Co-Supervisor
(Department of Electrical and Computer Engineering)

ABSTRACT

The coronavirus disease (COVID-19) has rapidly spread over the world since the end of 2019. The immediate and accurate diagnosis of COVID-19 is essential for improving the prognosis of this disease and reducing the pandemic spread. Although the PCR test is a standard test to diagnose COVID-19, radiography techniques such as chest X-rays and computed tomography (CT) scans are preferred for detection of COVID-19 disease. Deep learning and convolutional neural Networks (CNNs) play an important role in the early and accurate detection of COVID-19 using radiography images. In this project, a deep convolutional neural network framework based on a transfer learning technique with fine-tuning is suggested for detection and classification of COVID-19. Two pre-trained models i.e., VGG16 and DenseNet201 are trained using COVID-19 CT images dataset. Various experiments are performed to evaluate the performance of the pre-trained models using several evaluation parameters. The results show that the best accuracy of 99.4%, recall of 99.39%, precision of 99.4%, F1-score of 99.39%, and Area Under the Curve (AUC) of 99.93% are achieved by VGG-16 model. DenseNet201 model also shown a competitive result with an accuracy of 99.13 since it has lesser execution time and fewer parameters compared to other deep learning models.

Contents

List of Tables

List of Figures

List of Acronyms

AUC	Area Under Curve
CAP	Community Acquired Pneumonia
CNN	Convolution Neural Network
COVID-19	Coronavirus Disease
CT	Computed Tomography
DCNN	Deep Convolution Neural Network
DenseNet	Dense Convolution Network
DL	Deep Learning
FC	Fully Connected
FN	False Negative
FP	False Positive
FPR	False Positive Rate
LR	Learning Rate
ML	Machine Learning
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
ROC	Receiver Operating Characteristic
RT-PCR	Reverse Transcription Polymerase Chain Reaction
TN	True Negative
TP	True Positive
TPR	True Positive Rate
WHO	World Health Organization

ACKNOWLEDGEMENTS

In the name of Allah, the most Gracious, the most Merciful. All praise be to Allah the Almighty who has given me knowledge, patience, and perseverance to complete my Master's thesis.

I am indebt to my supervisor, Dr. Fayez Gebali, for his help, advice, guidance, support, and encouragement throughout this incredible journey of my studies and related research. I have benefited greatly from his wealth of knowledge.

My heartfelt thanks to my co-supervisor, Dr. Haytham El Miligi, for his advice, guidance, mentoring, support, and encouragement.

Finally, I must express my very profound gratitude to my parents, my wife, my sons, all my family members, and friends for their unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Special thank you.

DEDICATION

To my parents, my wife, my sons: Abdurhman and Mohammed, and all my family members for their support, encouragement, continuous guidance, and prayers.

To all my professors, instructors, friends, and colleagues who supported and encouraged me during this journey.

Chapter 1

Introduction

1.1 Overview

In December 2019, the Coronavirus Disease (COVID-19) was first reported in Wuhan, China which is known as a respiratory disease caused by a severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) [1]. With the quick spread and increase in the number of cases worldwide, on March 11, 2020, the World Health Organization (WHO) declared the novel COVID-19 as a pandemic. Currently around 594 million people are infected by COVID-19 worldwide, and more than 6 million people have died around the world (as of August, 2022) [2]. The coronavirus is a highly infectious virus that can be spread via the air by an infected person while sneezing or coughing. It can also be spread by contact with surfaces contaminated by an infected person. The most common symptoms of COVID-19, as declared by the WHO, are fever, dyspnea, cough, short breathing, sore throat, fatigue, and headache [3].

The standard test for diagnosing COVID-19 cases is based on reverse transcription polymerase chain reaction (RT-PCR). COVID-19 RNA can be detected in respiratory specimens using nasopharyngeal or oropharyngeal swabs. But the detection of this disease using RT-PCR is very time consuming, the test takes around 6 to 8 hours to process the sample and show results [4]. It also gives error-prone results such as high false-negative rates [5]. Such a time is long compared with the continuously growing spread rate of COVID-19. As result, given the widespread and the lack of fast accurate tests, a faster screening method is required for COVID-19 outbreaks [6].

Besides RT-PCR, medical images, such as chest X-ray and chest computed tomography (CT), have become a primary method for diagnosing COVID-19 cases [7].

Patches of Ground-Glass Opacity (GGO) and consolidation on radiographs are the most common signs of COVID-19 infection according to [8]. However, the viral infection indicators can be subtle and it is difficult for radiologists to differentiate COVID-19 from normal cases or other pneumonia. Thus, intelligent diagnostic systems that can detect COVID-19 and pneumonia in chest radiography images are highly required.

1.2 Motivation

In recent years, machine learning (ML) and deep learning (DL) have been proven as promising methods in solving different types of problems, such as image recognition [9], object detection [10], etc. ML and DL have helped a lot in the medical field, such as health monitoring, robot surgery, disease detection, and many more. One method to detect disease is done by using convolutional neural networks (CNNs). Convolutional neural network (CNN) is a type of deep learning that is able to obtain information from digital images, videos, and other visual inputs. CNN gives very promising results in disease detection [10]. CNN can be used to detect COVID-19 and pneumonia from chest CT images while providing cost-effective, reliable and reducing the time required for disease detection with maintaining a high accuracy.

1.3 Related Work

Several research works and studies have been proposed in recent years to detect and classify COVID-19 from radiological imaging. Wang et al. [11] proposed a structure that achieved a more outstanding performance. Pre-trained models (PTMs) firstly were utilized to learn the features, and a unique (L, 2) transfer feature learning approach was suggested to extract the features. Then, they introduced a pre-trained network selection approach for fusion to choose the best two models defined by PTM and NLR. Finally, discriminant correlation analysis was developed to help fuse the two features from the two models via deep chest CT (CCT) fusion. They achieved sensitivity, precision, and F1-score of 98.30%, 97.38%, and 97.62%, respectively.

Sarker et al. [12] proposed CNN model using DenseNet-121 for automatic classification of COVID-19 patients. They used transfer learning technique to train the deep learning network by removing gradient problem. They developed a website that take radiology images and produced the infected regions. The results showed that the accuracy obtained from this technique was 87%.

Wang et al. [13] identified the COVID19 accurately using a learning framework known as redesigned COVID-Net, which performed separate feature normalization. They used 2492 CT scan images as Site (A) and 568 CT scan images as Site (B) for their model. They achieved an accuracy, precision, recall, F1 score, and AUC of 90.83%, 90.87%, 85.89%, 95.75%, and 96.24%, respectively for Site (A). They achieved an accuracy, precision, recall, F1 score, and AUC of 78.69%, 78.83%, 79.71%, 78.02%, and 85.32%, respectively for Site B.

Soares et al. [14] created a SARS-CoV-2 CT-scan dataset, where there were 1252 Covid-19 CT-scans and 1230 non Covid-19 CT-scans. A total of 2482 CT-scans were obtained from a hospital located in Sao Paulo, Brazil. The dataset was tested using eXplainable Deep Neural Network (xDNN) and obtained 97.31% F-measure.

Li et al. [15] developed COVID-19 detection neural network (COVNet) to extract the features from chest CT images for detection COVID-19 patients. COVNet was trained over dataset contains 4356 chest CT images. The accuracy obtained from COVNet was 95%.

El Asnaoui et al. [16] proposed CNN models for binary classification of COVID-19 using chest X-ray and CT images. They used pre-trained models namely VGG-16, InceptionResNet-V2, VGG-19, Xception, Inception-V3, MobileNet-V2, and ResNet50 for classification. The classification accuracy that obtained from MobileNet-V2 was 96%.

Panwar et al. [17] used a transfer learning algorithm using VGG-19 model for detection of COVID-19 patients from chest X-ray and CT images. They also used Gradient Weighted Class Activation Mapping (Grad-CAM) to visualize the infected area in the images. They achieved an overall accuracy of 95.61%.

Jaiswal et al. [18] proposed deep learning model using pre-trained DenseNet201 model to classify COVID-19 disease on chest CT-scans. The results of their study showed that the proposed DenseNet201 model achieved accuracy of 97% with the training and validation accuracy as 99.82% and 97.4%, respectively.

Zheng et al. [19] proposed a weakly-supervised deep learning-based software for diagnosis of COVID-19 patients using 3D CT scans. They used pre-trained UNet technique for segmentation of 3D lung images. The segmented regions are applied on deep neural network for prediction of infected regions. The result obtained from their model was 95.9% ROC AUC and 97.6% PR AUC.

1.4 Contribution

The contributions of this work can be summarized as follows:

1. We propose two deep CNN models for detection and classification of COVID-19 using chest CT image dataset. The proposed models is based on a transfer learning technique using VGG-16 and DenseNet201 pre-trained models.

2. The proposed models are trained and tested using an imbalance CT images dataset. To validate the applicability of the proposed models, we created a balanced dataset using the undersampling technique.

3. Several experiments and result analysis are performed to evaluate the performance of the proposed models using various evaluation metrics such as accuracy, precision, recall, F1-score, and Area Under the Curve (AUC).

1.5 Outline

The structure of the project is organized as follows:

Chapter 1 presents the problem considered, scope of the research, the motivation, related works, and the contributions of this research.

Chapter 2 reviews the background and fundamentals of CNN and describes transfer learning technique.

Chapter 3 presents the methodology that has been used for the analysis, explains the dataset, data preprocessing steps, data splitting, and CNN models that has been used in this research. Fine-tuning, training the models, and the parameters used to evaluate the performance of the proposed models are also described.

Chapter 4 presents the performance analysis and experiment results of the models for the unbalanced and balanced streams of data. The performance metrics are presented along with a discussion of the results.

Chapter 5 contains conclusion of this project and describes the future work.

Chapter 2

Preliminaries

This chapter presents a basic concepts of CNN networks. We also present the conventional architectures of CNNs and describe different components of each layer of a CNN networks. Moreover, in this chapter, we describe different activation functions used in CNNs. Finally, we end this chapter by presenting an overview of the transfer learning techniques.

2.1 Convolution Neural Network

Convolution neural network (CNN), also known as ConvNet, is a type of deep learning (DL) architectures, which can be used for different applications, such as object detection, classification of images, video, texts, and other computer vision related applications. The first architecture of CNNs, known as LeNet-5, proposed by LeCun et al. [20], which is widely used to classify handwritten digits.

A CNN is a DL algorithm that can take an image as input, assign learnable weights or importance to various aspects in the image and differentiate one from the other. The amount of preprocessing required in a CNN is much lower as compared to other classification algorithms. While filters in primitive approaches are often hand-engineered, CNNs are able to learn these filters/features with sufficient training. The architecture of a CNN is similar to that of the connectivity pattern of neurons in the human brain and was inspired by the organization of the visual cortex. The role of CNN is to reduce the image into a form that is easier to process, without losing critical features for a good prediction. This is important when designing an algorithm that is scalable to massive datasets [21]. Figure 2.1 shows the concept

of CNN operations. The main CNN operations are input layer, convolution layer, pooling layer, fully connected layer, batch normalization layer, and dropout layer.

