COVID-19-related Skin Disease Classification using Convolutional Neural Network

Tatek Gugsa Kassa

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

Sep. 2022
Adama, Ethiopia

COVID-19-related Skin Disease Classification using Convolutional Neural Network

Tatek Gugsa Kassa

Advisor: Tilahun Melak (PhD)

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

Sep. 2022
Adama, Ethiopia

Declaration

I hereby declare that this Master's Thesis entitled **"COVID-19-related Skin Disease Classification using Convolutional Neural Network"** is my original work. That is, it has not been submitted for the award of any academic degree, diploma or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

_____	_____	_____
Name of student	Signature	Date

Recommendation of Advisors

I, the advisor of this thesis, hereby certify that I/we have read the revised version of the thesis entitled **"COVID-19-related Skin Disease Classification using Convolutional Neural Network"** prepared under my guidance by Tatek Gugsa submitted in partial fulfillment of the requirements for the degree of Mater's of Science in Computer Science and Engineering. Therefore, I recommend the submission of revised version of the thesis to the department following the applicable procedures.

_____	_____	_____
Major Advisor	Signature	Date
_____	_____	_____
Co-advisor	Signature	Date

Approval of the Advisors

I, the advisor of the thesis entitled **"COVID-19-related Skin Disease Classification using Convolutional Neural Network"** and developed by Tatek Gugsa, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

_____	_____	_____
Major Advisor	Signature	Date
_____	_____	_____
Co-advisor	Signature	Date

Approval of Board of Examiners

We, the undersigned, members of the Board of Examiners of the thesis by Tatek Gugsa have read and evaluated the thesis entitled **"COVID-19-related Skin Disease Classification using Convolutional Neural Network"** and examined the candidate during open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Computer Science and Engineering.

Chairperson	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

Department Head	Signature	Date
School Dean	Signature	Date
Office of Postgraduate Studies, Dean	Signature	Date

ACKNOWLEDGEMENT

I would like to thank my adviser, Assistant Professor Tilahun Melak (PhD) for his advice and supporting comments starting from the beginning of the research up to the end of the thesis report. I also always admire the computer science and engineering department for the quick responses and providing me all the encouragement and support needed for completing this paperwork successfully. Finally I would also like to thank my friends for those who motivated and supported me with kindness.

Table of Contents

List of Tables

List of Figures

List of Acronyms and Abbreviations

CNN	Convolution Neural Network
Colab	Colaboratory
COVID-19	Coronavirus Disease 2019
FN	False Negative
FP	False Positive
GPU	Graphics Processing Unit
ILSVR	ImageNet Large Scale Visual Recognition challenge
ImageJ	Image Java
Max Pooling	Maximum Pooling
Opencv	Open Source Computer Vision
ReLU	Rectified Linear Unit
RGB	Red, Green, and Blue
ROC Curve	Receiver Operating Characteristics Curve
SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus 2
SoftMax	Soft Maximum
TN	True Negative
TP	True Positive
VGG-16	Visual Geometry Group 16

Abstract

COVID-19 is affecting the skin by the virus in a wide range according to the global reports. The main purpose of this research is to develop a classification model for the four COVID-19-related skin diseases. In this thesis, an ImageJ tool is used to produce segmented images and a CNN method is used to develop the COVID-19-related skin disease classification model. The COVID-19 skin related images are collected from DermNet and Bing images. Initially ImageJ tool is used to segment the lesion region from the nearby healthy skin manually then CNN algorithm is used to classify skin diseases as Livedo-like pattern, Maculopapular rash, Purpuric or Urticaria. Classification results are found with and without segmented images. The proposed classification model result with segmented-image dataset produced training accuracy of 97% and validation accuracy of 99% with epoch 50, and an average testing accuracy of 99%. According to these values the CNN based model with segmented-images yields better result and it also improves classification performance. The research is mainly used for keeping documents of the skin symptoms related to COVID-19 and it is also helpful for the frontline physicians to identify the patients early.

Keywords: *COVID-19-related Skin diseases, Skin lesion, Transfer Learning, Convolutional Neural Networks, VGG16, ImageJ.*

CHAPTER ONE

1. INTRODUCTION

1.1. Background of the study

Skin diseases are recently becoming the COVID-19-related disease and there have been increasing reports of skin manifestations in COVID-19 patients. The disease is affecting the skin by the virus in a wide range according to the recent global reports (A.V. MarzanoiD *et al*., 2020; Giovanni Genovese *et al*., 2020; Giulia Daneshgaran *et al*., 2020). A lot of researches are published in the domain of skin disease classification using deep learning techniques as well as traditional machine learning techniques to solve skin problems. Today, deep learning methods attracted the attention of researchers to diagnose the skin diseases. The main application of deep learning in skin disease recognition is skin disease classification, that is, the quantitative feature extraction of lesion tissues through human skin disease images (Ling-Fang Li *et al*., 2020).

According to the study of COVID-19 and skin relations, maculopapular rash was the main reported skin involvement, 37.3%, commonly occurred in middle-aged females with intermediate severity of the COVID-19 disease. The mild disease was mostly associated with chilblain-like and urticarial-like lesions and patients with the vascular lesions experienced a more severe COVID-19 disease. Seventy-two percentages of patients with the chilblain-like lesions enhanced without any treatment. The overall death rate was 4.5%. The patients with the vascular lesions had the highest death rate of 18.2% and patients with the urticarial-like lesions had the lowest death rate of 2.2% (Giulia Daneshgaran *et al*., 2020).

The Coronavirus disease 2019 (COVID-19) is a respiratory disease that is caused by acute respiratory syndrome coronavirus-2 (SARS-CoV-2). This virus was first recognized in Wuhan, which is a city in the province of Hubei, China, in late 2019 of December (Ren LL *et al*., 2020). On 11th March 2020, the World Health Organization declared the COVID-19 outbreak as a pandemic. Then, it has become a public health emergency of the unprecedented proportions, caused major restrictions on the everyday lives of billions of the population, and strained the healthcare systems worldwide (WHO director, 2020). Italy was the first to experience a dramatic increase in the number of cases, starting in 2020 of February, facing one of the largest clusters of

COVID-19 in Europe (Livingston E *et al*., 2020). The infection caused by the recently identified SARS-CoV-2, known as COVID-19, has still spread worldwide quickly. By the exponential increase of patients throughout the world, the clinical spectrum of COVID-19 is better defined and new symptoms are emerging. Several reports are documenting the occurrence of different skin manifestations in patients by the COVID-19 (A.V. MarzanoiD *et al*., 2020).

COVID-19 Symptom Study application was developed by Zoe Global Limited, supported by physicians and scientists at King's College London and Massachusetts General Hospital, Boston (Drew DA *et al*., 2020). The app collects, on sign up, data on sex, age, ethnicity, and core health risk factors, including height, weight, and common disease (e.g., cancer, diabetes, heart, kidney, and lung disease) status, the use of a set of medications (e.g., corticosteroids, immunosuppressants, and blood pressure medications), and whether the individual is a healthcare worker. Since May 7th, 2020, the app also prompts users to self-report detailed information, also retrospectively, regarding whether they have ever had a SARS-CoV-2 test, and, for each test, how this was performed (e.g., nose/throat swab, antibody testing), and the test result. By using the app, users can provide updates on their daily health status by answering the question "How do you feel right now?" If they feel unwell, the app further collected self-reported presence of 14 COVID-19-related symptoms, namely: abdominal pain, chest pain, delirium, diarrhea, fatigue, fever, headache, hoarse voice, loss of smell, persistent cough, shortness of breath, skipped meals, sore throat, and unusual muscle pains (Marzano AV *et al*., 2020).

Starting from 29th, April 2020, skin manifestations of the disease were added to the application were: raised, red, itchy wheals on the face or body, or sudden swelling of the face or lips (body rash), and red/purple sores or blisters on the feet or toes (acral rash). Asking the participants to differentiate between urticarial rash and erythematopapular/vesicular rash was problematic, therefore the body rashes were collected together, and the second skin question only covered the more specific acral rash (Marzano AV *et al*., 2020). The application also divided the different types of rashes in two broad categories: inflammatory/exanthematous rashes for the various body rashes, and vasculopathic rashes for the fingers or toes (Marzano AV *et al*., 2020).

CNNs are a very powerful model for skin lesion classifier and it is emerged in the field of medicine. Image classification is one of the hottest applications of computer vision. A few years after its development, CNN already won its first image recognition contest 2011, the next year, in 2012; four image recognition contests winners were using Convolutional Neural Network as a base architecture. And even now, CNN is still reigning as the queen in computer vision (Chun Hei *et al*., 2020). To solve the problem CNN is very good at image classification. CNNs are fully connected feed forward neural networks. CNNs are very effective in reducing the number of parameters without losing on the quality of models. Images have high dimensionality (as each pixel is considered as a feature) which suits the above described abilities of CNNs. Also, CNNs were developed keeping images into consideration. CNNs are trained to identify the edges of objects in any image (Prafful Mishra, 2019). Deep Learning is becoming supper interesting area for most researchers. CNN is currently becoming one of the most popular algorithms for deep learning with image data (Chun Hei *et al*., 2020).

On the 13th of March 2020, The Federal Ministry of Health has confirmed a COVID-19 disease case in Addis Ababa, Ethiopia. The case announced on March 13th, 2020, was the first to be reported in Ethiopia since the beginning of the outbreak in China in Dec. 2019. The case was a 48-year old Japanese reported to have moved from Japan to Burkina Faso then he arrived in Ethiopia. He has developed symptoms and presented at the health center in Addis Ababa from where the rapid response team (RRT) moved him to the isolation camp in Kotebe, Yeka. The man was currently clinically stable, with no serious symptoms. The WHO Country Office Ethiopia (WCO) was following up closely to ensure an outbreak in Ethiopia was quickly controlled. The COVID-19 Emergency Operations Centre at Ethiopian Public Health Institute (EPHI) was working closely with the Ministry of Health and the WHO Country Office to answer to this case and implement firm control measures (Alemtsehay Zergaw, 2020).

In Dec. 13, 2020, Ethiopia has conducted 1,706,321 tests fom the beginning of the COVID-19 pandemic, according to the report of the Federal Ministry of Health of Ethiopia. Initially, the country was sending samples to South Africa to get test results. Then slowly established a testing facility at the national reference laboratory and later on expanded to other laboratories. Yet the Ethiopian national testing capacity was very low related to over a 110,000,000-population,

and the facilities concentrate mainly in major cities and university laboratories. One of the critical challenges was inadequate well-trained human power, capital in terms of sample collection and delivery and the test results dispatching throughout the country. The second challenge was lack of infrastructures such as: healthcare infrastructures in particular laboratory facilities with trained personnel, is very limited in sub-Saharan African countries. Laboratory testing has a critical role in isolating the patient and breaks the disease transmission cycle (Andargachew Mulu *et al*., 2020).

In this thesis, the model is developed for COVID-19-related skin disease classification using CNN technique with segmented image dataset. The model can identifies the four types of COVID-19-related skin diseases without the help of Dermatologists. This model also enhanced the performance of the skin disease classification.

1.2. Motivation

The COVID-19-related skin disease has been increasing reports of the skin manifestations in COVID-19 patients. It is affecting the skin by the virus in a wide range according to the global reports (A.V. MarzanoiD *et al*., 2020; Giovanni Genovese *et al*., 2020; Giulia Daneshgaran *et al*., 2020). The manifestations of the coronavirus on the skin as well as the CNN architectures motivated me to select the thesis area and solve the COVID-19-related skin disease classification problem by using segmented images of ImageJ and CNN models.

Image classification is one of the hottest applications of computer vision. Deep Learning is becoming supper interesting area for most researchers throughout the world. Convolutional Neural Network (CNN) is currently becoming one of the most popular algorithms for deep learning with image data (Chun Hei *et al*., 2020). It uses powerful algorithms that help the people to identify the skin diseases without being Dermatologist. CNNs are very effective in reducing the number of parameters without losing on the quality of models. Images have high dimensionality (as each pixel is considered as a feature) which suits the above described abilities of CNNs. Also, CNNs were developed keeping images into consideration. CNNs are trained to identify the edges of objects in any image (Prafful Mishra *et al*., 2020).

The VGG-16 model is a pre-trained model on a huge dataset (ImageNet) with a lot of different image categories. Considering this point, the model should have learned a robust hierarchy of features, which are spatial, rotation, and translation invariant, as we have discussed before with regard to features learned by CNN models. Hence, the model, having learned good representation of features for over a million images belonging to 1,000 classes, can act as a good feature extractor for new images suitable for the computer vision problems. The new images could never exist in the ImageNet dataset or might be different classes, but the model should still be able to extract important features from these images, by considering the principles of the transfer learning technique.