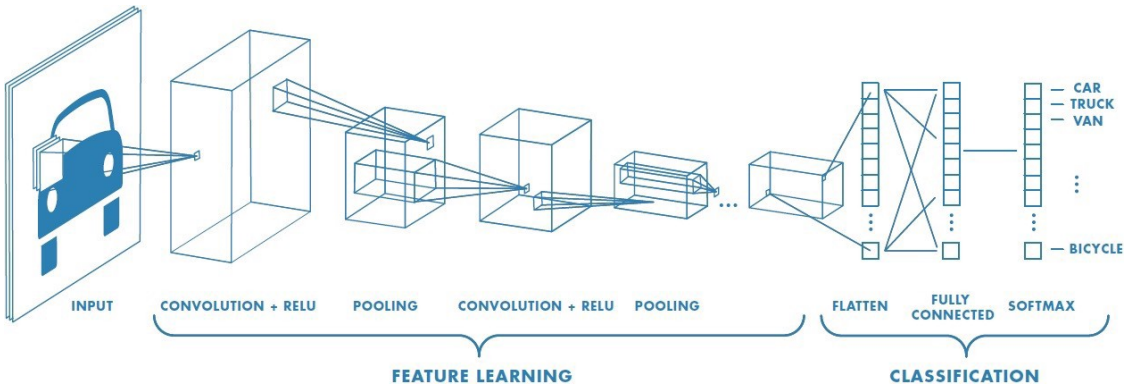


Figure 2.1: The operations of a convolution neural network [21].

2.1.1 Input Layer

The CNN takes an image as an input with an n-dimensional array of pixel values. An array of values is called a tensor, a tensor can be a matrix with (2-dimensional) or a vector with (1-dimensional) that represents all data types. The input image has a number of channels. The number of channels is commonly three for RGB images [22].

2.1.2 Convolutional Layer

The convolutional layer is a main component of the CNN architecture. This layer performs feature extraction using a combination of linear and nonlinear operations, such as convolution operations and activation functions [23]. In other words, the purpose of the convolution layer is to extract different types of features from the input image. This layer is applied across the input (a tensor) which is an array of numbers to obtain the output which is called a feature map or convolved feature. Conventionally, the first convolutional layers capture low-level features such as edges, colour, gradient orientation, etc. The next convolutional layers learn more complex features such as patterns and textures. The last convolutional layers learn high-level features such as objects or parts of objects [21].

The element that is responsible for performing the convolution operation is called a kernel, which is a small array of numbers. The kernel filters out everything that is not important for the feature map, focusing only on the most important information. The feature map is extracted by passing the kernel filter over the input tensors with a certain stride length. Stride is the number of pixels shifting over the input tensor. At each stride, the kernel filter matrix is multiplied by the input matrix and summed to obtain the output value in the feature map matrix [23].

Figure 2.2 shows an input image with a (5×5) dimensions (shown in green) and a (3×3) kernel filter (shown in yellow) to obtain a (3×3) convolved feature (shown in purple). The stride size is one the kernel convolves around the input by shifting one unit at a time, with each shift performing a matrix multiplication of the kernel and the input image to obtain the output feature map [21].

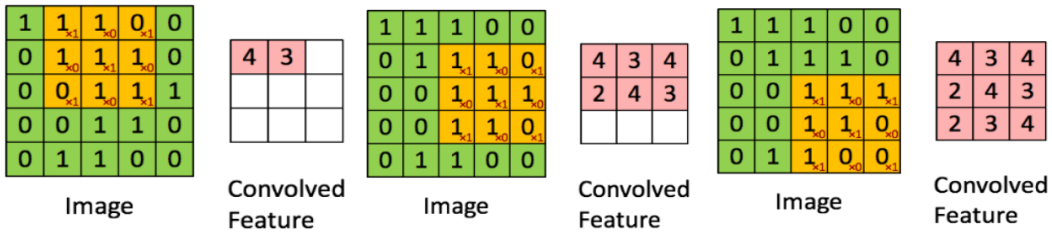


Figure 2.2: Convoluting a (5×5) input image with a (3×3) kernel filter and a stride size 1 to obtain a (3×3) convolved feature [21].

The convolved feature matrix can remain the same size of the kernel dimensions or the input dimensions. This is done by applying valid or same padding. The valid padding is when convolved feature has the same size of the kernel dimensions. The same padding is when convolved feature has the same size of the input dimensions [21].

2.1.3 Pooling Layer

The pooling layer performs a down-sampling operation to reduce the dimension of the convolved feature. This is done to extract dominant features which are rotational and positional invariant, thus reducing the computations required to process the data and maintaining the process of efficiently training the model. There are two common types of pooling operation: average pooling and max pooling. Average pooling returns the average of the corresponding values from the portion of the image covered by the

kernel. On the other hand, max pooling returns the largest value from the portion of the image covered by the kernel [24]. Figure 2.3 shows an example of a pooling operation.

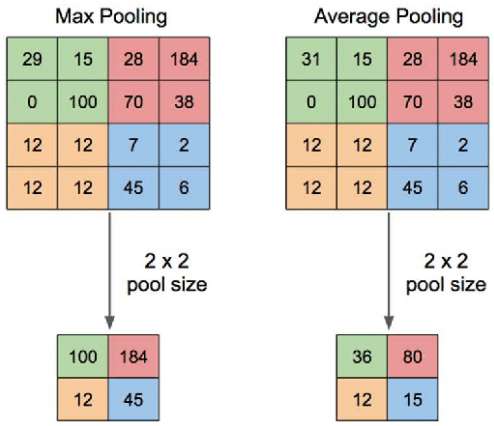


Figure 2.3: Examples of max pooling and average pooling operation [24].

2.1.4 Fully Connected Layer

The output feature maps from the final convolution or pooling layer are generally flattened, i.e. converted into a one-dimensional vector, and connected to one or more fully connected (FC) layers, in which every input is connected to every output from the previous layer [23]. The FC layer is also called a dense layer. The features extracted by the final convolution layers and down-sampled by the pooling layers are used to produce the output of the network [25]. The final output is the probability for each class in a classification example. Figure 2.4 shows a fully connected neural network with four layers.

2.1.5 Dropout Layer

Dropout refers to discarding the neurons during the training process, so they are not considered during a certain forward or backward pass which reduces the network. Dropout layer randomly sets a portion of nodes from the FC layers to zero with a frequency of rate at each step during training time. Dropout is commonly used method to reduce the effect of overfitting [26]. Overfitting in a machine learning model occurs

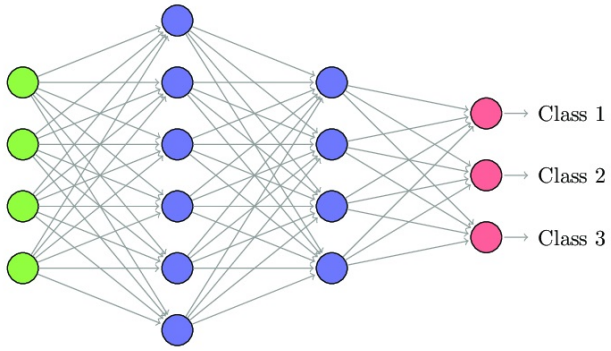


Figure 2.4: Example of a fully connected neural network.

when the training accuracy is much greater than the testing accuracy [23]. Dropout is a commonly used after multiple fully connected layers towards the tail end of a CNN. An example of dropout is shown in Figure 2.5. The dropout rate is the probability of retaining a given unit in the layer, where 0.0 indicates that there are no outputs from the layer and 1.0 indicates that there is no dropout [27].

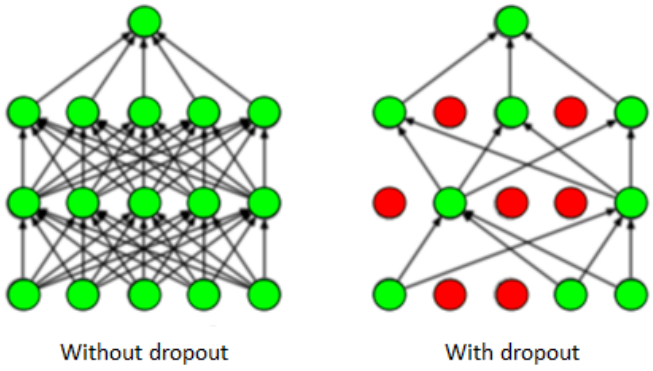


Figure 2.5: Dropout in neural network.

2.1.6 Batch Normalization

Training ML model is stable and more efficient when the input distributions of layer inputs are the same. A model can be biased due to variations in these distributions. Batch normalization is used to perform normalizing operation on the inputs to a layer coming from a previous layer [28].

2.1.7 Activation functions

Activation functions are non-linear mathematical functions, which estimate non-linear functions to generate the output. The activation function is added at the end of each layer of CNNs. Softmax [29] and Rectified Linear Unit (ReLU) [30] are activation functions mostly used in CNNs and are described below. The softmax activation function is given by:

$$Softmax = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.1}$$

where z_i is the input tensor and k is the number of the input tensors.

Softmax function converts real output values from the last FC layer to target class probabilities, where all output values sum to one and each value ranges between zero and one. Softmax activation function is commonly used in multi-class classification tasks [31]. The softmax function is used in the last FC layer of the proposed models so the results can be interpreted as a probability distribution for three classes. Figure 2.6 shows the softmax function in neural networks.

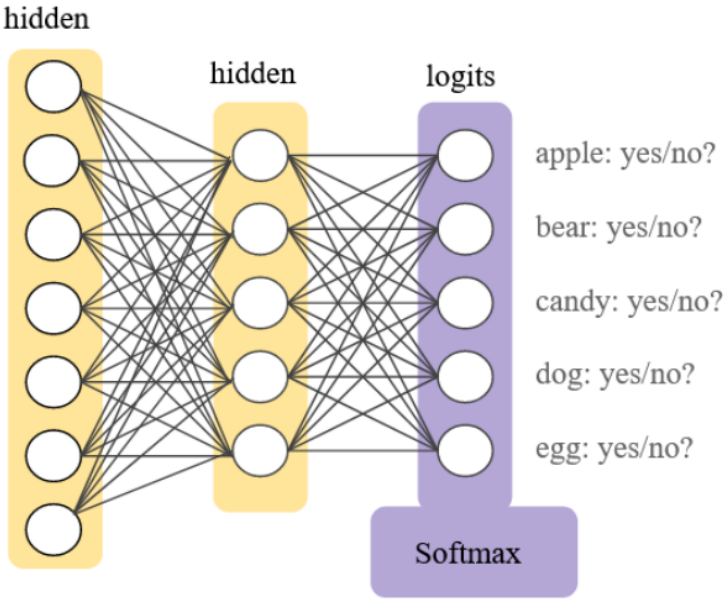


Figure 2.6: Softmax function in a neural network.

The Rectified Linear Unit (ReLU) function is given by:

$$ReLU(x) = \max(0, x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} \tag{2.2}$$

where x is the input value.

This activation function ranges from zero to the input value x, if x is negative, it will be converted to zero by this activation function. The ReLU activation function is the maximum value between zero and the input.

2.2 Transfer Learning

Transfer learning is a deep learning technique to train a model on a small data set, in which model is already pre-trained on large scale dataset, such as ImageNet, which consists of 1.4 million images with 1000 categories (i.e., car, truck, lion, fish, orange, etc.) [32]. Transfer learning contains adopting features learned on the source problem and leveraging them on a new similar problem. This portability of learned generic features is a unique advantage of machine learning that makes it useful in various tasks with small datasets.

Transfer learning method has been successfully applied for improving CNNs performance and solving many deep learning problems [33]. In image classification, transfer learning is a method where a model trained on one task is reused to a new but similar task, requiring minimal retraining and fine-tuning. For example, a task for COVID-19 image classification on small dataset can be initiated using a CNN model trained on the ImageNet dataset. Since deep learning needs a large amount of training data to learn certain patterns, the need for large amounts of data is a significant issue, particularly in the medical imaging field. The effectiveness of transfer learning can be limited when transferring data contents from one type to another. In this case, transfer learning is no better than training from scratch [34].

There are two commonly used strategies to exploit transfer learning on a pre-trained model. Feature extraction and fine-tuning. Feature extraction method is a process to remove FC layers from a model pre-trained on ImageNet. The initial architecture and the learned parameters, which contains a series of convolution and

pooling layers, referred to as the convolutional base, is used as a features extractor for the input of the new classification model. In this method, any machine learning classifier, as well as the FC layers in CNNs, can be added on top of the feature extractor. Due to the differences between ImageNet and medical images, this technique is not common in deep learning research on medical images.

Fine-tuning is a method that can not only replace FC layer of the pre-trained model with a new FC layer to retrain on the given dataset, but also make some modifications to the pre-trained model, such as architecture adjustments and parameter fine-tuning. In this method, all or part of the layers in the convolutional base can be fine-tuned [23] [34]. This technique is more often applied in deep learning research on medical images [23]. The concept of the transfer learning technique is illustrated in Figure 2.7.

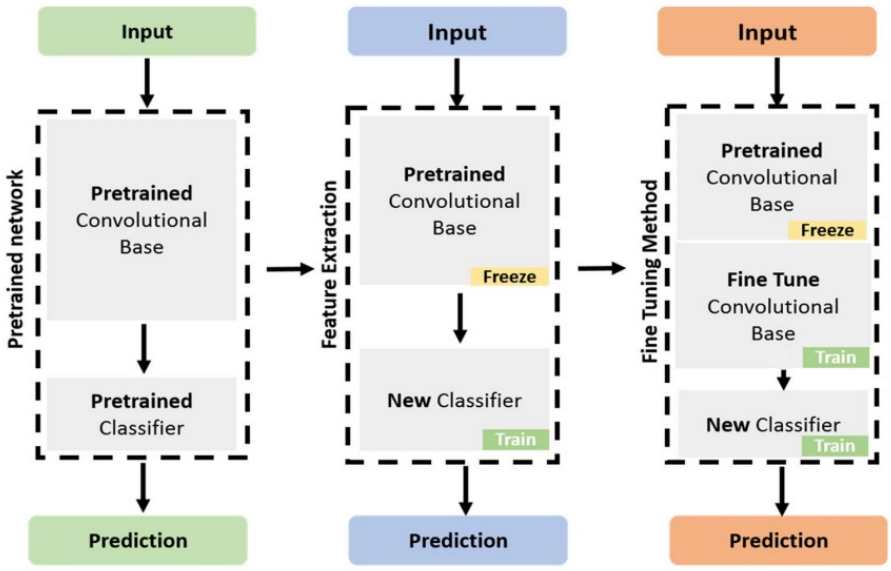


Figure 2.7: Concept of the transfer learning technique.

Chapter 3

Methodology

In this chapter, we present a detailed description of the suggested methodology for detecting COVID-19 in chest CT scan images. The process of our suggested method for detecting COVID-19 is illustrated in Figure 3.1. The proposed framework includes the following steps: the COVID-19 CT scan dataset description, preprocessing of the data, selecting different CNN models i.e., VGG-16 and DenseNet201, training the models, and evaluating the performance of the models using evaluation metrics. The steps are discussed in more detail in the following sections.

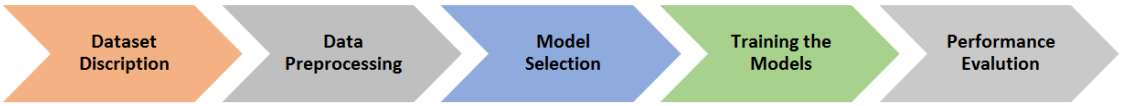


Figure 3.1: The proposed framework.

3.1 Dataset Description

The dataset used in this project was obtained from the Kaggle COVID-19 radiography database [35]. This dataset was constructed by collecting data from 7 public datasets to create a large CT scan dataset of the lungs for COVID-19 [36]. In this dataset, the CT images are divided into three classes, namely COVID-19, Community Acquired Pneumonia (CAP), and Normal. The COVID-19 CT images dataset consist of 7593 COVID-19 CT images from 466 patients, 2618 Pneumonia CT images from 60 patients, and 6893 Normal CT images from 604 patients [35]. These datasets have

been publicly used in the COVID-19 diagnostic literature and have demonstrated their efficiency in deep learning applications. The details of COVID-19 CT images dataset are given in Table 3.1, where, a sample of CT images is illustrated in Figure 3.2.

Type	**Total Patients**	**Total Images**
Normal	604	6893
COVID-19	466	7593
Pneumonia	60	2618
Total	1130	17104

Table 3.1: Details of COVID-19 CT scan dataset.

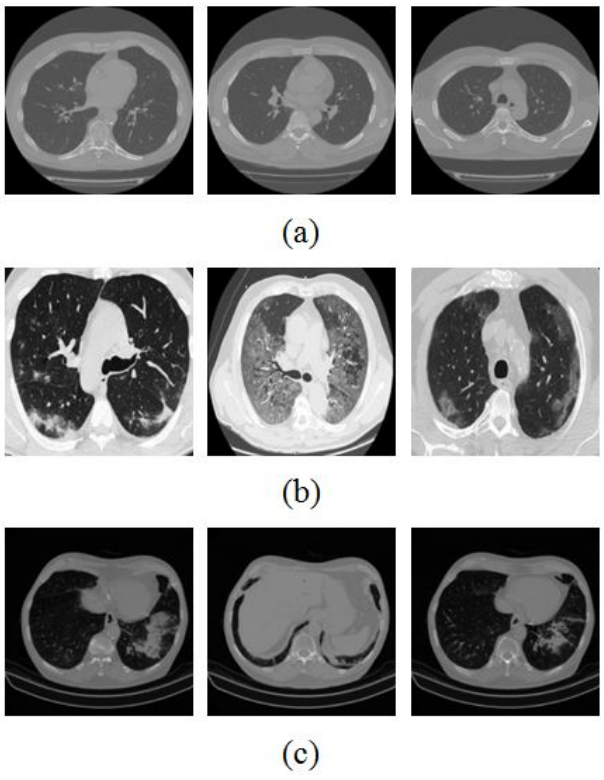


Figure 3.2: A sample of CT images from the dataset: (a) Normal, (b) COVID-19 and (c) Pneumonia.

3.2 Data Preprocessing

Following data collection, the images are preprocessed using several preprocessing techniques. These techniques could help with noise reduction and emphasizing sections of the image which will benefit during the model training. Image preprocessing is an important task for getting a sufficient result [37].

The size of the CT images should be adjusted to match the input dimensions of the CNN network, since various CNN models have varying input requirements. To accommodate the input requirements of the CNN network architectures that used in this research, all CT images are resized to (224 × 224) dimension before being used as input to the CNN networks. Also, these CT images are normalized by multiplying 1/255 by each pixel of the input image; this will normalize the image pixels in the range from 0 to 1 instead of 0 to 255.

Medical datasets are usually imbalanced due to limit of data for evaluating methods. The data distribution is very important during model training. In classification problems, an imbalanced dataset might produce biased predictions, where a balanced dataset can give better results. As shown in figure 3.3, the considered CT scan dataset is imbalanced. To alleviate imbalanced data problem, an undersampling technique was used by randomly removing instances of the majority classes to balance the data. Resample is employed independently to Covid-19, Normal and Pneumonia.

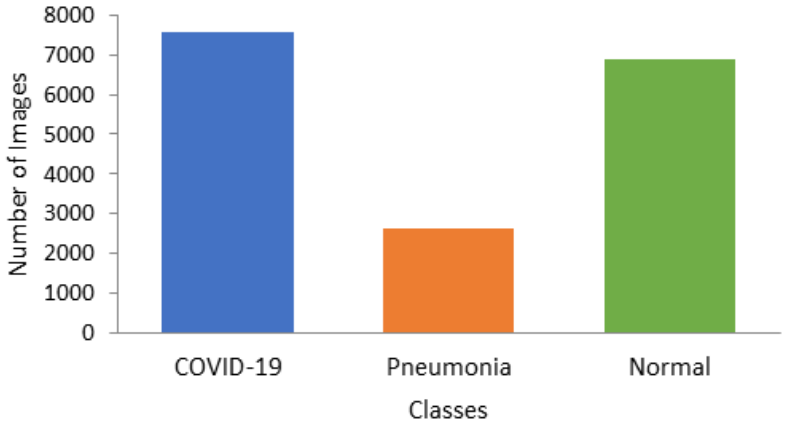


Figure 3.3: Dataset distribution.

3.3 Data Splitting

The most common techniques for splitting the data are percentage split and k-fold cross-validation. Percentage split is a simple way to split data into training, validation and test sets. In this work, the percentage split technique is used with 70:10:20 split ratio. From the total, 70% was used for training, 10% was used for validation, and 20% was used for testing. Training set is used for training the model, validation set is used to evaluate the model during the training process, testing set is used to provide an evaluation of the trained model. The same ratio of data splitting was used for both imbalanced and balanced datasets. The details of data splitting for initial and modified datasets are given in Table 3.2 and Table 3.3.

Type	**Training Set**	**Validation Set**	**Testing Set**	**Total**
Normal	4825	689	1379	6893
COVID-19	5315	759	1519	7593
Pneumonia	1833	262	523	2618

Table 3.2: Details of training, validation and testing set for the initial dataset.

Type	**Training Set**	**Validation Set**	**Testing Set**	**Total**
Normal	1750	250	500	2500
COVID-19	1750	250	500	2500
Pneumonia	1750	250	500	2500

Table 3.3: Details of training, validation and testing set for the modified dataset.

3.4 Model Selection

In this section, we present two different deep convolutional neural network (DCNN) models namely, VGG-16 and DenseNet201. We also describe the model architectures and transfer learning techniques used in these models for COVID-19 images classification.

3.4.1 VGG-16

VGG-16 is a convolutional neural network (CNN) architecture developed by Karen Simonyan and Andrew Zisserman [38]. VGG-16 model is a powerful model for image classification problema and is easy to use with a transfer learning approach. VGG-16 was used for ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014 and trained on ImageNet dataset that contains 1.4 million images in 1,000 different categories [38].