1.3. Statements of the Problem

Numerous reports were documenting the occurrence of different types of skin manifestations in patients with COVID-19. The infection caused by the recently identified SARS-CoV-2, called coronavirus disease-19 (COVID-19), has still rapidly spread throughout the world. With the exponential increase of patients worldwide, the clinical spectrum of COVID-19 is being better defined and new symptoms in skin are emerging (A.V. MarzanoiD *et al*., 2020; Giovanni Genovese *et al*., 2020).

In Ethiopia there is a lack of COVID-19 testing equipment as well as skin disease laboratories. So deep learning plays a crucial role in detecting and classification of the COVID-19-related skin diseases. According to the reports, COVID-19 is affecting the human skin by the virus in a wide range (A.V. MarzanoiD *et al*., 2020; Giovanni Genovese *et al*., 2020; Giulia Daneshgaran *et al*., 2020). The manifestations of the coronavirus on the skin as well as the Convolutional Neural Network architectures motivated me to select the thesis area and solve the COVID-19-related skin disease classification problem by using CNN method and ImageJ tool. Skin symptom plays a crucial role in detecting infections in the people. Keeping documents of the skin symptom related to COVID-19 are important for the frontline physicians to identify the patients early.

To develop COVID-19-related skin identification model we use Convolutional Neural Network (CNN) technique for skin diseases classification using Google Colab (Colaboratory) and Jupyter notebook for development environment with GPU. The VGG16 transfer learning architecture

uses TensorFlow with Keras API to build the model. Keras is compact, easy to learn, high-level Python library run on top of TensorFlow framework. It is made with focus of understanding deep learning techniques, such as creating layers for neural networks maintaining the concepts of shapes and mathematical details.

1.4. Research Questions

These questions were formulated from the research problem to be answered:

a) How can we segment COVID-19-related skin lesions using ImageJ?
b) Which CNN method is used for classification of the skin diseases?
c) Can we enhance the accuracy of the classification model?
d) How can measure the performance of the overall classification model?

1.5. Objectives

1.5.1. General objective:

- The general objective of this research is to build a COVID-19-related skin diseases classification model using CNN technique.

1.5.2. Specific objectives:

- To study different types of CNN methods and related works.
- To use an ImageJ tool for the COVID-19-related skin images segmentation process
- To build a CNN classification model for the four COVID-19-related skin diseases using segmented-image dataset.
- To evaluate performance of the model by using confusion metrics.

1.6. Significance of the Study

- Health care institutes will be beneficiaries from this research for their COVID-19 case identification and assessments.
- Students can use this thesis for further study in COVID-19-related skin diseases using deep learning.
- Dermatologists will be benefited for skin disease diagnosis to identify the disease easily without using laboratories.
- The study shows how segmented images increase the classification accuracy.

1.7. Scopes and Limitations of the Study

1.7.1. Scopes

In this research the model has been trained for the most frequently reported dermatological symptoms of COVID-19-related skin dataset. The skin rashes included in this thesis are: Livedo reticularis-like pattern, Maculopapular rash, Purpuric 'vasculitic' pattern, and Urticaria. We compared and contrasted the proposed model performance with transfer learning and fine-tuned of VGG-16 model using segmented dataset.

1.7.2. Limitations

The study we did not include rare case of COVID-19-related skin rashes in the classification model. To incorporate more other kinds of the diseases, dataset gathering is also a challenging activity, because COVID-19-related skin is currently emerging symptom. Since the study used a deep learning architecture, it requires more computing power to train the model. So the experiment is conducted with online Google Colab Jupyter Notebook with virtual GPU.

1.8. Organization of the Thesis

Chapter one, *Introduction*, includes the background information of the study, statement of the problem, research questions, the objectives of the study, significance of the study, and objectives of the study. Chapter two, *Literature Review*, deals with the existing literatures of related works with the objective of contributions, weaknesses and gaps. Chapter three, *Materials and Methods*, describes the research methodology, methods, tools and procedures to attain the objectives of the study. Chapter four, *design and Implementation*, discusses the how the proposed methodology is designed and implemented as well as it described the hardware used and the platform used for the experiment. Chapter five, *Results and Discussions*, discusses the original work of the research besides analysis and results are reported from the experimented result of the study. In chapter six, *Conclusions and Recommendations*, findings and conclusions of the study are summarized and recommendations for future works are also discussed.

CHAPTER TWO

2. LITERATURE REVIEW

2.1. Introduction

Skin is the largest organ of the human body, it consists epidermis, dermis and hypodermis. Our skin is a good indicator of our health conditions. If a person is sick, it often shows in their skin. The three main functions of the skin are: auspice, sensation and thermoregulation, providing an excellent protection against aggression of the environment. Stratum corneum is the top layer of the epidermis and optically neutral protective layer with varying thickness of the skin. The stratum corneum consists of keratinocytes that generate keratin responsible for benefiting the skin to protect the body. The incident of light on our skin is scattered due to the stratum corneum. The epidermis includes melanocytes in its basal layer. Mostly, melanocytes make the skin to generate pigment known as melanin, which provides tan or brown color of the skin. Melanocytes act as a filter and protect the skin from harmful Ultraviolet rays by generating more melanin. The extent of absorption of Ultraviolet rays depends on the concentration of melanocytes. But, the unusual growth of melanocytes causes melanoma. The dermis is located at the middle layer of the skin, it consists: collagen fibers, sensors, receptors, blood vessels and nerve ends. It also provides elasticity and vigor to the skin of our body.

The Coronavirus disease 2019 (COVID-19) is a respiratory illness caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). It was first identified in Wuhan, a city in the province of Hubei, China, in late December 2019 (Ren LL *et al*., 2020). On March 11, 2020, the World Health Organization (WHO) declared the SARS-CoV-2 outbreak a pandemic. Since then, it has become a public health emergency of unprecedented proportions, caused major restrictions on the everyday lives of billions of people, and strained healthcare systems worldwide. Italy was the first European country to experience a dramatic increase in the number of cases, starting in February 2020, facing one of the largest clusters of COVID-19 in Europe (Livingston E *et al*., 2019). The infection caused by the recently identified SARS-CoV-2, called coronavirus disease-19 (COVID-19), has still rapidly spread throughout the world. With the exponential increase of patients worldwide, the clinical spectrum of COVID-19 is being better defined and new

symptoms are emerging. Numerous reports are documenting the occurrence of different cutaneous manifestations in patients with COVID-19 (A.V. MarzanoiD *et al*., 2020).

The COVID Symptom Study application has been described in detail previously (Drew DA *et al*., 2020). The application collects, on sign up, data on sex, age, ethnicity, and core health risk factors, including height, weight, and common disease status, the use of a set of medications (e.g., corticosteroids, immunosuppressants, and blood pressure medications), and whether the individual is a healthcare worker. Since May 7th, 2020, the app also prompts users to self-report detailed information, also retrospectively, regarding whether they have ever had a SARS-CoV-2 test, and, for each test, how this was performed (e.g., nose/throat swab, antibody testing), and the test result. By using the app, users can provide updates on their daily health status by answering the question "How do you feel right now?" If they feel unwell, the app further collected self-reported presence of 14 COVID-19-related symptoms, namely: abdominal pain, chest pain, delirium, diarrhea, fatigue, fever, headache, hoarse voice, loss of smell, persistent cough, shortness of breath, skipped meals, sore throat, and unusual muscle pains.

From April 29th, 2020, skin manifestations of the disease were added: raised, red, itchy wheals on the face or body or sudden swelling of the face or lips (body rash), and red/purple sores or blisters on the feet or toes (acral rash). Asking the participants to differentiate between a transient urticarial rash and a fixed erythematopapular/vesicular rash was problematic, so the body rashes were collected together, and the second skin question only covered the more specific acral rash. It also divided the many different types of rashes in two broad categories: inflammatory/exanthematous rashes for the various body rashes, and vasculopathic rashes for the fingers or toes (Marzano AV *et al*., 2020).

A lot of researches are published in the domain of skin disease classification using deep learning techniques as well as traditional machine learning techniques to solve skin problems. Today, deep learning methods attracted the attention of researchers to diagnose the diseases. The main application of deep learning in skin disease recognition is skin disease classification, that is, the quantitative feature extraction of lesion tissues through skin disease images (Ling-Fang Li *et al*., 2020).

Deep learning is a specific subfield of machine learning. The deep in deep learning isn't a reference to any kind of deeper understanding achieved by the approach; rather, it stands for this idea of successive layers of representations. How many layers contribute to a model of the data is called the depth of the model. Modern deep learning often involves tens or even hundreds of successive layers of representations and they're all learned automatically from exposure to training data (François Chollet, 2018). A few years after its development, CNN already won its first image recognition contest (2011), the next year, in 2012, four image recognition contests winners were using CNN as a base architecture. And even now, CNN is still reigning as the queen in computer vision (Chun Hei *et al*., 2020).

2.2. Classification of COVID-19-associated skin rashes:

Urticarial rash/hive-type rash; confluent erythematous maculopapular/morbilliform rash; papulovesicular exanthema/ckickenpox-type rash; chilblain-like acral/pseudo-chiblain/COVID-fingers and toes/COVID digits/perio-like pattern; livedo reticularis/livedo racemosa-like pattern; and purpuric 'vasculitic' pattern are the most commonly reported cutaneous symptoms of COVID-19 (Alin Laurentiu Tatu *et al*., 2020).

i. **Urticarial rashes (hives)** are raised bumps on the skin which come and do quite quickly and usually very itchy. Can involve any part of the body. Often starts with intense itching of the palms or soles. These rashes can present quiet early on the infection, but can also last a long time afterwards.

ii. **Maculopapular rashes** are made of both flat and raised skin lesions. The name is a blend of the words "macule", which are flat discolored skin lesions, and "papule", which are small raised bumps. These skin lesions are usually red and can merge together. Maculopapular rashes are also associated with coronavirus.

iii. **Papulovesicular exanthema** is small and itchy red bumps that can occur anywhere on the body, particularly the elbows and knees, back of the hands and feet. The rash can persist for days or weeks. papulovesicular exanthem tends to involve more frequently the adult population, with a median age of 60 years.

iv. **COVID fingers and toes (Chilblains)** are reddish and purplish bumps on the fingers and toes and they may be sore but not usually itchy. This type of rash is the most specific to COVID-19 and more common in younger people. Chilblains also known as pernio and chill burns, are a medical condition in which damage occurs to capillary beds in the skin, most often in the hands or feet, when blood perfuse into the nearby tissue resulting in redness, itching, inflammation, and possibly blisters.

v. **Livedo reticularis pattern** is a particular type of skin discoloration that appeared as mild, transient, symmetrical, lace-like, dusky patches forming complete ring surrounding a pale center. It is seen primarily on the legs and occasionally on the arms. It vaguely resembles a pair of blue or purple fishnet stockings. It may get worse when the temperature is cold. Livedo has many causes, it may due to spasms of the blood vessels or an abnormality of the local circulation. It is also becoming a late manifestation of COVID-19.

vi. **Purpuric 'vasculitic' pattern** is occurred when small blood vessels burst, causing blood to pool just under the skin. They appear as small purple spots just beneath the skin's surface. This skin rash is emerging in COVID-19 related sign. The main symptom of the purpura is a purplish-red rash just beneath the skin's surface. The rash can appear anywhere on the body, including on mucous membranes such as the lining of the mouth.