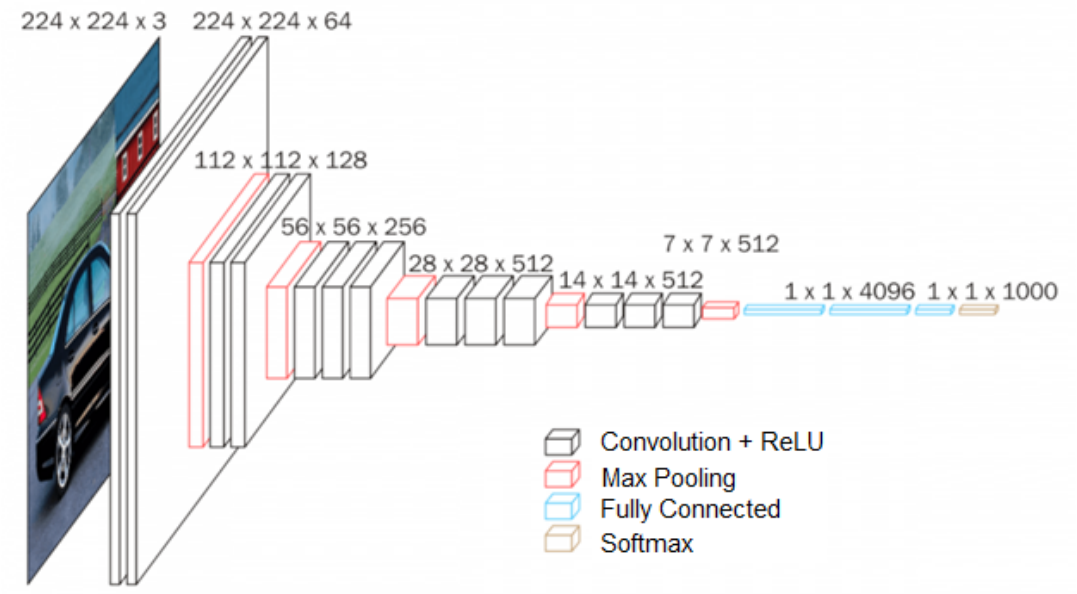


Figure 3.4: VGG-16 network architecture.

The architecture of VGG-16 is depicted in Figure 3.4. VGG-16 architecture contains two convolution layers of (3 × 3) filter with a stride 1, followed by max-pooling layer of (2 × 2) filter with a stride 2 repeated two times. Then, three convolution layers of (3 × 3) filter with a stride 1, followed by max-pooling layer of (2 × 2) filter with a stride 2 repeated three times. The combination of convolutional layers and max-pooling layer is called a convolutional layer block. At the end of its architecture, it has three fully connected (FC) layers.

The first two FC layers have 4096 units. The last FC layer has 1000 units with SoftMax activation function. Between these layers, a Rectified Linear Unit (ReLU) activation function is performed. The first convolution layer receives a (224 × 224)

image with three- RGB channels as input. Figure 3.5 shows VGG-16 network layers.

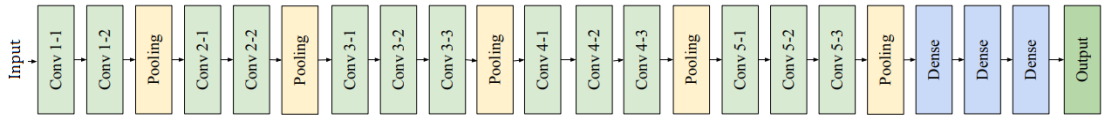


Figure 3.5: VGG-16 network layers.

3.4.2 DenseNet201

Dense Convolutional Network (DenseNet) is one of the CNN networks used for image classification [39]. DenseNet is composed of several dense convolutional blocks, and each dense block contains several layers densely connected, as illustrated in Figure 3.7. DenseNet is introduced as a solution for the vanishing gradient problem by adding feed-forward connections between layers, so the feature maps from each preceding layer are used as input into all next layers, as shown in Figure 3.6. According to the connection in DenseNet, the network connects each layer in the dense block to other subsequent layers in a feed-forward method. In other words, instead of a traditional CNN where is only L connection between the previous layer and the current layer, in DenseNet, there are L(L+1)/2 direct connections. In each dense block, a sequence of batch normalization, ReLU activation, and a (3×3) convolution is added, to reduce the number of input feature maps.

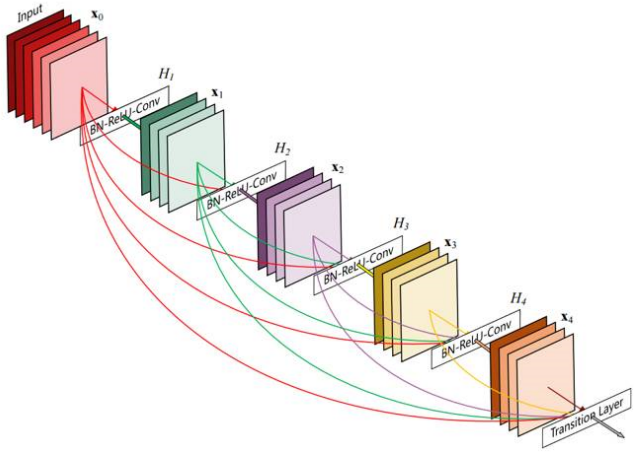


Figure 3.6: A dense block representing direct connections between layers [39].

Transition layer is used to connect each dense block to the next. The transition layer is composed of a batch normalization layer, a (1 × 1) convolutional layer, and (2 x 2) average pooling layer. The transition layers with average pooling are used each dense block, to perform down-sampling of the feature map from the previous dense block.

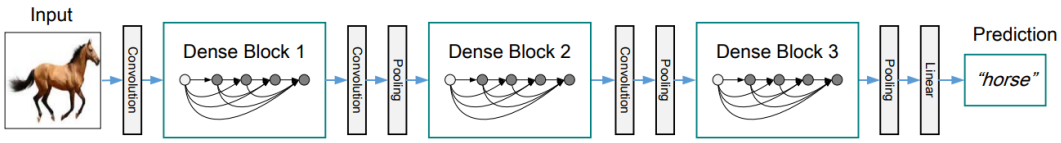


Figure 3.7: DenseNet with three dense blocks [39].

DenseNet has different architectures such as DenseNet121, DenseNet169, DenseNet201, and DenseNet264. In this project, we used DenseNet201 to detect and classify COVID-19 in CT images.

The DenseNet201 network architecture starts with a (7 × 7) convolution layer with a stride of 2 followed by a (2 × 2) max-pooling layer with a stride of 2, and followed by four dense blocks and three transition layers. The first dense block contains a (1 × 1) convolution and a (3 × 3) convolution repeated 6 times. The second dense block contains a (1 × 1) convolution and a (3 × 3) convolution repeated 12 times. The third dense block has the same layers as the second dense block, but is repeated 48 times. The fourth dense block has the same layers of dense blocks two and three, but is repeated 32 times. The transition layers contain a (1 × 1) convolution layer and a (2 × 2) average pooling layer with a stride of 2. Finally, the classification layers consist of a (7 × 7) global average pooling layer and FC layer with SoftMax activation function. The DenseNet201 network accepts images with (224 × 224) dimension as input. The DenseNet201 has a small number of parameters and provides of the best performances for image classification [40]. The architecture of DenseNet201 is shown in Figure 3.8.

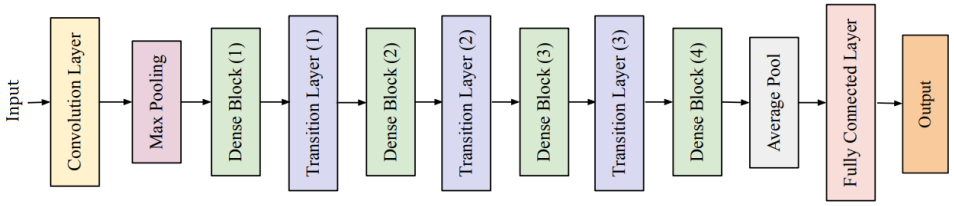


Figure 3.8: The architecture of DenseNet201.

3.5 Transfer Learning on the Models

VGG-16 and DenseNet-201 networks are available in Keras applications and can be used and instantiated [41].

First, the VGG-16 network is loaded with weights that pre-trained on ImageNet dataset. After that, the entire convolutional layer blocks (i.e., from the first convolutional layer to the last max-pooling layer) are frozen. Freezing preserves ImageNet weights from being updated during the training phase. Then, the top layers are removed (the last three FC layers) and replaced with additional new layers that fit with the classes of our case to enhance the learning ability of the mode in COVID-19 image classification.

The first added layer was a FC layer 1 with 512 units and a ReLU activation function. After the FC layer 1, a dropout layer was added next as a second layer with drop rate of 0.5 to avoid overfitting and divergence. After that, FC layer 2 was added with 256 units and the activation function ReLU. Finally, FC layer 3 was added as an output layer with a SoftMax activation function. The number of outputs in the last FC layer was equal to the number of classes, three in our case. This layer is used to predict output images for three classes: COVID-19, Pneumonia and Normal.

Similarly, the DenseNet201 network was loaded with weights from the model that pre-trained on ImageNet. Transfer learning was utilized by freezing the convolutional base layers (i.e., from the first convolutional layer to the last dense layer block). Moreover, the last layer was removed and replaced with additional new layers as classification layers that fit with the classes and our dataset.

The first added layer was the batch normalization layer. After the batch normalization layer, a FC layer 1 was added with 1024 units and ReLU activation function. A batch normalization layer was added after FC layer 1. After that, a dropout layer was added with a dropout rate of 0.5. Then, a FC layer 2 was added with 512 units

Layers	Output Shape	VGG-16
Conv x 2	224 × 224 × 64	3 × 3 conv, stride 1
Pool	112 × 112 × 64	2 × 2 max pool, stride 2
Conv x 2	112 × 112 × 128	3 × 3 conv, stride 1
Pool	56 × 56 × 128	2 × 2 max pool, stride 2
Conv x 3	56 × 56 × 256	3 × 3 conv, stride 1
Pool	28 × 28 × 256	2 × 2 max pool, stride 2
Conv x 3	28 × 28 × 512	3 × 3 conv, stride 1
Pool	14 × 14 × 512	2 × 2 max pool, stride 2
Conv x 3	14 × 14 × 512	3 × 3 conv, stride 1
Pool	7 × 7 × 512	2 × 2 max pool, stride 2
Flatten*	25088	25088
FC 1*	512	512 units, relu
Dropout*	512	50 percent
FC 2*	256	256 units, relu
FC 3*	3	3 category, softmax

Table 3.4: VGG-16 architecture after introducing new layers.

and ReLU activation function. Another batch normalization layer was then added after the FC layer 2. After the batch normalization layer, another dropout layer was added with a dropout rate of 0.5. Finally, a FC layer 3 was added as an output layer with SoftMax as an activation function. The number of outputs in the last FC layer was equal to the number of classes, three in our case. This layer is used to predict output images for three classes: COVID-19, Pneumonia and Normal.

The number and type of layers were chosen after studying different networks and performing many different experiments. Table 3.4 shows the layer used in the pre-trained VGG-16 model. From the table, all the base layers (i.e., from the first convolutional layer to the last max-pooling layer) are used as the pre-trained layers of the VGG-16 model. Table 3.5 shows the layer used in the pre-trained DensNet201 model. From the table, all the base layers (i.e., from the first convolutional to the last dense block) are used as the pre-trained layers of the DensNet201 model. The symbol (*) denotes the layers that were added to each model architecture for COVID-19 image classification.