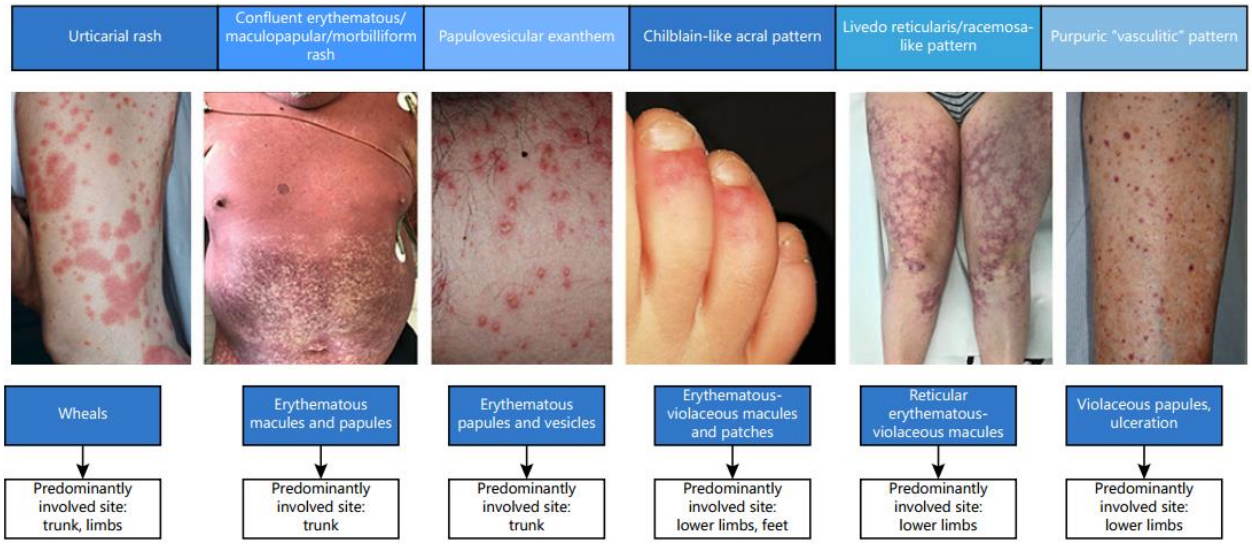


Figure 2. 1: Clinical features of COVID-19-related skin manifestations (Giovanni *et al*., 2020)

2.3. Convolutional Neural Network (CNN)

CNNs are composed of convolutional layer, pooling layer, ReLU layer, fully connected layer, and loss layer. Convolutional layer is the core block of a CNN consisting of filters (or kernels) to detect different types of features from the input and pass them forward. Pooling layer is in-between successive convolutional layers to reduce the parameters and computers in the network. In this layer we shrink the image stack into a smaller size**.** Pooling is done after passing through the activation layer. Rectified Linear Unit **(**ReLU) layer is the activation function that sets negative values to zero. ReLU transform function only activates a node if the input is above a certain quantity, while the input is below zero, the output is zero, but when the input rises above a certain threshold, it has a linear relationship with the dependent variable. Fully connected layer has full connections to all activations in previous layer to flatten the output that represents high-level features in the data (A. Rao, 2019).

A Convolution Neural Network or CNN is a class of deep learning neural networks models that is most commonly applied to analyzing visual imagery. It's effectiveness comes from the fact that CNNs are fully connected feed forward neural networks that reduce the number of parameters very efficiently without losing out on the quality of models. Images have high dimensionality (as each pixel is considered as a feature) which suits the above described abilities of CNNs. The model usually keeps learning new higher dimensionality features with every layer.

i. Convolution Layer

The term “convolution” refers to the mathematical combination of two functions to form a third function. When that happens, two sets of information are merged. In the context of CNNs, a convolutional layer (called filter or kernel) is applied to the input data to then produce a feature map.

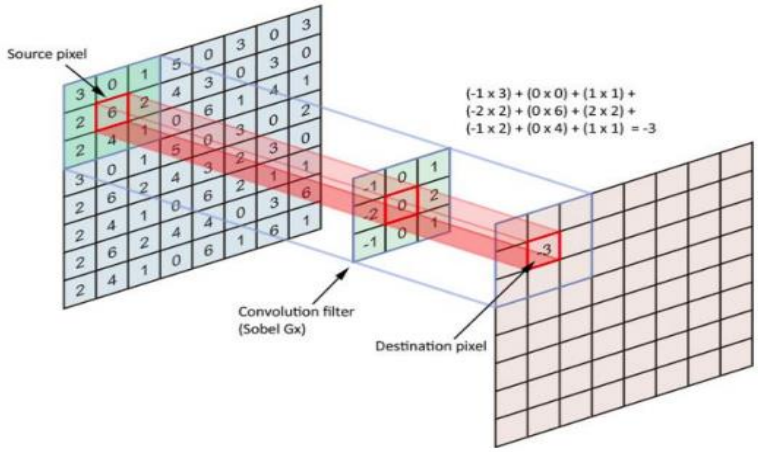


Figure 2. 2: Filter slides over input and performs its output on new layer (Cornelisse, D., 2019)

In the figure, a dot product multiplication is made between a 3x3 sized filter matrix and a 3x3 sized area of the input image's matrix. The elements of the resulting matrix are summed and the sum is the output value ("destination 16 pixel") on the feature map. The filter then slides over the input matrix, repeats the dot product multiplication with every remaining combination of 3x3 sized areas and completes the feature map. Multiple filters are used for one input and the resulting feature maps are joined together for the final output of one convolutional layer (Cornelisse *et al.*, 2019).

There are two other important concepts in convolutional layers: strides and padding. Strides are the number of pixel a kernel or a filter slides over the input matrix. Padding is what is used when the filter does not fit the input matrix. There are two types of padding: valid padding, when the border pixels of the input matrix are discarded; and zero or "same" padding, when zeros are added to the borders so that the filter fits the input matrix (Prabhu, 2018).

ii. Rectified linear unit activation functions (ReLU)

Activation function is also added on each Convolutional layer. The rectified linear unit activation function (ReLU) is a piecewise linear function that will output the input if is positive, otherwise it will output zero. ReLu layer is the activation function that sets negative values to zero. Fully connected layer has full connections to all activations in previous layer to flatten the output that

represents high-level features in the data. Loss layer is used to show the deviation between predicted (output) and true labels (Dertat *et al*., 2017).

iii. Pooling Layer

Pooling layers are responsible for reducing the dimensionality of feature maps, specifically the height and width, preserving the depth (Dertat *et al*., 2017). Doing so is beneficial because it decreases the required computational power to process the data, while extracting the dominant features in feature maps. Max pooling outputs the maximum value of the elements in the portion of the image covered by the filter, while average pooling returns the average value. Max pooling is better at extracting dominant features (Saha *et al*., 2018).

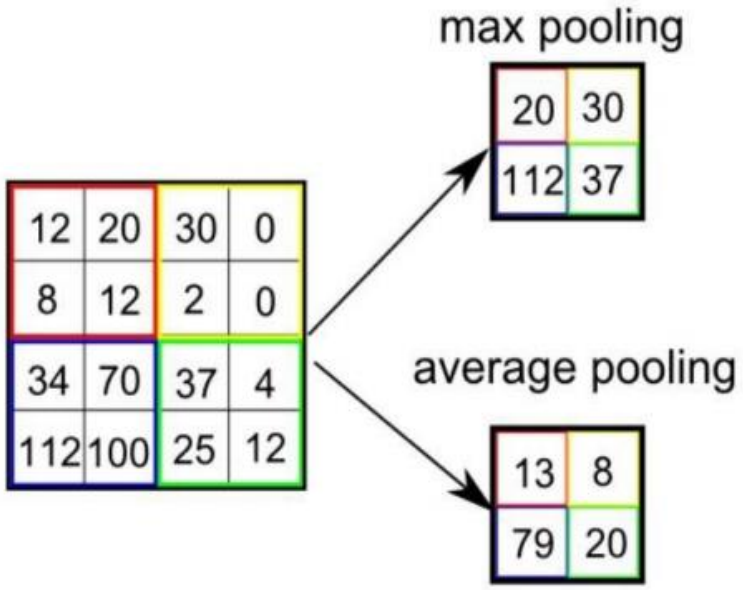


Figure 2. 3: Two types of pooling layers: max pooling and average (Saha *et al*., 2018)

iv. Fully Connected Layer

Fully connected layers are where classification actually happens. The input matrix is flattened into a column vector and is fed into a set of fully connected layers (Saha *et al*., 2018). Each fully connected layer (called Dense layer) is passed through an activation function (e.g. tanh or ReLu), but the output Dense layer is passed through Softmax. In the Softmax multiclass classification, the loss function used is CrossEntropy (categorical_crossentropy in Keras).

The output of the Softmax function is an N-dimensional vector, where N is the number of classes the CNN has to choose from. Each number in this N-dimensional vector represents the probability that the image belongs to each certain class. For example, if the output vector is (0 .1 .1 .75 0 0 0 0 0 .05), then there is a 10% probability that this image belongs to the class 2, 10% probability that it belongs to the class 3, 75% probability that this image belongs to the class 4, and 5% probability that it belongs to the class 10 (Deshpande *et al*., 2016).

Normalization

Normalization is a pre-processing stage where we can find new range from an existing one. It is helpful in prediction a lot. To maintain the large variation of the prediction, normalization technique is required to make them closer by structuring the given dataset in well-organized manner. As the next pre-processing step, our proposed model's input data is normalized. According to (Andrew *et al*., 2016), normalization of input data can speed up the training process. Normalization is done on the image by applying the linear scaling to squeeze all pixels of the binarized image to values between 0 and 1. Dividing each input image by its standard deviation squeezes the pixel values from pixel values between 0 and 255 to between 0 and 1 (Daniel *et al*., 2018). Generally, applying normalization on an image makes optimization a little bit easier, it makes all features to be in the same range and contribute equally, and it also avoids some contrast issues (D. S. Gupta, 2017).

a) Dropout

The dropout layer is useful to avoid the overfitting of the data during the training process and it is applied to the fully connected layers at the end of a neural network. In practise it drops out randomly an amount of neuron units proportional to its setting parameter. For example a dropout of 0.5 connects only 50% of the units. Applying this during training makes that different units are trained randomly for each iteration. This can prevent to overfit the data. At inference, all units are present in the network, but the output weights are scaled by the probability of its presents during training, so in the example $0.5*w_i$ for other details take reference to the paper (H. C. Shin *et al*., 2016).

b) Softmax, Loss, and regularization

The Softmax, also called multinomial logistic regression, is a math function that normalizes the predicted scores of the classes. The scores are the output values of a convolutional neural network. They are placed in a vector form as outputs of the classifier, where each position corresponds to a class, and so in an inference test the maximum value of the score vector should correspond to the correct output class. The softmax function is used after a classifier, as computing the probabilities of each class. The Softmax take the scores and squashes it to a vector of values between zero and one. So the outputs of a softmax can be interpret as a vector of probabilities of a label in a classification task. During the training, the goal is to maximize the probability of the correct class. So, what is done is to minimize the negative log likelihood of the correct class (H. C. Shin *et al*., 2016).

The Loss function is also called cost function or objective, and quantifies the goodness of the prediction, based on the scores, along the training data. The lower is the value of the loss, the better is the classification of the training data. At the initialization step all weights are near zero, so also all the class scores, and the loss should be of value –log(1/Nclasses). This is useful as sanity check at the beginning of the training process, to see for example if there's something wrong in the developed code, or if the loss is implemented in a good way (H. C. Shin *et al*., 2016).

The role of regularization:

Unfortunately, the solution in finding optimal parameters can be not unique, so it is important to add a regularization in the loss formula. This regularization term, permits to generalize better the data and have less test errors. This occurs because regularization fights against the data loss, penalizing unbalanced weights, so large weights and zero weights in a layer, and increase the generalization property spreading out better the weights values around zero, avoiding that a particular input influence most the output than others. So, in practise, the regularization takes care about having better weights, whereas the data loss looks only to the class score predictions (Nitish *et al*., 2014).

2.3.1. VGG16

VGG16 is a CNN architecture that was the first runner-up in the 2014 ImageNet Challenge. It's designed by the Visual Graphics Group at Oxford and has 16 layers in total, with 13 convolutional layers themselves. We will load the pre-trained weights of this model so that we can utilize the useful features this model has learned for our task (Nikhil B., 2017).

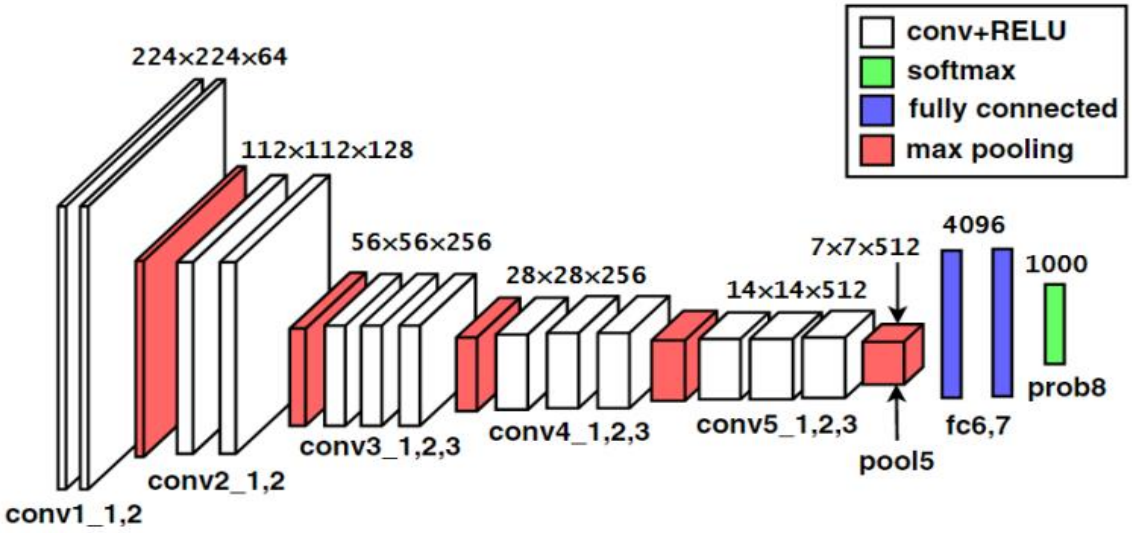


Figure 2. 4: VGG16 model architecture (Nikhil B., 2017)

- **Employment of VGG16 model**

Due to time restrictions or computational cost, it's not always possible to build a model from scratch which is why pre-trained models exist, I will employ one of the pre-trained model VGG16 to classify image and check accuracy for training data and validation data. The VGG16 model has 13 convolutional layers, 2 fully connected layers, and 1 SoftMax classifier. The precise structure of the VGG16 network is described as follows:

a) The first and second convolutional layers are comprised of 64 feature kernel filters and size of the filter is 3×3. As input image (RGB image with depth 3) passed into first and second convolutional layer, dimensions changes to 224x224x64. Then the resulting output is passed to max pooling layer with a stride of 2.

b) The third and fourth convolutional layers are of 124 feature kernel filters and size of filter is 3×3. These two layers are followed by a max pooling layer with stride 2 and the resulting output will be reduced to 56x56x128.
c) The fifth, sixth and seventh layers are convolutional layers with kernel size 3×3. All three use 256 feature maps. These layers are followed by a max pooling layer with stride 2.
d) Eighth to thirteen are two sets of convolutional layers with kernel size 3×3. All these sets of convolutional layers have 512 kernel filters. These layers are followed by max pooling layer with stride of 1.
e) Fourteen and fifteen layers are fully connected hidden layers of 4096 units followed by a softmax output layer (Sixteenth layer) of 1000 units.

- **Transfer Learning**

In context of deep learning, most models need lots of data, and obtaining a large amount of labeled data for a variety of domains can be difficult, considering the time and effort it takes to label the data. Although there is ImageNet dataset with millions of data, getting data for every domain is tough. While there are state-of-the-art models which give high accuracy, they would be only on specific datasets and when used on new similar task, they suffer a significant loss. This forms the basis for transfer learning which used knowledge gained from pre-trained networks and use it to solve new similar problems (S.Gopal *et al*., 2015). Despite deep learning models achieved excellent performance on various experimental datasets, one should also consider the fact that most deep learning models require a whole lot of data for training, and obtaining vast amounts of labeled data (especially medical data) can be really difficult and expensive in terms of both time and money.

Fortunately, transfer learning can be a strategy to alleviate this issue, enabling deep learning models to achieve satisfying performance on small datasets. The basic concept of transfer learning is to train a model on a large dataset and transfer its knowledge to a smaller one. Thus, one can utilize a deep network trained on unrelated categories in a massive dataset (usually ImageNet) and apply it to our own problems (e.g., skin disease classification). As only limited skin disease data can be obtained publicly, transfer learning is widely adopt in skin disease classification tasks. (Haofu *et al*., 2016) used the pretrained VGG-16, VGG-19 and GoogLeNet networks to construct a universal skin disease diagnosis system.

2.4. Leveraging Transfer Learning with Pre-trained CNN Models

Considering the transfer learning with the pre-trained VGG-16 model, we can perform the classification using three approaches as follows (Ulzii-Orshikh *et al*., 2018).

2.4.1. Feature Extractor

In this method, just use it as a simple feature extractor by freezing all the five convolution blocks to make sure their weights do not get updated after each epoch.

2.4.2. Feature Extractor with Image Augmentation

To overcome small training data problem in the first approach, we use the advantage of Image Augmentation here. It is the process in which we apply image transformation operations such as rotation, shearing, translation, zooming, and so on, to produce new, altered versions of existing images. Due to these random transformations, we do not get the same images each time and these results in more training image datasets. With more training data we perform the feature extraction and image classification (Q. Guan *et al*., 2019).

2.4.3. Fine-Tuning with Image Augmentation

For the last model, we will apply fine-tuning to the model, where we will unfreeze the last two blocks (Block 4 and Block 5 in case of VGG-16 shown in Figure 30) so that their weights get updated in each epoch. This is a more involved technique as we selectively retrain some of the previous layers instead of just replacing the final layer (Keras *et al*., 2020).

Thus, we are mostly concerned with leveraging the convolution blocks of the pre-trained model and then flattening the output (from the feature maps) so that we can feed it into our own dense layers for our classifier. Figure 30 describes a better block diagram visual perspective (D. Sarkar, 2018). Pre-trained models are used in the following two popular ways when building new models or reusing them:

- Using a pre-trained model as a feature extractor
- Fine-tuning the pre-trained model

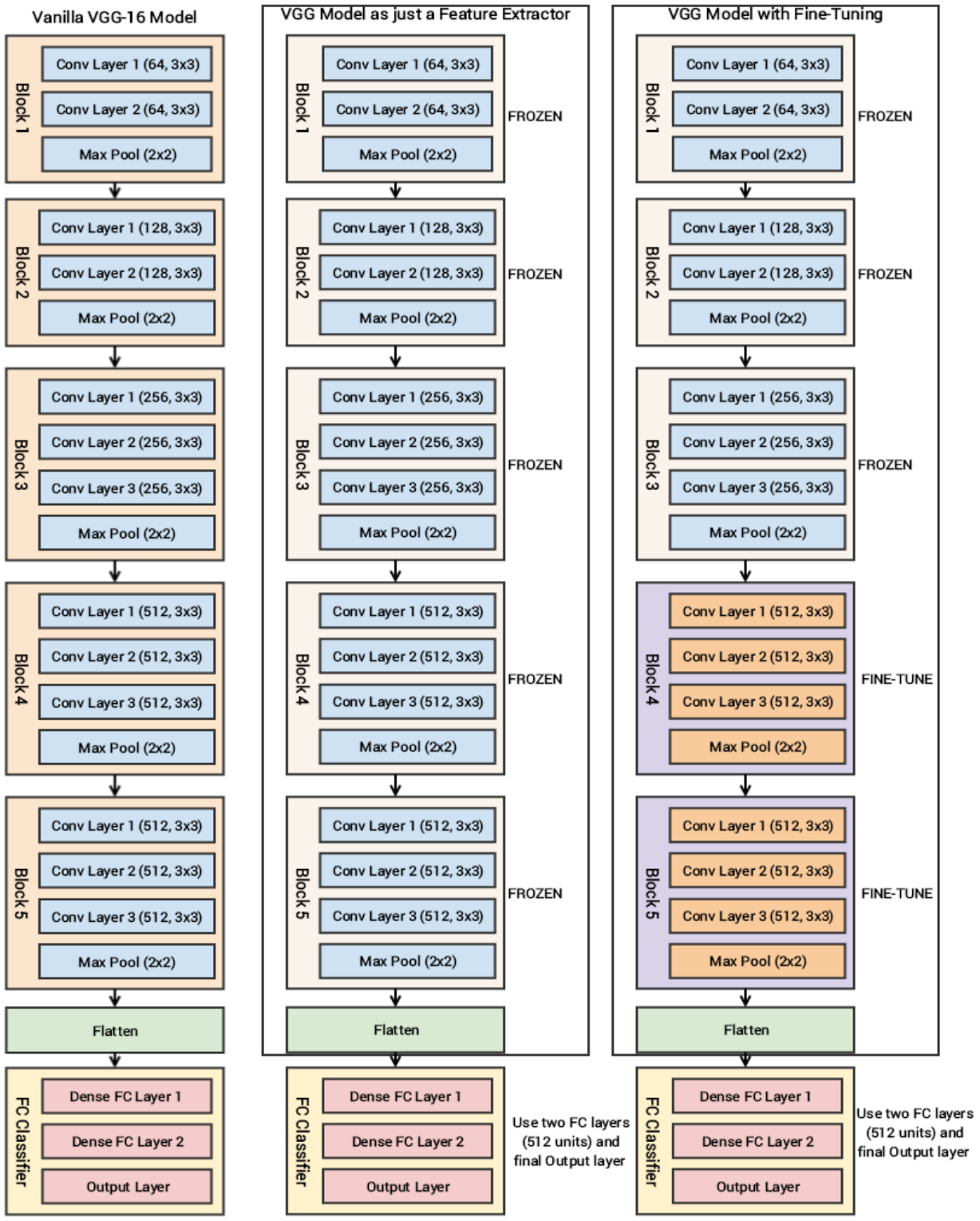


Figure 2. 5**:** Block Diagram of Transfer Learning Strategies on VGG16 Model (D. Sarkar, 2018)

We compared and contrasted the proposed model performance with transfer learning and fine-tuned of VGG-16 model using segmented dataset.

2.5. ImageJ

ImageJ is a powerful image analysis program that was created at the National Institutes of Health, Wisconsin University, USA. It is in the public domain, runs on a variety of operating systems and is updated frequently. ImageJ is powerful software for the segmentation of image. However, and this is true for any software based on thresholding that we can use, the result depends on the threshold that we decided to have the binary image. Then, the final results, such as the number of particles and other related measurements, are linked to the value of the threshold (A. C. Sparavigna, 2020).

In the image processing segmentation is a method of partitioning an image into multiple segments, which consist of sets of pixels. These segments are also known as image objects or super-pixels. Many can be the goals of segmentation; basically they consist in extracting features by the image representation into the set of segments. Among the image segmentation methods, we have also those which are devoted to the determination of the contours of objects; in this case, the segmentation is known as edge detection. Many methods of segmentation are based on the thresholding of the image. That is, the image -made by grey pixels for instance - is converted into a black and white image, by choosing a specific value of grey (the threshold). The pixels having a grey tone with a value larger than that of the threshold are converted into white tones, whereas the pixels having a grey tone lower or equal to the threshold are converted into black pixels. Once the image is converted in this manner, the black or the white domains, that is the super-pixels, can be identified, that is labeled, by means of an integer or a color (A. C. Sparavigna, 2020).

2.5.1. Segmentation

Segmentation aims to divide an image into distinct regions that contain pixels with similar attributes. Segmentation is significant for skin disease diagnosis since it avails clinicians to perceive the boundaries of lesions.

We can use segmentation to separate a single image into separate components based on shape, size or intensity to make image analysis of each of those components possible. For this we use “masks”. Masks are just a binary image being used for “Image calculation” (e.g. digital

subtraction) to extract or remove specific structures. In this example we'll use a particle analysis to make the masks and remove the chloroplasts from an image of a plant section.