Layers	**Output Size**	**DenseNet201**
Convolution	112 × 112	7 × 7 conv, stride 2
Pooling	56 × 56	3 × 3 max pool, stride 2
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 & \text{conv} \\ 3 \times 3 & \text{conv} \end{bmatrix} \times 6$
Transition layer (1)	56 × 56	1 × 1 conv
	28 × 28	2 × 2 average pool, stride 2
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 & \text{conv} \\ 3 \times 3 & \text{conv} \end{bmatrix} \times 12$
Transition layer (2)	28 × 28	1 × 1 conv
	14 × 14	2 × 2 average pool, stride 2
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 & \text{conv} \\ 3 \times 3 & \text{conv} \end{bmatrix} \times 48$
Transition layer (3)	14 × 14	1 × 1 conv
	7 × 7	2 × 2 average pool, stride 2
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 & \text{conv} \\ 3 \times 3 & \text{conv} \end{bmatrix} \times 32$
Average Pooling*	1 × 1	7 × 7 global average pool
Batch Normalization*	1920	-
FC 1*	1024	1024 units, relu
Batch Normalization*	1024	-
Dropout*	1024	50 percent
FC 2*	512	512 units, relu
Batch Normalization*	512	-
Dropout*	512	50 percent
FC 3*	3	3 category, softmax

Table 3.5: DensNet201 architecture after introducing new layers.

3.6 Fine-tuning and Training the Models

A two-step process is used to train the VGG-16 and DenseNet201 models. In the first step, we froze all pre-trained layers of the models and trained only the new layers in TensorFlow for 5 epochs using an Adam [42] optimizer with a learning rate (LR) of 10^{-3}. This phase enhances the learning of the features in the considered dataset.

In the second step, to increase the performance of the models we fine-tuned and unfroze the last pre-trained layers of the models (i.e., last convolutional layer block of the pre-trained VGG-16 model and the last dense block of the pre-trained DenseNet201 model) and trained along with the new layers for further 20 epochs with Adam optimizer and LR is reduced to 10^{-5}. The batch size of 128 is used in the training process. In both phases of training, the activation function is ReLU, the dropout rate is 0.5, and the output layer is SoftMax. The cross entropy loss function is used to minimize the loss. The adam optimizer is used to update the weights in the model based on the loss. The number of trainable parameters and non-trainable parameters in the models is given in Table 3.6. The hyper-parameters values used to train the models are given in Table 3.7.

Parameters	**VGG-16**	**DenseNet201**
Total Parameters	27,692.355	20,829.251
Trainable Parameters	25,956,867	3,609,475
Non-Trainable Parameters	1,735,488	17,219,776

Table 3.6: The trainable parameters and non-trainable parameters in the models.

Parameter	**Values**
Learning rate	$10^{-3} - 10^{-5}$
Batch size	128
Activation function	Softmax (Final Classification Layer)
Total Epochs	25
Optimizer	Adam

Table 3.7: The hyper-parameters values used to train the models.

The Learning Rate (LR) is a hyper-parameter that controls how quickly model weights are adjusted. A lower LR may provide more accurate weights (up to convergence), but it requires more computation time. A large LR may cause the model to converge too rapidly [43]. The number of epochs is a hyper-parameter that defines the number of times the learning algorithm is transported through the training dataset. Batch size refers to the number of training examples utilized in each batch.

Chapter 4

Experimental Results

In this chapter, we present the evaluation metrics used to evaluate the models performance. We present several experiment results on the datasets. We also discuss the performance results of the DCNN models.

The proposed work was implemented using Python programing language, Scikit-learn libraries, Keras libraries, and TensorFlow frameworks. The pre-trained models used in this study were downloaded from the Keras application website [41]. The experiments were conducted on a Google Colaboratory platform with a Tesla P100–PCIE 16 GB graphics card, an Intel(R) Xeon(R) CPU with a 2.20 GHz, 16 GB of RAM, and a 256 GB hard disk.

4.1 Evaluation Metrics

Performance evaluation metrics are very important for evaluating the performance of CNN models. The performance of each model on the test dataset is evaluated and compared based on seven performance metrics: confusion matrix, accuracy, precision, sensitivity (recall), F1-score, area under the receiver operating characteristic curve (AUC-ROC). The performance metrics used to evaluate the DL models are listed below.

4.1.1 Confusion Matrix

The confusion matrix is a performance evaluation metric for classification algorithms. The classification problems can be a binary or multi-class classification. The confusion matrix provides the exact value for the four combinations, True Positive (TP), False

Positive (FP), True Negative (TN), and False Negative (FN) by comparing the actual target values with predicted values from the classification algorithm. It gives us a holistic view about the performance of the classification model and the kinds of errors it is making [44]. The sample confusion matrix of a binary classification problem is shown in Figure 4.1.

- **True Positive (TP)**: The predicted value was correctly predicted. The actual value was positive, and the model predicted as positive.
- **True Negative (TN)**: The predicted value was correctly predicted. The actual value was negative, and the model predicted as negative.
- **False Positive (FP)**: The predicted value was incorrectly predicted. The actual value was negative, but the model predicted as positive.
- **False Negative (FN)**: The predicted value was incorrectly predicted. The actual value was positive, but the model predicted as negative.

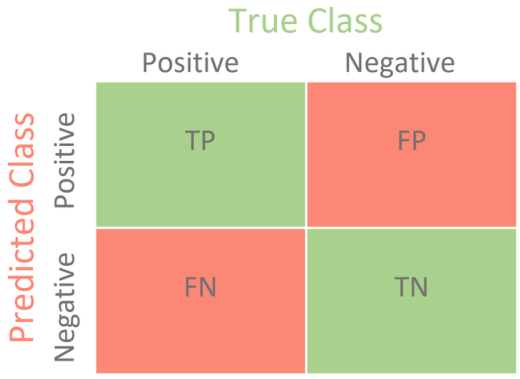


Figure 4.1: Confusion matrix [44].

The accuracy, precision, sensitivity (recall), specificity (selectivity) and F1-score can be calculated using the confusion matrix as follows:

Accuracy: Accuracy is the number of correct predictions out of the total number of predictions and can be calculated by the sum of true positive and true negative divided by the sum of true positive, true negative, false positive, and false negative and is given by:

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

Recall: Recall gives the ratio of true positive to the sum of true positive and false negative and is given by:

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

Precision: Precision can be defined as the ratio of the true positive to the sum of true positive and false positive and is given by:

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

Specificity: Specificity can be defined as the ratio of the true negative to the sum of true negative and false positive and is given by:

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

F1-score: F1-score can be defined as the harmonic mean of the recall and precision

$$F1 - score = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{4.5}$$

For multi-class classification problems, the macro average and weighted average of metrics can be used to evaluate the overall model performance. The macro average is the average of the metric (i.e., accuracy, recall, precision, F1-score) over all classes, while the weighted average is the average of the metric over classes weighted by sample size. If each class has the same sample size, the macro average and the weighted average are the same.

4.1.2 ROC Curve and AUC

The receiver operating characteristic curve (ROC) is a graph showing the performance of a binary and multi-classification model. This curve plots the ratio of true positive

rate (TPR) against false positive rate (FPR) at several classification thresholds, where TPR is known as a recall, and FPR can be calculated as (1 - specificity). For a binary classification model, the ROC curve would be a diagonal line from (0, 0) to (1, 1). Any curve that is above the diagonal line means the classification model is better than random, while a curve that is below the diagonal line indicates a bad classification model, which is worse than random. A perfect binary classification model would yield a straight line from (0, 1) to (1, 1), meaning 100% sensitivity and 100% specificity [45].

Area under curve (AUC) refers to the area under the ROC curve, which is a range in value from 0 to 1. According to the ROC curve principle, the classification model performs better when the AUC is larger. A model whose predictions are 100% wrong has an AUC of 0, while a model whose predictions are 100% correct has an AUC of 1 [45]. An example of ROC curve and AUC are shown in Figure 4.2.

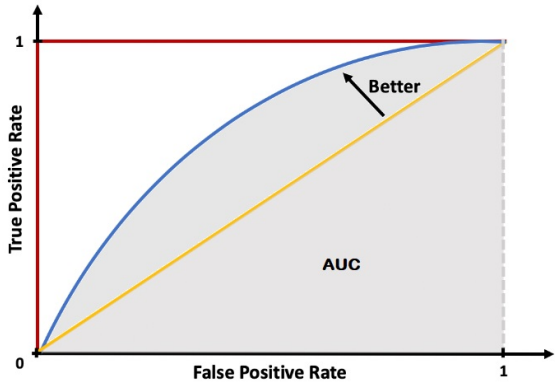


Figure 4.2: ROC curve and AUC: red line: a perfect classifier, blue curve: a good classifier, yellow line: a random classifier, and shaded area: AUC for the good classifier.

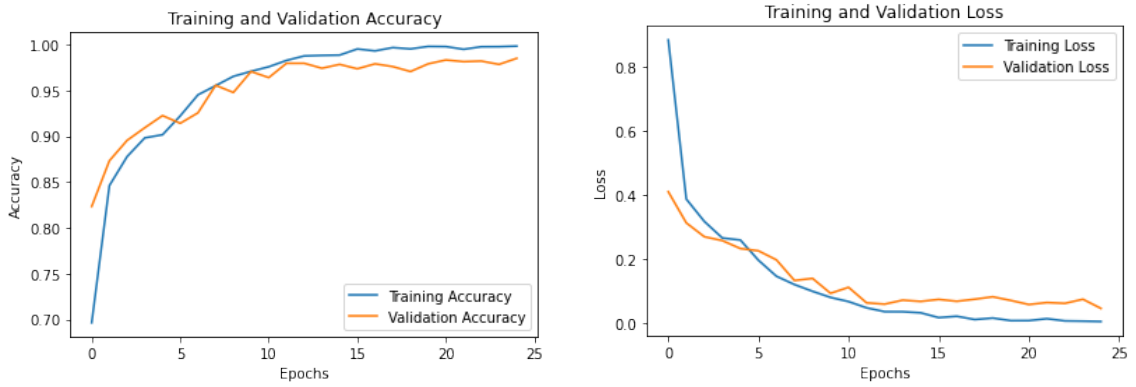
4.2 Performance of deep learning models on the initial dataset

Initial (Imbalanced) dataset was used to analyze the training performance of the DCNN models in terms of important parameters, i.e., training accuracy, validation accuracy, training loss, and validation loss at different epochs. These parameters were calculated and determined to estimate the overfitting of the trained models. Furthermore, confusion matrices for DCNN models were generated in order to calculate the

performance metrics, i.e., accuracy, recall, precision, and F1-score. The performance of VGG-16 and DenseNet201 models for the initial dataset were individually evaluated and discussed in this section.

4.2.1 VGG-16 Results

VGG-16 was trained with the parameter values LR = 0.001 for first 5 epochs, LR = 0.0001 for further 20 epochs, batch size = 128, total epochs = 25 and Adam optimizer. The computational time was around 30 min 12 s. After training the last layers of VGG-16 for 5 epochs, the training and validation accuracy was between 0.90 and 0.92, the training and validation loss was between 0.23 and 0.26. This result was further improved upon by unfreezing all layers in VGG-16 network. After the model was unfrozen and trained for further 20 epochs, the model achieved a training accuracy of 0.998, validation accuracy of 0.985, training loss of 0.015, and validation loss of 0.047. The graphs of the learning curves of the VGG-16 are illustrated in Figure 4.3. As shown, the model achieved high performance with only 25 epochs. It can be shown that the validation loss was low and slightly higher than the training loss, which means the model fits well on the dataset.



(a) (b)

Figure 4.3: Learning curves of the VGG-16: (a) training and validation accuracy (b) training and validation loss.

The confusion matrix was obtained using the testing dataset that contains 3421 images of COVID-19, Pneumonia and normal samples. The confusion matrix of VGG-16 model is shown in Figure 4.4. Labels are shown Normal class, COVID-19 class, and Pneumonia, respectively. The blocks located on the diameter of the matrix show

how well each class is distinguished. The confusion matrix shows that VGG-16 model is correct in 3368 images out of 3421 images, 31 images of COVID-19 incorrectly predicted as Normal, one image of COVID-19 incorrectly predicted as Pneumonia, and 21 images of Normal incorrectly wrongly predicted as COVID-19.

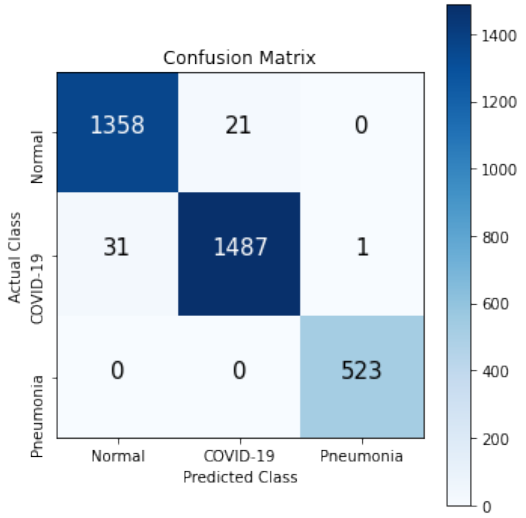


Figure 4.4: VGG-16 Confusion matrix.

Table 4.1 gives the test results of VGG-16 model. The table shows that VGG-16 model achieves an average accuracy of 98.45%, precision of 98.72%, recall of 98.47%, and F1-score of 0.98.74%. Normal has precision, recall, and F1-score of 97.76%, 98.47%, and 98.1%, respectively. COVID-19 has precision, recall, and F1-score of 98.6%, 97.89%, and 98.23%, respectively. Pneumonia has precision, recall, and F1-score of 99.8%, 100%, and 99.89%, respectively.

Figure 4.5 shows the ROC curves and AUC scores for the VGG-16 model. The probability measurements of the true positive rate against the false positive rate show the distinguishability among classes. The more area under the ROC curve is the more the ability of the model to show the sensitivity to distinguish the classes correctly. For Normal class, the AUC score is 0.9984, 9984 for COVID-19 class, and 1.0000 for Pneumonia class.

Label	Accuracy	Precision	Recall	F1-score	Support
Normal		0.9776	0.9847	0.9810	1379
COVID-19		0.9860	0.9789	0.9823	1519
Pneumonia		0.9980	1.0000	0.9989	523
Macro average	0.9845	0.9872	0.9878	0.9874	3421
Weighted average	0.9845	0.9844	0.9844	0.9883	3421

Table 4.1: Performance of VGG-16 on the initial dataset.

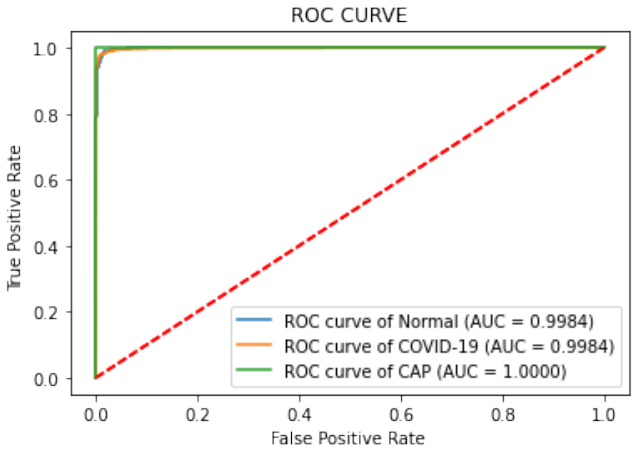
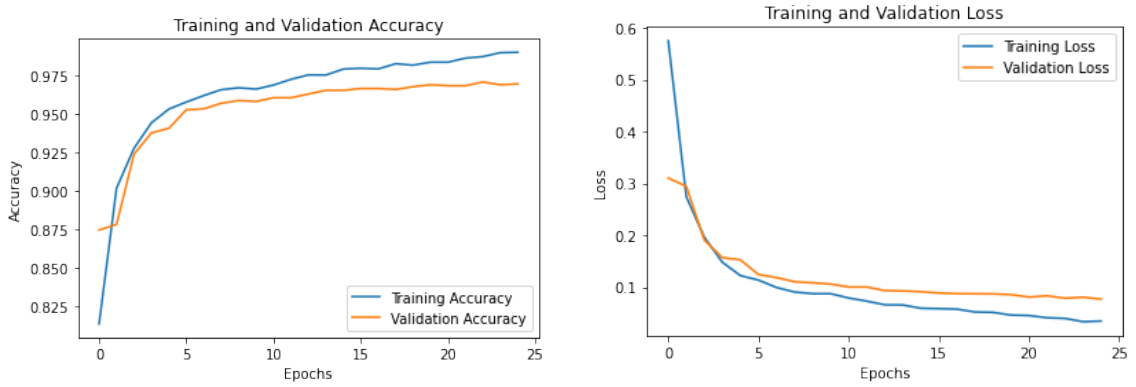


Figure 4.5: ROC curve and AUC score for VGG-16.

4.2.2 DenseNet201 Results

DenseNet201 was trained for 25 epochs with the parameter values LR = 0.001 for 5 ephocs, decreased to 0.0001 for further 20 epochs, batch size = 128 and Adam optimizer. The computational time was around 28 min 30 s. After running DenseNet201 network on training and validation datasets for 5 epochs, the training accuracy was 0.953, validation 0.94, the training loss was 0.122 and the validation loss 0.153. These results were further improved at 25 epochs. The training accuracy and validation accuracy were increased to 0.989 and 0.969, the training loss was dropped to 0.033, and the validation loss to 0.076. The graphs of training accuracy and validation accuracy vs. training loss and validation loss are illustrated in Figure 4.6. It is observed that the model is good fit on the data and not suffered from overfitting.



(a) (b)

Figure 4.6: Learning curves of the DenseNet201: (a) training and validation accuracy (b) training and validation loss.

Figure 4.7 shows confusion matrix of the DenseNet201 model. The confusion matrix shows that the DenseNet201 model correctly classifies 3318 images out of 3421 images. However, 67 images of COVID-19 are wrongly classified as Normal, and three images infected with COVID19 are wrongly classified as Pneumonia. On the other hand, 27 images of Normal are misclassified as COVID-19, one image of Normal is misclassified as Pneumonia, and five images of Pneumonia are misclassified as a COVID-19.

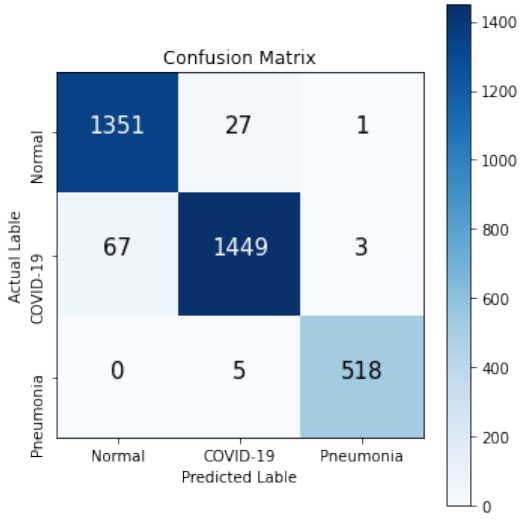


Figure 4.7: DenseNet201 Confusion matrix.

Table 4.2 gives the test results of DenseNet201 model. The table shows that DenseNet201 model achieves an average accuracy of 96.98%, precision of 97.44%, recall of 97.46%, and F1-score of 97.43%. Normal has precision, recall, and F1-score of 95.27%, 97.96%, and 96.59%, respectively. COVID-19 has precision, recall, and F1-score of 97.83%, 95.39%, and 96.59%, respectively. Pneumonia has precision, recall, and F1-score of 99.23%, 99.04%, and 99.13%, respectively.

Label	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Support**
Normal		0.9527	0.9796	0.9659	1379
COVID-19		0.9783	0.9539	0.9659	1519
Pneumonia		0.9923	0.9904	0.9913	523
Macro average	0.9698	0.9744	0.9746	0.9743	3421
Weighted average	0.9698	0.9701	0.9698	0.9697	3421

Table 4.2: Performance of the DenseNet201 on the initial dataset.

ROC curves and AUC scores are generated to evaluate the model's ability to separate each class from other classes. The AUC score is 0.9949, the AUC score is 9947 for COVID-19 class, and the AUC score is 1.0000 for CAP class. Figure 4.8 shows three ROC curves and AUC scores for the DenseNet201 model.

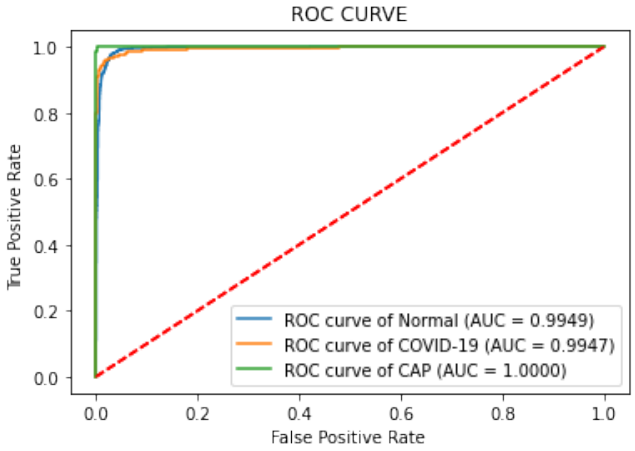


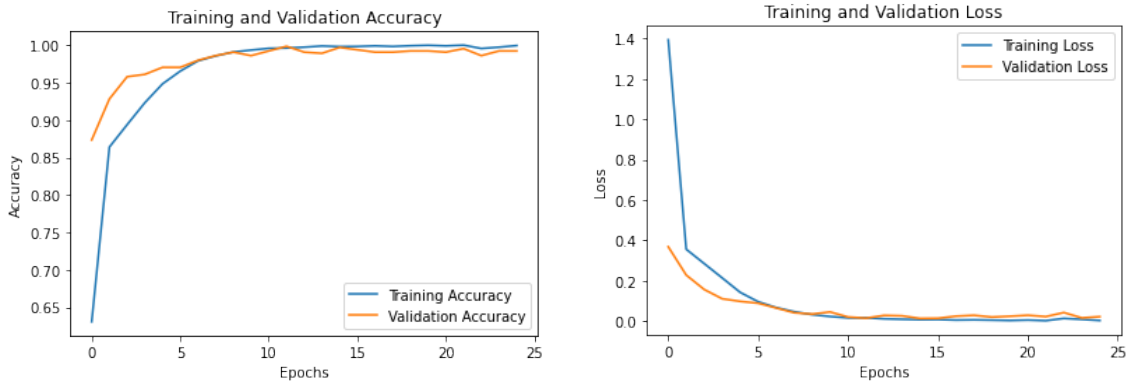
Figure 4.8: ROC curve and AUC score for DenseNet201.