Image>Adjust>Threshold (Threshold using the "Minimum" (top) slider to find the range of image intensities) and click "Apply". Then "Process>Binary>Make binary"

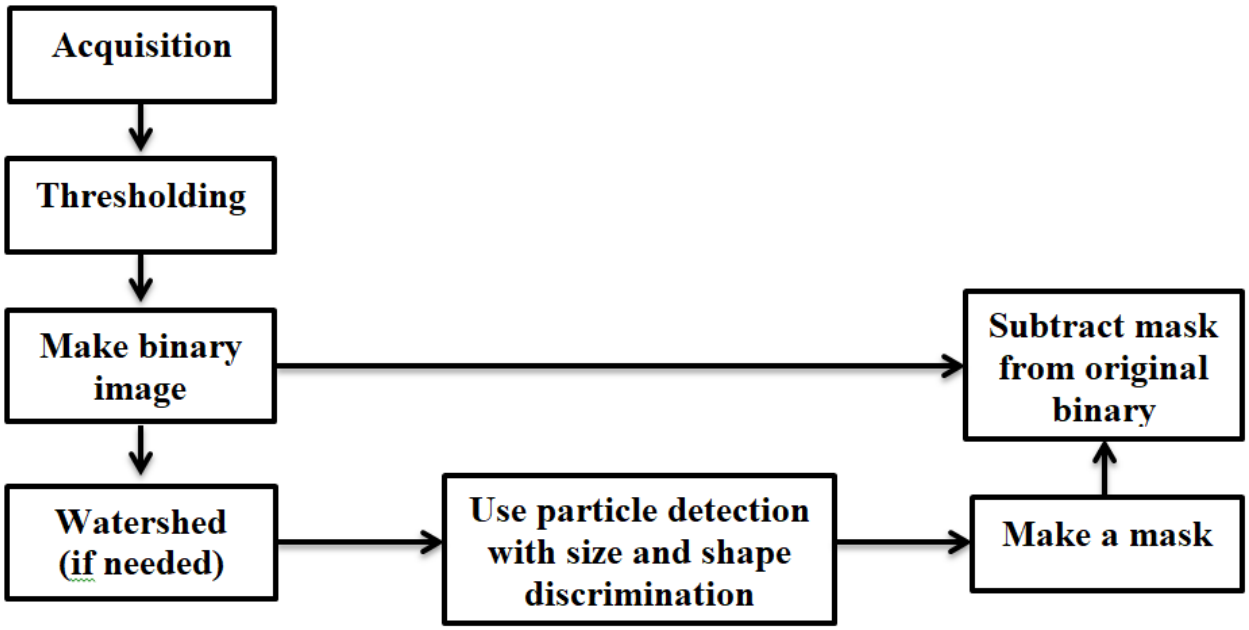


Figure 2. 6: ImageJ segmentation process (A. C. Sparavigna, 2020)

After the image is uploaded, the first processing that we have to make is its thresholding. The suggested approach is that of change the color image into an image having pixels in grey tones, and uploads the grey-tone image, such as that in the middle of the Figure 2.7. The grey-tone image has been obtained by means of GIMP, the GNU Image Manipulation Program. Then, let us make the thresholding of this image, by means of the ImageJ instruction sequence Image > Adjust > Threshold. The result is given on the right of the Figure 2.7, for the default value of threshold. Let us note that we can make the thresholding using GIMP too. Moreover, a preprocessing of the image is often required, before any segmentation process we can imagine. The micrograph in the Figure 2.7 on the left is a special case, because it is not requiring any specific preprocessing for being segmented (A. C. Sparavigna, 2020).

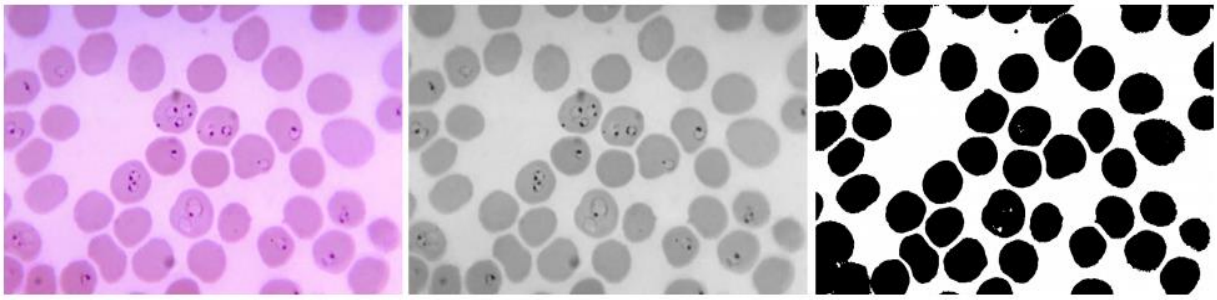


Figure 2. 7: ImageJ segmentation example (A. C. Sparavigna, 2020)

After the black and white has been obtained, we have to make it binary for ImageJ. We use the following sequence of instructions: Process > Binary > Make Binary. That is, we make it truly binary for ImageJ. We have the image that software can analyse. The segmentation is obtained by means of the following sequence of instructions: Analyze > Analyze Particles. The result is given in the Figure 2.7. The domains on edges can be excluded as in the given image.

2.6. Related Work

A lot of researches are published in the domain of skin disease classification using deep learning techniques. In these works, there are a lot of different approaches are implemented including classification only, segmentation and detection, image processing using different types of filters.

.

A model is developed for diagnosis of melanoma using deep convolutional neural network. Automated classification of skin lesions using images is a challenging task owing to the fine-grained variability in the appearance of skin lesions. Here demonstrate classification of skin lesions using a single CNN, trained end-to-end from images directly, using only pixels and disease labels as inputs. The first case represents the identification of the most common cancers; the second represents the identification of the deadliest skin cancer. It is projected that 6.3 billion smartphone subscriptions will exist by the year 2021 and can therefore potentially provide low-cost universal access to vital diagnostic care. Deep convolutional neural networks (CNNs) show potential for general and highly variable tasks across many fine-grained object categories (Esteva et al., 2017).

It is used a deep convolutional neural network to classify the clinical images of 12 skin diseases in Ensemble deep of residual networks (DRN), Convolutional Neural Network, and fully

convolutional U-Net architecture. There are two limitations of the study. The first involves a lack of statistical significance of comparisons. Performance evaluations were carried out on a fixed dataset partition, which is necessary for a public challenge to maintain a held-out blind test dataset. In addition, software implementations of approaches used for comparison were not available to support multiple nfold evaluations. The second limitation is that the diagnostic performance of clinical experts with dermoscopic images represents an approximation of performance in practice, where clinicians may benefit from the situational context of each lesion, including patient history, temporal evolution, comparison of the lesion to other lesions on the patient, and physical inspection (Codella *et al*., 2017).

A deep learning approach is implemented to universal skin disease classification by a pre-trained VGG16, VGG19 and GoogleNet models to classify clinical skin cancer images including dermatoscopic images. ImageNet data are not specialized for skin data, the ImageNet pretrained models may not be the best choice for skin disease classification (Liao *et al*., 2016).

In (Dorj *et al*., 2018), a model is developed using convolutional neural network with over 50 layers on ISBI 2016 challenge dataset for the classification of malignant melanoma. The acquired images are extracted by deep convolutional neural network, and obtained convolutional neural network features are classified into four groups, i.e., Actinic Keratoses, Basal cell carcinoma, Squamous cell carcinoma, and Melanoma, using ECOC SVM with pre-trained AlexNet convolutional neural network model is used in the extract training features. The main goal of this study is to classify the skin cancer images, which are extracted by the deep convolutional neural network. A total of four different kinds of skin cancers are classified using ECOC SVM. The results are obtained with a total of 3753 images, which are collected from the internet (google.com, naver.com, baidu.com, and bing.com).

In (Rezvantalab *et al*., 2018) this paper, the effectiveness and capability of convolutional neural networks have been studied in the classification of 8 skin diseases. Different pre-trained state-of-the-art architectures (DenseNet 201, ResNet 152, Inception v3, InceptionResNet v2) were used.

Table 2. 1: Summary of Related Work

No.	Title	Methodology used	Gaps
1	Dermatologist-level classification of skin cancer with deep neural networks	Deep Convolutional Neural Network (Inception-v3)	Uses a single pre-trained model with only transfer learning deep neural networks.
2	Deep learning ensembles for melanoma recognition in dermoscopy images	Ensemble deep of residual networks (DRN), CNN, and fully convolutional U-Net architecture	Performance evaluations were carried out on a fixed dataset.
3	A deep learning approach to universal skin disease classification	VGG16, VGG19 and GoogleNet models	ImageNet data are not specialized for skin data, the ImageNet pretrained models may not be the best choice for skin disease classification.
4	The skin cancer classification using deep convolutional neural networks.	Combined SVM with AlexNet	Uses the pretrained model as feature extractor, not finetuned some block of layers. The traditional SVM classifier is utilized in the classification skin cancers.
5	Dermatologist level dermoscopy skin cancer classification using different deep learning convolutional neural networks algorithms	Deep convolutional neural network approach (DenseNet 201, ResNet 152, Inception v3, InceptionResNet v2)	Different approaches were used to classify melanoma, but these approaches rely on comparison. No ensemble model on the experiment.

CHAPTER THREE

3. RESEARCH METHODOLOGY

3.1. General Approach

In this chapter we Review literatures that are related to skin disease classification that are used deep learning techniques to formulate the research questions. The GoogleScholar, PubMed, Science Direct, and Web of Science databases were searched for systematic reviews on deep learning skin disease classification. The search terms are: convolutional neural networks, deep learning, skin disease, skin lesions, and covid-19 skins. Only the papers that are related to the research are selected in this review.

COVID-19-related skin disease images are collected from dermNet and Bing images and the dataset is prepared by means of preprocessing and segmentation of the images using ImageJ tool. The proposed CNN model is developed to detect and classify the disease then the model is evaluated using the segmented-image dataset with the hyper-parameters including: learning rate, batch size, number of epochs, optimizers and loss function. The collected images have different sizes with in the ranges between 150 and 1500 pixels. In Convolutional Neural Network all the images should have the same size. So image resize is the basic step in image preprocess because of the data set must have similar size for the proposed work and the large pixel's size consumes too much computational cost and time. Preprocessing is required to clean image data for the model input. The all images are cropped to make convenient for segmentation processes. Since In convolutional neural networks required that all images are the same sized arrays, we resize the image to size of 224x224 pixels.

3.2. Research Design

The research design is answering our research questions using the collected dataset. We used experimental research design to achieve the goal of the study, since research design used in deep learning is mainly quantitative. Therefore, this method involved the design of the experiment and conducting the experiment to obtain the COVID-19-related skin classification results. The approach used in conducting the experiment is as follows: data preprocessing and dataset preparation, splitting the dataset into training set and testing set, data augmentation, model building, model training, model testing and evaluation, finally we get the disease classification model. The steps are summarized in Figure 3.1.

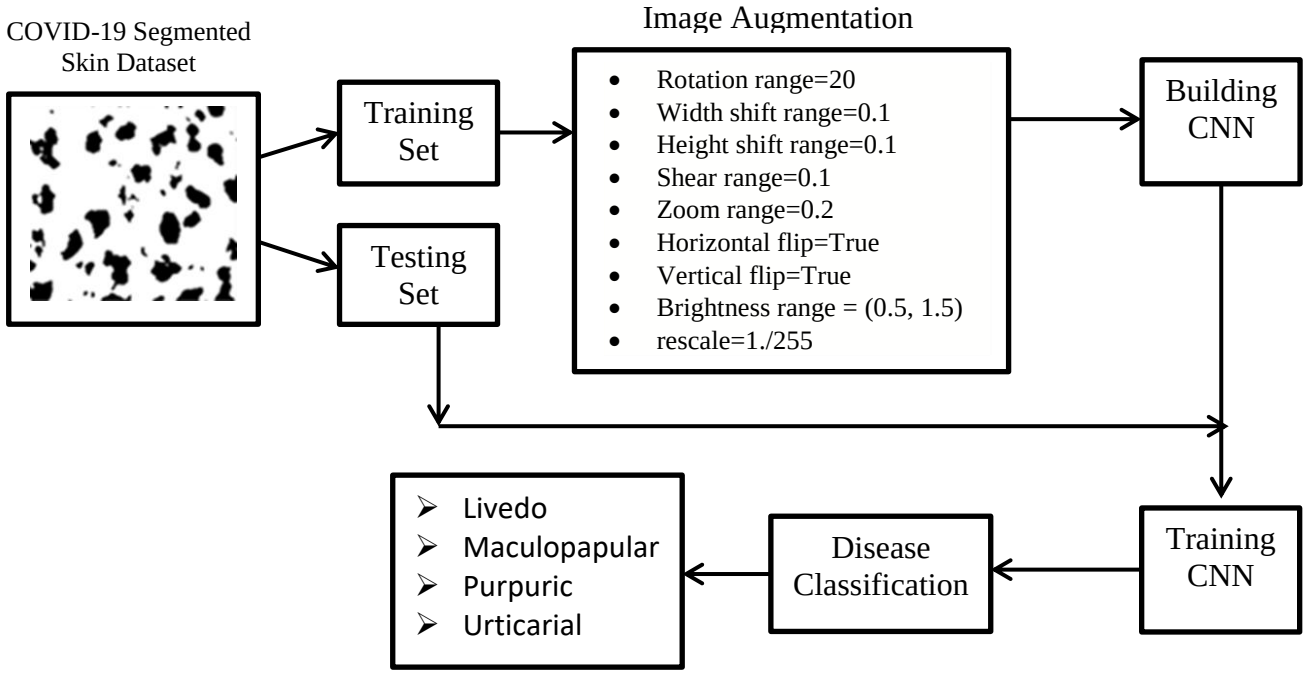


Figure 3. 1: Block diagram of proposed COVID-19-related skin disease classification