4.3 Performance of deep learning models on the modified dataset

In this section, the evaluation results of the deep learning classification models on the modified (balanced) dataset were examined and explained in detail. Individually, the performance of VGG-16 model and DenseNet201 model on the modified dataset was evaluated based on performance metrics: i.e., accuracy, precision, recall, F1-score, and AUC score.

4.3.1 VGG-16 Results

VGG-16 was trained using the training set on the modified dataset with the parameter values LR = 0.001 for first 5 epochs, LR = 0.0001 for further 20 epochs, batch size = 128, total epochs = 25 and Adam optimizer. The computational time was around 13 min 50 s. After training the last layers of VGG-16 for 5 epochs, the training and validation accuracy was between 0.94 and 0.96, the training and validation loss was between 0.14 and 0.11. This result was increased at 25 epochs. At 25 epochs, the model achieved a training accuracy of 0.9998, validation accuracy of 0.9925, training loss of 0.0018, and validation loss of 0.015. The graphs of the learning curves of the VGG-16 are illustrated in Figure 4.9. As shown, the model achieved high performance at 22 epochs.



(a) (b)

Figure 4.9: Learning curves of the VGG-16 on the modified dataset: (a) training and validation accuracy (b) training and validation loss.

Figure 4.10 shows the confusion matrix of VGG-16 model. The result was obtained using the testing set on the modified dataset. Labels are shown Normal class, COVID-19 class, and Pneumonia, respectively. As shown, the blocks on the diameter are almost the same, which means the classes are classified more accurately when the number of samples in each class is balanced. The confusion matrix shows that the VGG-16 model correctly predicts 1491 images among a total of 1500 images. However, three images of COVID-19 are wrongly classified as Normal, one image of COVID-19 is wrongly classified as Pneumonia, and five images of Normal are wrongly classified as COVID-19. Therefore, from the total of 1500 test data, the VGG-16 wrongly predicts only eight images.

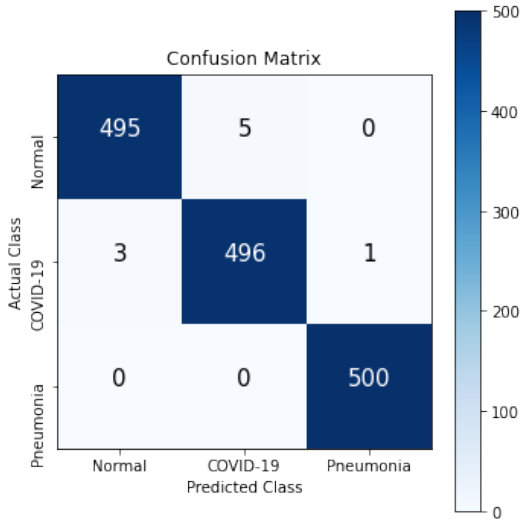


Figure 4.10: VGG-16 Confusion matrix for the modified dataset.

Table 4.3 summarizes the performance evaluation results of the VGG-16 model on the modified dataset based on the accuracy, precision, recall, and F1-score. This shows that VGG-16 model achieves an average accuracy of 99.4%, with an average precision, recall, and F1-score of 99.39%, 99.4%, and 99.39%, respectively. This result shows that the VGG-16 in the modified dataset is performing better than the VGG-16 in the initial dataset for COVID-19 image classification.

Figure 4.11 shows the ROC curves and AUC scores for the VGG-16 model on the modified dataset. It shows the ROC curves for the three classes. From the ROC curve, it can be seen that the AUC score is 0.9998 for Normal class, the AUC score is 0.9997 for COVID-19 class, and the score is 0.9999 for Pneumonia class.

Label	Accuracy	Precision	Recall	F1-score	Support
Normal		0.9939	0.9900	0.9919	500
COVID-19		0.9900	0.9920	0.9909	500
Pneumonia		0.9980	1.0000	0.9989	500
Macro average	0.9940	0.9939	0.9940	0.9939	1500
Weighted average	0.9940	0.9939	0.9940	0.9939	1500

Table 4.3: Performance of the VGG-16 on the modified dataset.

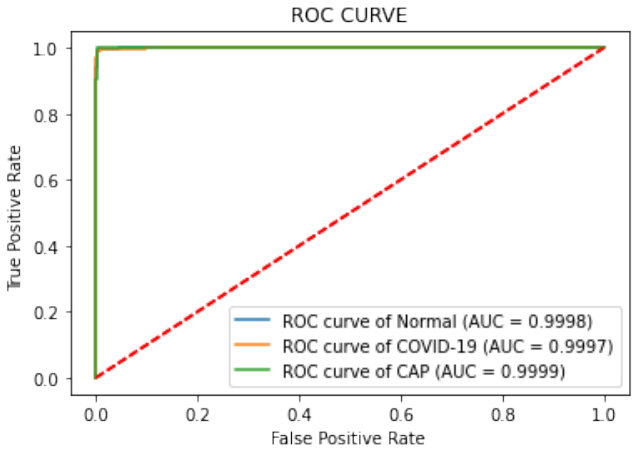
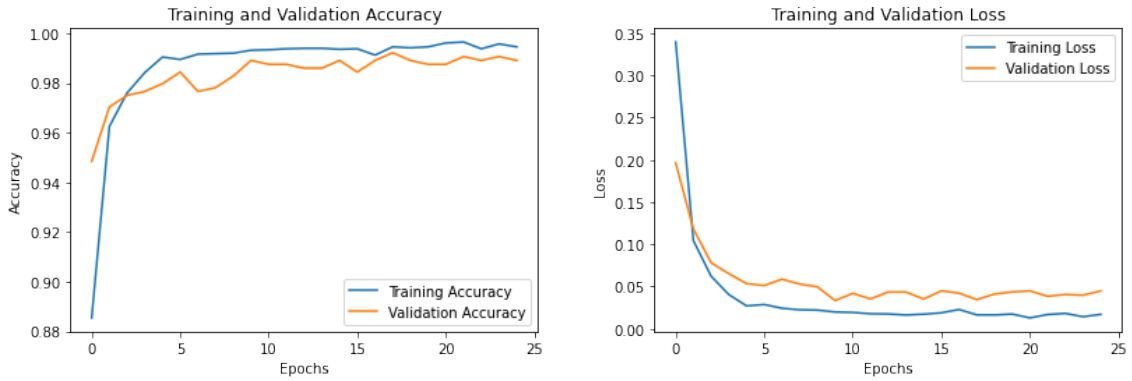


Figure 4.11: ROC curve and AUC score for VGG-16 on the modified dataset.

4.3.2 DenseNet201 Results

DenseNet201 was trained using the training set on modified dataset for 25 epochs with the parameter values LR = 0.001 for 5 ephocs, decreased to 0.0001 for further 20 epochs, batch size = 128 and Adam optimizer. The computational time was around 13 min 35 s. After running DenseNet201 for 5 epochs, the training accuracy was 0.975, validation was 0.974, and the training loss was 0.0623 and the validation loss 0.0782. At 25 epochs, the training accuracy and validation accuracy were increased to 0.9945 and 0.9906, the training loss was dropped to 0.014, and the validation loss was dropped to 0.038. The graphs of training accuracy and validation accuracy vs. training loss and validation loss are illustrated in Figure 4.12.



(a) (b)

Figure 4.12: Learning curves of the DenseNet201 on the modified dataset: (a) training and validation accuracy (b) training and validation loss.

Figure 4.13 shows the confusion matrix for the DenseNet201 model on modified dataset. The blocks on the diameter are almost the same. This confirms the effectiveness of the balanced dataset compared with the initial dataset. The confusion matrix shows that the DenseNet201 model wrongly predicts 13 images out of 1500 images; four images of COVID-19 are misclassified as Normal, and one image of COVID-19 is misclassified as Pneumonia. On the other hand, seven images of Normal are misclassified as COVID-19, and one image of Normal is misclassified as Pneumonia. Therefore, the VGG-16 correctly predicts 1487 from the total of 1500 test data.

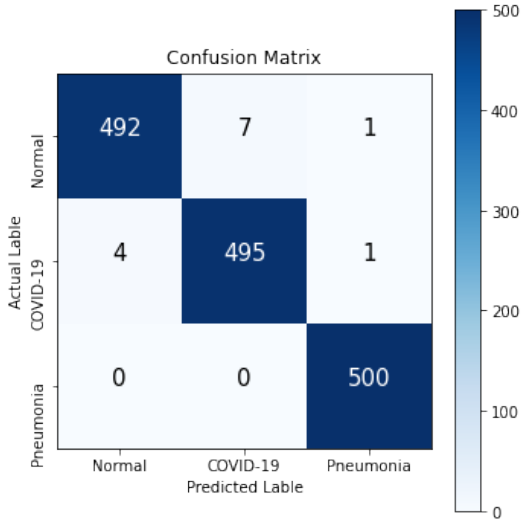


Figure 4.13: DenseNet201 Confusion matrix for the modified dataset.

Table 4.4 summarizes the performance evaluation results of DenseNet201 model on the modified dataset. This shows that DenseNet201 model achieves an average accuracy of 99.13%, with an average precision, recall, and F1-score of 99.13%, 99.13%, and 99.12%, respectively. This result proves that the DenseNet201 in the modified dataset is superior to the DenseNet201 in the initial dataset.

Label	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Support**
Normal		0.9919	0.9840	0.9879	500
COVID-19		0.9860	0.9900	0.9879	500
Pneumonia		0.9960	1.0000	0.9979	500
Macro average	0.9913	0.9913	0.9913	0.9912	1500
Weighted average	0.9913	0.9913	0.9913	0.9912	1500

Table 4.4: Performance of the DenseNet201 on the modified dataset.

Figure 4.14 shows the ROC curves and AUC scores for the DenseNet201 model on modified dataset. It shows the ROC curves for each class. it can be seen that the AUC score for Normal class is 0.9991, for COVID-19 class is 0.9992, and for Pneumonia is 0.9997. These scores also prove that the DenseNet201 in modified dataset is performing better than the DenseNet201 in initial dataset.

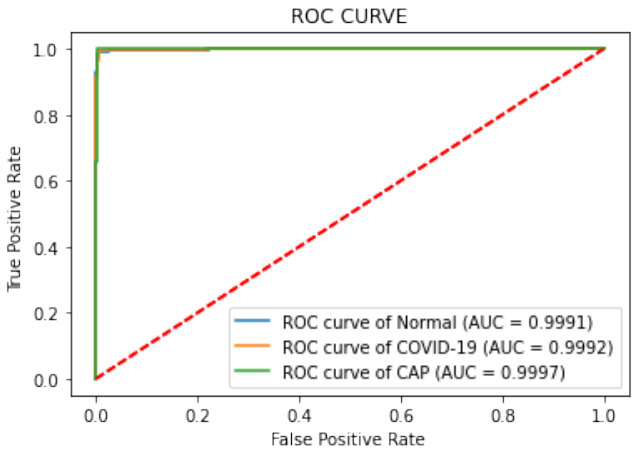


Figure 4.14: ROC curve and AUC score for DenseNet201 on the modified dataset.

4.4 Discussion and Comparison

In this section, the performance results of DCNN models for initial and balanced datasets were compared.

For the initial dataset, VGG-16 model performed better than DesneNet201 model in terms of accuracy, precision, recall, and F1-score. However, in terms of computational time, DesneNet201 model required approximately 1710 s, whereas VGG-16 model required approximately 1812 s. Table 3.6 shows that the DesneNet201 model has fewer trainable parameters; hence it takes less time to train than the VGG-16 model. The average accuracy, precision, recall, and F1-score of 98.45%, 98.72%, 98.47%, and 98.74% for VGG-16 model and 96.98%, 97.44%, 97.46%, 97.43% for the DesneNet201 model (Table 4.1) and (Table 4.2).

For the modified dataset, VGG-16 model had the best performance in terms of accuracy, precision, recall, and F1-score. The average accuracy, precision, recall, and F1-score were 99.4%, 99.39%, 99.4% and 99.39% for VGG-16 model (Table 4.3). However, DenseNet201 model obtained good results in terms of computational time, where it only required 815 s. The average accuracy of DenseNet201 model was 99.13%, with average precision, recall, and F1-score of 99.13%, 99.13%, and 99.12% (Table 4.14).

Overall, VGG-16 model performed better than DesneNet201 model in terms of accuracy, precision, recall, and F1-measure with both imbalanced and balanced datasets. However, DesneNet201 model was faster than VGG-16 model in terms of execution time. The results discussed above are given in Table 4.5, which shows how VGG-16 and DesneNet201 were improved.

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
DesneNet201-Initial	96.98%	97.44%	97.46%	97.43%
VGG16-Initial	98.45%	98.72%	98.47%	98.74%
DesneNet201- Modified	99.13%	99.13%	99.13%	99.12%
VGG16- Modified	99.40%	99.39%	99.40%	99.39%

Table 4.5: Performance comparison of VGG-16 and DesneNet201.

We also compared the results of our suggested method with other methods that used the same dataset created and collected by Maftouni et al [36]. Table 4.6 shows the performance of our proposed VGG-16 and DenseNet201 models compared with other deep learning models. It is obvious from the table below that the proposed VGG-16 model achieved better results compared with the reported results from the literature in terms of accuracy, precision , recall, and F1-score. In addition, the proposed model was trained on a balanced dataset including the same number of images, making it more robust and easier to tackle this pandemic in the near future.

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
Ense with FC [36]	95.07%	98.32%	89.84%	93.89%
Ense with FC + SVM [36]	95.31%	97.93%	90.80%	94.23%
LACNN [46]	98.3%	98.7%	98.5%	98.6%
LACNN_CBAM [46]	98.93%	99.15%	99.16%	99.15%
Proposed DesneNet201	99.13%	99.13%	99.13%	99.12%
Proposed VGG-16	99.40%	99.39%	99.40%	99.39%

Table 4.6: Comparison of proposed models with other deep learning models.

Chapter 5

Conclusions

A deep convolution neural network based framework was employed in this project for the detection and classification of COVID-19 using chest CT radiography images. Two pre-trained models i.e., VGG-16 and DenseNet201 based on transfer learning technique with fine-tuning were trained and tested on multi-class COVID-19 CT images dataset that contained COVID-19, Pneumonia, and Normal cases. Under-sampling technique was used to create a balanced dataset. Various experiments were performed to evaluate the performance of the pre-trained models using evaluation metrics, i.e., confusion matrix, accuracy, recall, precision, F1-score, and AUC. Experiments results suggested the effectiveness of the balanced dataset in achieving better results compared with the imbalanced dataset. The results show that the most accurate model was VGG-16 model. It achieved average accuracy, recall, precision, F1-score, and AUC score of 99.4%, 99.39%, 99.4%, 99.39%, and 99.93%, respectively. Furthermore, DenseNet201 model achieved average accuracy, recall, precision, F1-score, and AUC score of 99.13%, 99.13%, 99.13%, 99.12%, and 99.93%, respectively. The proposed models have fewer parameters and low complexity compared to other deep learning models. The experiment results of the proposed method show that it is superior to the existing deep learning methods. We believe that the models proposed in this research can help healthcare professionals and physicians to diagnose COVID-19 disease effectively in practice.

5.1 Future Work

For future work, other datasets can be used to evaluate the performance of the CNN models. Different hyper-parameters such as optimizer, learning rate, activation functions, batch size, and dropout value can be utilised for further analysis. For data balancing, other techniques such as oversampling technique and Synthetic Minority Oversampling Technique (SMOTE) can be used to generate a balanced dataset. Instead of using a percentage split, K-fold cross-validation can be employed to obtain training, validation and testing sets. Other DL methods can also be used for evaluation and the results compared to those in this report.

Bibliography

[1] C. Huang, Y. Wang, X. Li, L. Ren, J. Zhao, Y. Hu, L. Zhang, G. Fan, J. Xu, X. Gu, *et al.*, "Clinical features of patients infected with 2019 novel coronavirus in wuhan, china," *The lancet*, vol. 395, no. 10223, pp. 497–506, 2020.

[2] W. H. Organization *et al.*, "World health organization (who) coronavirus (covid-19) dashboard (2022)," *World Health Organization, URL: https://covid19.who.int/.(Accessed: 24-August-2022)*, 2022.

[3] T. Singhal, "A review of coronavirus disease-2019 (covid-19)," *The indian journal of pediatrics*, vol. 87, no. 4, pp. 281–286, 2020.

[4] N. Jawerth, "How is the covid-19 virus detected using real time rt-pcr," *IAEA Bulletin*, vol. 61, no. 2, pp. 8–11, 2020.

[5] Z. Y. Zu, M. D. Jiang, P. P. Xu, W. Chen, Q. Q. Ni, G. M. Lu, and L. J. Zhang, "Coronavirus disease 2019 (covid-19): a perspective from china," *Radiology*, vol. 296, no. 2, pp. E15–E25, 2020.

[6] H. Mukherjee, S. Ghosh, A. Dhar, S. M. Obaidullah, K. Santosh, and K. Roy, "Deep neural network to detect covid-19: one architecture for both ct scans and chest x-rays," *Applied Intelligence*, vol. 51, no. 5, pp. 2777–2789, 2021.

[7] 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.

[8] W. Kong and P. P. Agarwal, "Chest imaging appearance of covid-19 infection," *Radiology: Cardiothoracic Imaging*, vol. 2, no. 1, 2020.

[9] M. T. Islam, B. N. K. Siddique, S. Rahman, and T. Jabid, "Image recognition with deep learning," in *2018 International conference on intelligent informatics and biomedical sciences (ICIIBMS)*, vol. 3, pp. 106–110, IEEE, 2018.

[10] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," *Advances in neural information processing systems*, vol. 26, 2013.

[11] S.-H. Wang, D. R. Nayak, D. S. Guttery, X. Zhang, and Y.-D. Zhang, "Covid-19 classification by ccshnet with deep fusion using transfer learning and discriminant correlation analysis," *Information Fusion*, vol. 68, pp. 131–148, 2021.

[12] L. Sarker, M. M. Islam, T. Hannan, and Z. Ahmed, "Covid-densenet: a deep learning architecture to detect covid-19 from chest radiology images," *Preprint*, vol. 2020050151, 2020.

[13] Z. Wang, Q. Liu, and Q. Dou, "Contrastive cross-site learning with redesigned net for covid-19 ct classification," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 10, pp. 2806–2813, 2020.

[14] 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.

[15] L. Li, L. Qin, Z. Xu, Y. Yin, X. Wang, B. Kong, J. Bai, Y. Lu, Z. Fang, Q. Song, *et al.*, "Artificial intelligence distinguishes covid-19 from community acquired pneumonia on chest ct," *Radiology*, 2020.

[16] K. El Asnaoui and Y. Chawki, "Using x-ray images and deep learning for automated detection of coronavirus disease," *Journal of Biomolecular Structure and Dynamics*, vol. 39, no. 10, pp. 3615–3626, 2021.

[17] 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.

[18] A. Jaiswal, N. Gianchandani, D. Singh, V. Kumar, and M. Kaur, "Classification of the covid-19 infected patients using densenet201 based deep transfer learning," *Journal of Biomolecular Structure and Dynamics*, vol. 39, no. 15, pp. 5682–5689, 2021.

[19] C. Zheng, X. Deng, Q. Fu, Q. Zhou, J. Feng, H. Ma, W. Liu, and X. Wang, "Deep learning-based detection for covid-19 from chest ct using weak label," *MedRxiv*, 2020.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] S. Saha, "A comprehensive guide to convolutional neural networks—the eli5 way," *Towards data science*, vol. 15, 2018.

[22] M. Shahid, "Convolutional neural network."," *Towards Data Science*, 2019.

[23] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[24] M. Yani *et al.*, "Application of transfer learning using convolutional neural network method for early detection of terry's nail," in *Journal of Physics: Conference Series*, vol. 1201, p. 012052, IOP Publishing, 2019.

[25] K. Patel, "Convolutional neural networks—a beginner's guide," *Toward Data Science*, 2019.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[27] J. Brownlee, "A gentle introduction to dropout for regularizing deep neural networks," *Machine Learning Mastery*, vol. 3, 2018.

[28] J. Brownlee, "A gentle introduction to batch normalization for deep neural networks," *Machine Learning Mastery; Machine Learning Mastery: San Juan, Puerto Rico*, 2019.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[30] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[31] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *towards data science*, vol. 6, no. 12, pp. 310–316, 2017.

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[33] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*, pp. 270–279, Springer, 2018.

[34] S. Asif, M. Zhao, F. Tang, and Y. Zhu, "A deep learning-based framework for detecting covid-19 patients using chest x-rays," *Multimedia Systems*, pp. 1–19, 2022.

[35] Maftouni, "Large covid-19 ct scan slice dataset." `https://www.kaggle.com/datasets/maedemaftouni/large-covid19-ct-slice-dataset`, 2021. [Online; accessed 20-October-2021].

[36] M. Maftouni, A. C. C. Law, B. Shen, Z. J. K. Grado, Y. Zhou, and N. A. Yazdi, "A robust ensemble-deep learning model for covid-19 diagnosis based on an integrated ct scan images database," in *IIE Annual Conference. Proceedings*, pp. 632–637, Institute of Industrial and Systems Engineers (IISE), 2021.

[37] S. Perumal and T. Velmurugan, "Preprocessing by contrast enhancement techniques for medical images," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 18, pp. 3681–3688, 2018.

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

[39] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[40] X. Yu, N. Zeng, S. Liu, and Y.-D. Zhang, "Utilization of densenet201 for diagnosis of breast abnormality," *Machine Vision and Applications*, vol. 30, no. 7, pp. 1135–1144, 2019.

[41] T. Keras, "Keras applications." `https://keras.io/api/applications`, 2021. [Online; accessed 10-October-2021].

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

[43] H. Zulkifli, "Understanding learning rates and how it improves performance in deep learning," *Towards Data Science*, vol. 21, no. 23, 2018.

[44] J. Mohajon, "Confusion matrix for your multi-class machine learning model," *Towards data science*, 2020.

[45] S. Narkhede, "Understanding auc-roc curve," *Towards Data Science*, vol. 26, no. 1, pp. 220–227, 2018.

[46] J. Wang, L. Lu, Z. Zhang, and N. Slam, "A novel deep convolution neural network model for ct image classification based on covid-19," in *2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)*, pp. 15–20, IEEE, 2022.