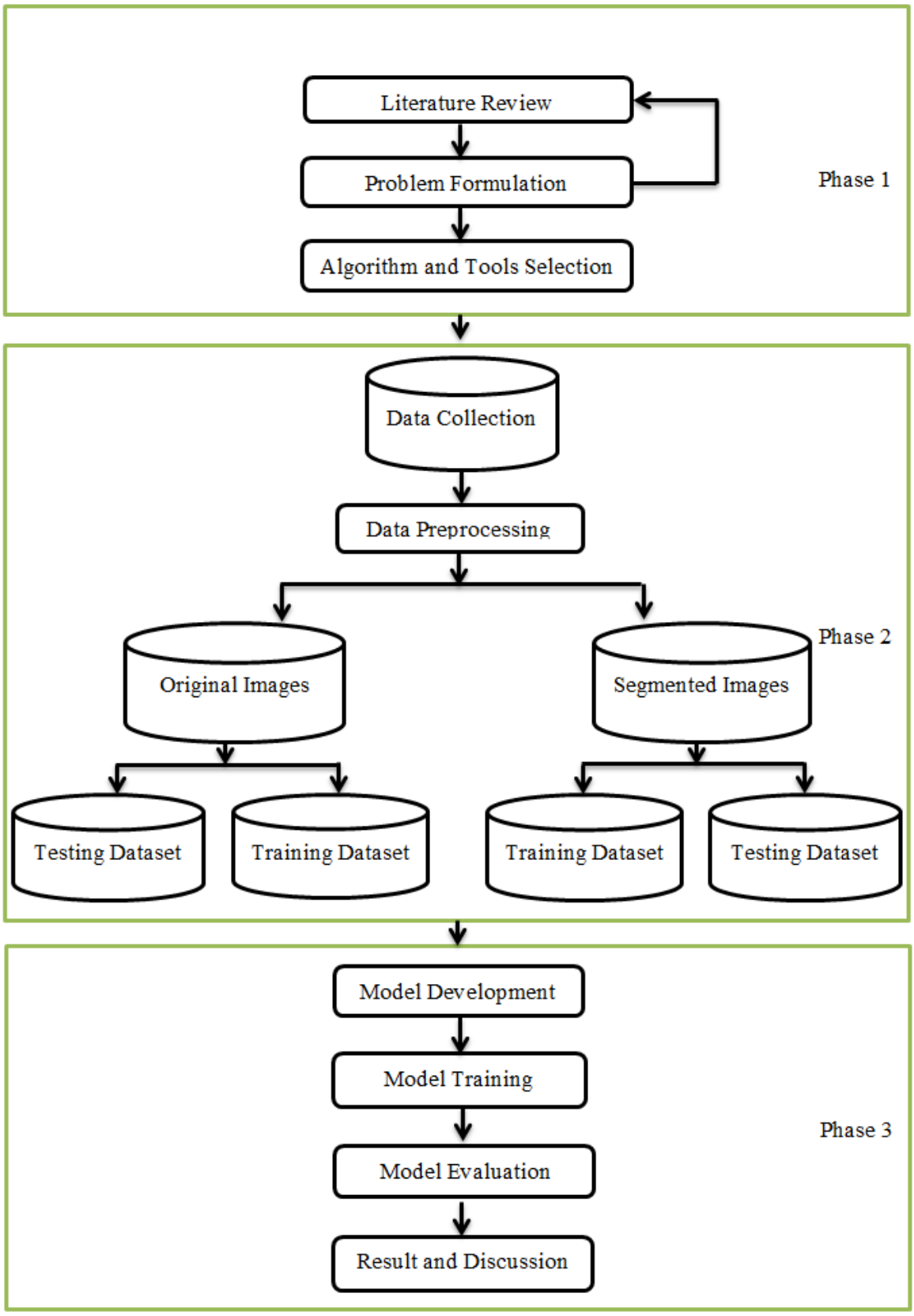


Figure 3. 2: Research methodology flowchart

3.3. Dataset

In this step, two main sources of data were used: First COVID-19-related skin images have been collected from DermNet and Bing images then split the input images into training and test datasets which are the category of four classes of the skin images. The four skin diseases in the datasets are: Urticarial rashes, Maculopapular rash, Livedo-like pattern, and Purpuric 'vasculitic' pattern. Examples for original images and segmented images datasets are involved for the experiment. The original images are cropped and segmented using ImageJ tool to prepare segmented images dataset for the proposed model and fine-tuning models experiment. The dataset is split into 80% training examples, 20% test examples and 20% of the training examples for validation examples. The main aim of this research is to generate a COVID-19-related skin disease classification based on the collected data that can work on the proposed CNN model.

3.4. Data Preprocessing

In this step data preprocessing has be performed before proceed with building the model. The collected images have different sizes with in the ranges between 150 and 1500 pixels in Convolutional Neural Network all the images should have the same size. So image resize is the basic step in image preprocess because of the data set must have similar size for the proposed work and the large pixel's size consumes too much computational cost and time. Preprocessing is required to clean image data for the model input. The all images are cropped to make convenient for segmentation processes. Since In convolutional neural networks required that all images are the same sized arrays, we resize the image to size of 224x224 pixels.

3.5. Image Segmentation

Here one of the solutions has been implemented for improvement of COVID-19-related skin diseases classification using segmented skin images. In this step segmentation of skin images and preparing the datasets for training CNN models are involved. ImageJ is used for the segmentation process of the images to separate a single image into separate components based on shape, size or intensity. the image filtered using bandpass, to make later thresholding more effective. And apply auto-threshold methods so that the result is reproducible later on the same data on the remaining image data to get the segmented images.

3.6. Data Augmentation

In the case of small datasets, data augmentation is a way of increasing data while training the classification model. Data augmentation is often used to improve generalization properties. We can multiply the size of the dataset with data augmentation. We will use it for creating images with different image transformation then it improves our network by applying different modifications on the image data.

Image augmentation creates new training examples out of existing training data. To get more images from the few training images and increase the accuracy of the model, we augmented the image data using random transformations. The data augmentation techniques used are: rotation of 20 degree, width shift of 0.1, height shift of 0.1, shear of 0.1, zoom of 0.2, brightness, size re-scaling, horizontal and vertical flipping. The data augmentation also prevents overfitting and generalization problems of the model. When the model is exposed with small datasets, the algorithm learns patterns that do not generalize to new data.

3.7. Model

The COVID-19-related skin disease CNN model is implemented to fit a small size dataset. Before training the CNN model, all images in the dataset were segmented and re-sized to 224x224 pixels to train the model and reduce the computational cost for processing.

The first layer consists of Convolutional 2D layer. The following are the Conv2D parameters:

- Number of Filters: 32
- Kernel/Filter Size: 3x3
- Activation Function: ReLu (Rectified Linear Unit. It's linear for all positive values, and zero for all negative values)
- Padding: Same (Pad so the output size is the same as the input size. Filter here is 3x3

Layer 2 consists of a MaxPool 2D layer, which acts as a downsampling filter. These are used to reduce computational cost, and to some extent also reduce overfitting. The following are the MaxPool2D parameters:

- Pool Size: 2x2

Layer 3 consists of a Dropout layer. It is a regularization technique where randomly selected neurons are ignored during training. This forces the network to learn features in a distributed way, while improving generalization and reduces overfitting.

Layer 4 consists of another Convolutional 2D layer. The following are the Conv2D parameters:

- Number of Filters: 64
- Kernel/Filter Size: 3x3
- Activation Function: ReLu (Rectified Linear Unit. It's linear for all positive values, and zero for all negative values)
- Padding: Same (Pad so the output size is the same as the input size. Filter here is 3x3

Layer 5 consists of a MaxPool 2D layer. The following are the MaxPool2D parameters:

- Pool Size: 2x2

Layer 6 consists of a Dropout layer.

Layer 7 consists of a Flatten layer. It flattens the input to a single 1D vector.

Layer 8 consists of a Dense layer. The following are the Dense layer parameters:

- Units: 128 (Dimensionality of the output space)
- Activation Function: ReLu

Layer 9 consists of a Dropout layer. The following are the Dropout parameters:

- Value: 0.5 (Basically half of the neurons are ignored during trianing by making their values = 0)

Finally, the last layer (Layer 12) consists of another Dense layer. The following are the Dense layer parameters:

- Units: 4 (Number of classes)
- Activation Function: Softmax

3.8. Performance Metrics for Classifiers

A classifier assigns each image to a class. This assignment is generally not perfect and images may be assigned to the wrong class. To evaluate a classifier, the actual class of the image must be known. To evaluate the classification quality, the class assigned by the classifier is compared with the actual class. This allows the images to be divided into the following four subsets:

- True positive (TP): the classifier correctly predicts the positive class.
- True negative (TN): the classifier correctly predicts the negative class.
- False positive (FP): the classifier incorrectly predicts the positive class.
- False negative (FN): the classifier incorrectly predicts the negative class.

Based on the cardinality of these subsets, statistical quantities for the classifier can now be calculated. A common and widely used quantity is accuracy, which is only a reasonable measure if the different classes in the dataset are approximately equally distributed.

Accuracy specifies the percentage of objects that have been correctly classified. Two other important metrics are sensitivity and specificity, which can be applied even if the different classes are not equally distributed. Sensitivity indicates the ratio of objects correctly classified as positive out of the total number of positive objects contained in the dataset. Specificity indicates the ratio of negative objects correctly classified as negative out of the total number of negative objects contained in the available dataset.

The output of a binary classifier is interpreted as a probability distribution over the classes. Normally, objects with an output value greater than 0.5 are assigned to the positive class in a binary classifier and objects with an output value less than 0.5 are assigned to the negative class. An alternative approach is used based on the receiver operating characteristic (ROC). The threshold used for classification systematically varies between 0 and 1, and the sensitivity and specificity are determined for each selected threshold. The ROC curve is calculated by plotting the sensitivity against 1-specificity and can be used to evaluate the classifier. The further the ROC curve deviates from the diagonal, the better the classifier . A suitable overall measure for the curve is the area under the curve.

3.8.1. Confusion Matrix

A Confusion Matrix in a classification problem is the tabular representation of prediction results with count values broken down by each class. As the name suggests, it shows how a classification model performs when making predictions. It gives insight into the errors being made by the classifier and the type of errors. A confusion matrix is also called an evaluation metric that is used to describe the performance of a classification task. Precision, Recall and Accuracy can all be calculated with the help of a confusion matrix.

Table 3. 1: Confusion Matrix

		Actual: Positive	Actual: Negative
Predicted	Positive	**TP**	**FP**
	Negative	**FN**	**TN**

- **Accuracy** $= \frac{TP + TN}{TP + TN + FP + FN}$
- **Sensitivity** $= \frac{TP}{TP + FN}$ = Recall
- **Specificity** = $\frac{TN}{TN + FP}$
- **Precision** = $\frac{TP}{TP + FP}$
- **F1 score** = $\frac{2}{1/precision + 1/recall}$

3.9. Materials and Tools

The model is trained using Google Colab cloud service, Tensorflow, Keras, and python3 framework with virtual GPU. And for the image preprocessing and segmentation, ImageJ software has been used. Google Colab is a free Jupyter notebook environment that runs on Google's cloud servers, letting the user leverage backend hardware like GPUs. Keras run on top of TensorFlow framework. It is made with focus of understanding deep learning techniques, such as creating layers for neural networks maintaining the concepts of shapes and mathematical details.

ImageJ is powerful software for the segmentation of image to separate a single image into separate components based on shape, size or intensity to make image analysis of each of those components possible.

3.9.1. Software Tools

Table 3. 2: Software tools

No.	Tools	Description
1	ImageJ	• ImageJ is powerful software for the segmentation of image to separate a single image into separate components based on shape, size or intensity to make image analysis of each of those components possible. For this we use "masks". • Masks are just a binary image being used for "Image calculation" (e.g. digital subtraction) to extract or remove specific structures.
2	Google Colab	• Google Colab is a free Jupyter notebook environment that runs on Google's cloud servers, letting the user leverage backend hardware like GPUs. This lets you do everything you can in a Jupyter notebook hosted in your local machine, without requiring the installations and setup for hosting a notebook in your local machine.
3	TensorFlow	• TensorFlow is a symbolic math library used for neural networks and is

		best suited for dataflow programming across a range of tasks. It offers multiple abstraction levels for building and training models.
4	Keras	• Keras is an effective high-level neural network Application Programming Interface (API) written in Python. This open-source neural network library is designed to provide fast experimentation with deep neural networks. • Keras run on top of TensorFlow framework. It is made with focus of understanding deep learning techniques, such as creating layers for neural networks maintaining the concepts of shapes and mathematical details.
5	Python	• Python is a general-purpose high level programming language that is widely used in machine learning and data science for producing deep learning algorithms.

3.9.2. Hardware Tools

Table 3. 3: Hardware tools

No.	Tools	Description
1	Laptop	Intel Core i7
2	Local CPU	2.50GHz
3	Local RAM	8GB
4	Virtual GPU	K80 GPU
5	Virtual RAM	12 GB

CHAPTER FOUR

4. IMPLEMENTATION

4.1. Overview

This chapter describes the implementation of the proposed solution using *ImageJ* for image analysis and *Jupyter Notebook* IDE for CNN models development. The datasets used for implementation are original and segmented images. The model is trained using Google Colab cloud service, Tensorflow, Keras, and python3 framework with GPU. And for the image preprocessing and segmentation, ImageJ software has been used. First baseline model is developed for the convenience of the comparison with the modified pre-trained models, then build fine-tune the VGG-16 model with the modified classifier and unfreeze convolution blocks 4 and 5 while keeping the first three blocks frozen.

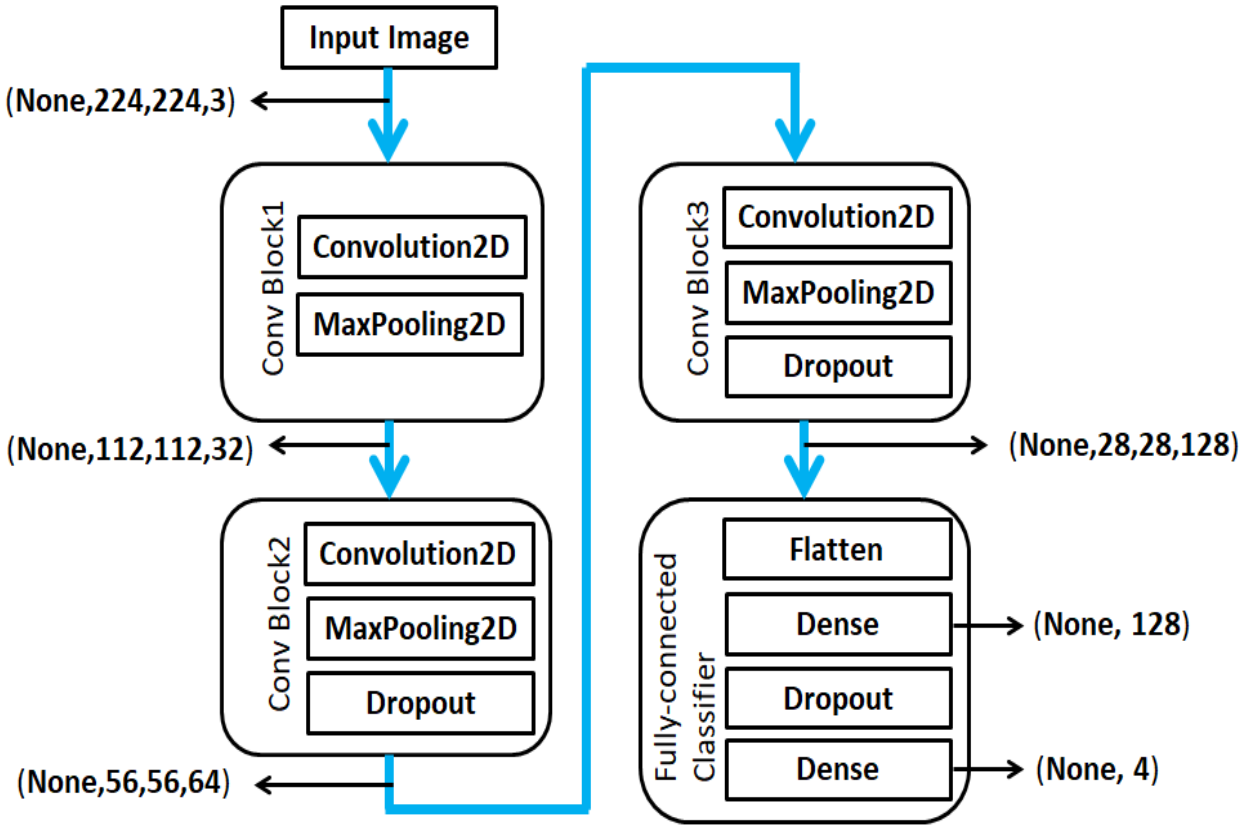


Figure 4. 1: Architecture of the Proposed Network

4.2. Preparing the Dataset

This step involves segmentation of the COVID-19-related skin images and preparing the datasets for training CNN models. Four skin diseases in the datasets are: Urticarial rashes, Maculopapular rash, Livedo-like pattern, and Purpuric 'vasculitic' pattern. Examples for original images/non-segmented images and segmented images datasets are involved for the experiment. The original images are cropped and segmented using ImageJ platform to prepare segmented images dataset for the proposed model and fine-tuning models experiment. The dataset is split into 80% training examples, 20% test examples and 20% of the training examples for validation examples.

Table 4. 1: Summary of the COVID-19-related skin diseases trainable datasets

Label	**Name**	**None-segmented training datasets**	**Segmented training datasets**
Livedo	Livedo-like pattern	1152	1152
Maculopapular	Maculopapular rash	1152	1152
Purpuric	Purpuric 'vasculitic' pattern	1152	1152
Urticarial	Urticarial rashes	1152	1152

4.3. Image segmentation example using ImageJ:

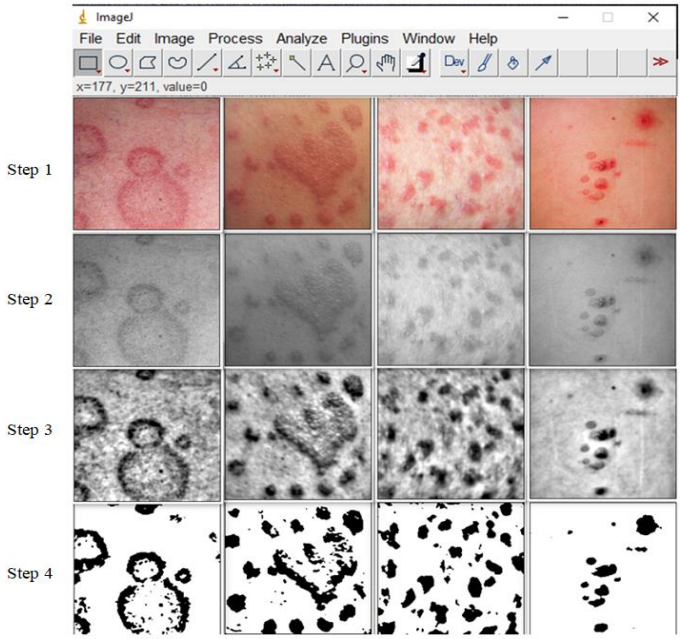


Figure 4. 2**:** Four sample images for segmentation process using ImageJ

4.4. Sample original and segmented images

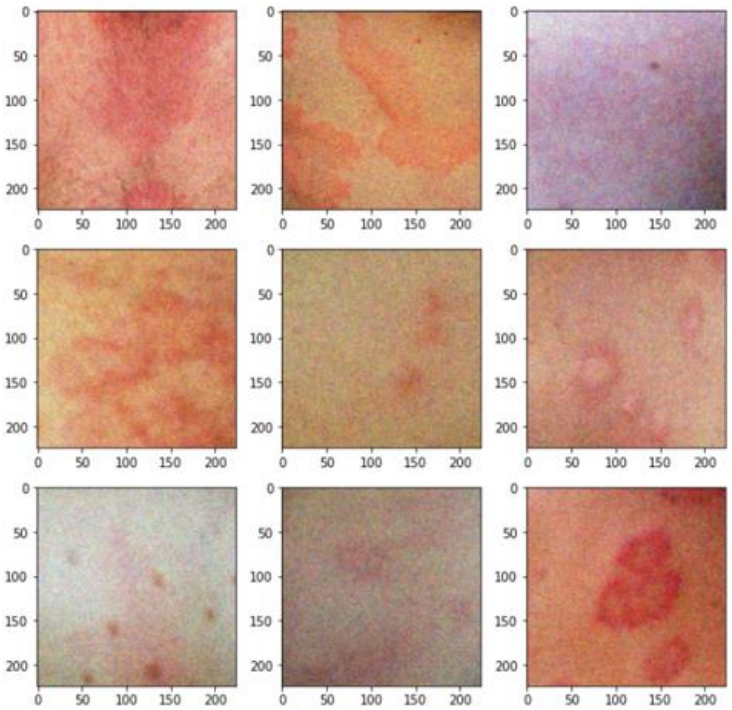


Figure 4. 3**:** Sample original images

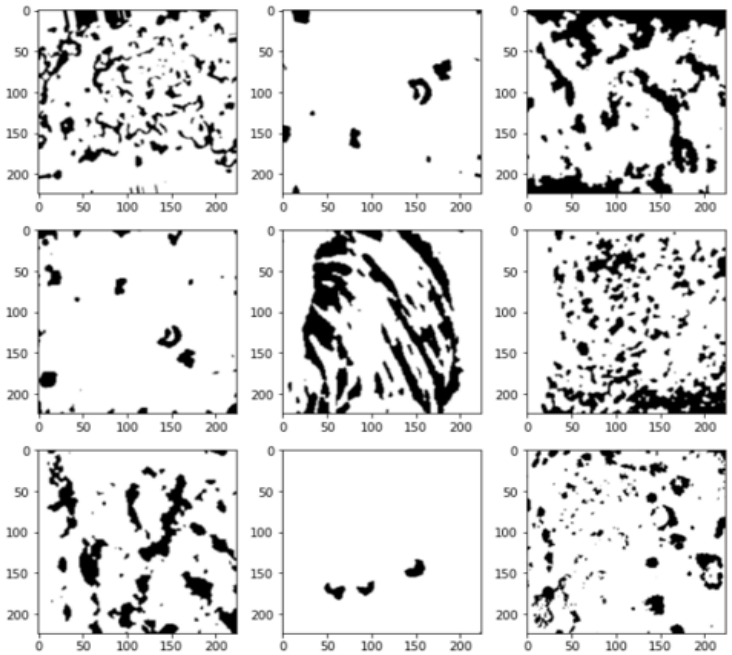


Figure 4. 4**:** Sample segmented images

4.5. Skin Disease Classification Model Implementation

The **Proposed model** is created using Jupyter notebook in keras API:

Conv →MaxPool →Dropout →Conv →MaxPool →Dropout →Flatten →Dense →Dropout →Dense (output)

a) The first layer consists of Convolutional 2D layer. The following are the Conv2D parameters:

- Number of Filters: 32
- Kernel/Filter Size: 3x3
- Activation Function: ReLu (Rectified Linear Unit. It's linear for all positive values, and zero for all negative values)
- Padding: Same (Pad so the output size is the same as the input size. Filter here is 3x3
- Input Shape = (224, 224, 3)

b) Layer 2 consists of a MaxPool 2D layer, which acts as a downsampling filter. These are used to reduce computational cost, and to some extent also reduce overfitting. The following are the MaxPool2D parameters:

- Pool Size: 2x2

c) Layer 3 consists of a Dropout layer. It is a regularization technique where randomly selected neurons are ignored during training. This forces the network to learn features in a distributed way, while improving generalization and reduces overfitting. The following are the Dropout parameters:

d) Layer 4 consists of another Convolutional 2D layer. The following are the Conv2D parameters:

- Number of Filters: 64
- Kernel/Filter Size: 3x3
- Activation Function: ReLu (Rectified Linear Unit. It's linear for all positive values, and zero for all negative values)

- Padding: Same (Pad so the output size is the same as the input size. Filter here is 3x3

e) Layer 5 consists of a MaxPool 2D layer. The following are the MaxPool2D parameters:

- Pool Size: 2x2

f) Layer 6 consists of a Dropout layer. The following are the Dropout parameters:

g) Layer 7 consists of a Flatten layer. It flattens the input to a single 1D vector.

h) Layer 8 consists of a Dense layer. The following are the Dense layer parameters:

- Units: 128 (Dimensionality of the output space)
- Activation Function: ReLu

i) Layer 9 consists of a Dropout layer. The following are the Dropout parameters:

- Value: 0.5 (Basically half of the neurons are ignored during trianing by making their values = 0)

j) Finally, the last layer (Layer 12) consists of another Dense layer. The following are the Dense layer parameters:

- Units: 4 (Number of classes)
- Activation Function: Softmax

4.5.1. The Skin Disease Classification model configuration:

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', padding = 'Same', input_shape = (2
24,224,3)))
model.add(MaxPool2D(pool_size = (2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', padding = 'Same'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.1))
model.add(Conv2D(128, (3, 3), activation='relu', padding = 'Same'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4, activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_12 (Conv2D)           (None, 224, 224, 32)      896
 max_pooling2d_6 (MaxPooling2D)  (None, 112, 112, 32)   0
 conv2d_13 (Conv2D)           (None, 112, 112, 64)      18496
 max_pooling2d_7 (MaxPooling2D)  (None, 56, 56, 64)     0
 dropout_19 (Dropout)         (None, 56, 56, 64)        0
 conv2d_14 (Conv2D)           (None, 56, 56, 128)       73856
 max_pooling2d_8 (MaxPooling2D)  (None, 28, 28, 128)    0
 dropout_20 (Dropout)         (None, 28, 28, 128)       0
 flatten_2 (Flatten)          (None, 100352)            0
 dense_4 (Dense)              (None, 128)               12845184
 dropout_21 (Dropout)         (None, 128)               0
 dense_5 (Dense)              (None, 4)                 516
=================================================================
Total params: 12,938,948
Trainable params: 12,938,948
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. 1**:** The Skin Disease Classification model summary

4.6. Pre-trained VGG16 models

The pre-trained VGG-16 is an already trained model on a huge dataset (ImageNet) with a lot of diverse image categories. Considering this fact, the model should have learned a robust hierarchy of features, which are spatial, rotation, and translation invariant. Hence, the model, having learned a good representation of features for over a million images belonging to 1,000 different categories, can act as a good feature extractor for new images suitable for computer vision problems. These new images might never exist in the ImageNet dataset or might be of totally different categories, but the model should still be able to extract relevant features from these images, considering the principles of transfer learning.

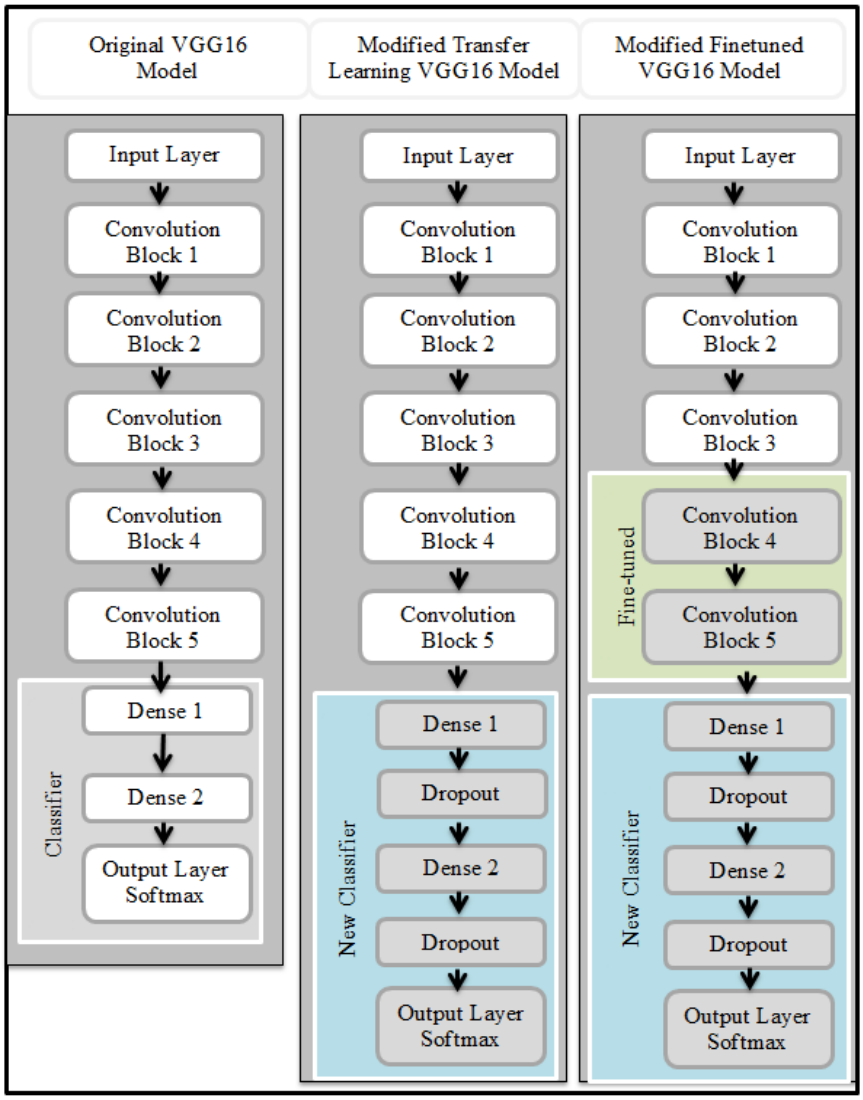


Figure 4. 1**:** Block diagram of the original and modified pre-trained VGG16 models

4.7. Transfer-learning VGG16 model implementation

In this model two dropout layers are added to avoid over-fitting and the last two dense layers and output dense layers are also modified. The transfer learning on COVID-19-related skin problem gives an advantage of using pre-trained models as effective modified feature extractors and fine-tuned model for the new and small amount of images.

4.7.1. Model configuration:

Make all layers untrainable by freezing weights except the last 3 fully connected layers and this network is used as feature extractor. To load modified Vgg16 model without the fully connected layers the following code is implemented:

```
for layer in model.layers[:-5]:
    layer.trainable = False
    # ensure the last 3 fc layers are trainable/not frozen
for layer in model.layers[-5:]:
    layer.trainable = True
# check to see it is correct:
for layer in model.layers:
    print(layer.trainable)
```

- **Code to generate the model summary**

```
from keras.applications.vgg16 import VGG16, preprocess_input
model = VGG16(include_top=True, weights='imagenet')

placeholder = Input(shape=(224, 224, 3))
model = VGG16(input_tensor=placeholder,
            include_top=True,
            weights='imagenet')
last_layer = model.get_layer('block5_pool').output
x = Flatten(name='flatten')(last_layer)
x = Dense(512, activation='relu', name='fc1')(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu', name='fc2')(x)
x = Dropout(0.5)(x)
out = Dense(units=4, activation='softmax', name='output')(x)
model = Model( placeholder, out)
for layer in model.layers[:-5]:
layer.trainable = False
model.summary()
```

4.7.1.1. Model summary

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 224, 224, 3)]     0
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168
 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
 flatten (Flatten)           (None, 25088)             0
 fc1 (Dense)                 (None, 512)               12845568
 dropout_6 (Dropout)         (None, 512)               0
 fc2 (Dense)                 (None, 512)               262656
 dropout_7 (Dropout)         (None, 512)               0
 output (Dense)              (None, 4)                 2052
=================================================================

Total params: 27,824,964
Trainable params: 13,110,276
Non-trainable params: 14,714,688
```

Figure 4. 2**:** The Transfer-learning model summary

4.8. Finetuned VGG16 model implementation

In this model two dropout layers are added to avoid over-fitting as well as the last two dense layers and output dense layers are also modified to the new layers as transfer learning model. But here the last two convolution blocks, block4 and block5, are finetuned to be trainable and updated in every epoch.

4.8.1. Model configuration:

Make all layers untrainable by freezing weights except the last 3 fully connected layers and this network is used as feature extractor. To load modified Vgg16 model without the fully connected layers the following code is implemented:

```
for layer in model.layers[:-14]:
   layer.trainable = False
for layer in model.layers[-14:]:
   layer.trainable = True
for layer in model.layers:
   print(layer.trainable)
```

- **Code to generate the model summary**

```
placeholder = Input(shape=(224, 224, 3))
model = VGG16(input_tensor=placeholder,
            include_top=True,         # the VGG16 script will automatically download the
            weights='imagenet')       # weigths if they are stored not locally

last_layer = model.get_layer('block5_pool').output
x = Flatten(name='flatten')(last_layer)
x = Dense(128, activation='relu', name='fc1')(x)
x = Dropout(0.1)(x)
x = Dense(256, activation='relu', name='fc2')(x)
x = Dropout(0.3)(x)
out = Dense(units=4, activation='softmax', name='output')(x)
model = Model( placeholder, out)
for layer in model.layers[:-14]: # fix weights from all but the new layer
   layer.trainable = False
model.summary()
```

4.8.1.1. Model summary

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_7 (InputLayer)        [(None, 224, 224, 3)]     0
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168
 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
 flatten (Flatten)           (None, 25088)             0
 fc1 (Dense)                 (None, 512)               12845568
 dropout_8 (Dropout)         (None, 512)               0
 fc2 (Dense)                 (None, 512)               262656
 dropout_9 (Dropout)         (None, 512)               0
 output (Dense)              (None, 4)                 2052
=================================================================
Total params: 27,824,964
Trainable params: 26,089,476
Non-trainable params: 1,735,488
_________________________________________________________________
```

Figure 4. 3**:** The Finetuned model summary

CHAPTER FIVE

5. RESULTS AND DISCUSSIONS

5.1. Overview

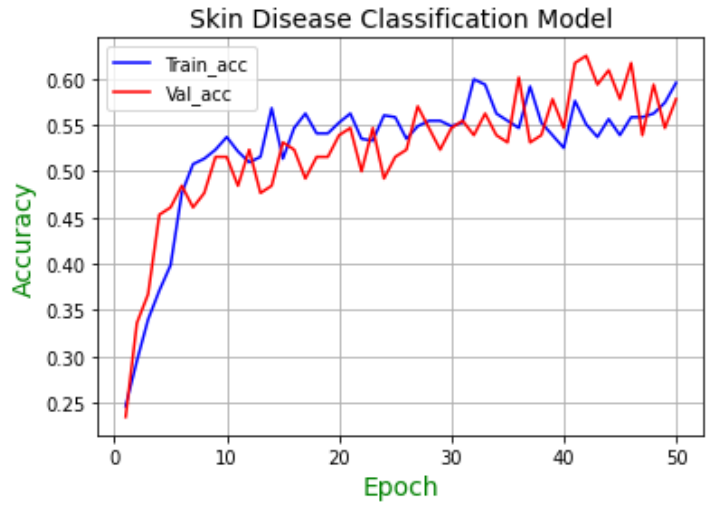
In this chapter, the results that have been obtained in the skin disease classification using the three CNN models experiments are reported. The models are: Skin Disease Classification model (Proposed model), transfer learning VGG16 model and finetuned VGG16 model. All the models are trained in two types of dataset which are original images/none segmented images dataset and segmented images dataset. All the models have been trained in epochs 50, and the output obtained of the training and validation accuracies are evaluated using confusion matrix.

5.2. Results of models accuracy and loss:

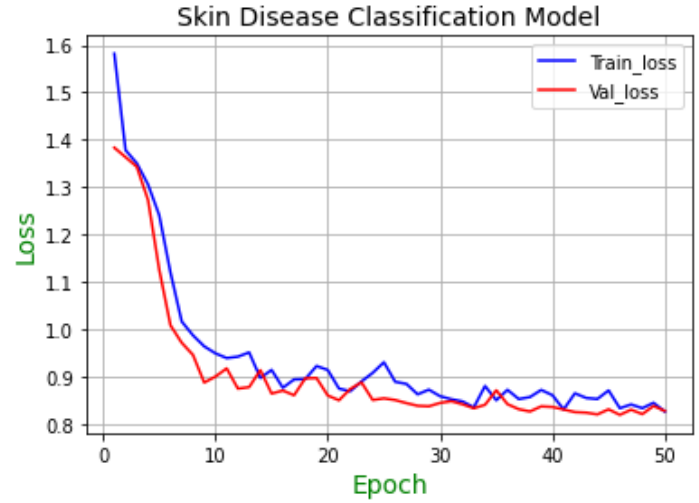
The graphs show visualization report to ensure the model kept learning with every epoch, improving its accuracy and reducing its losses and errors.

5.2.1. For Skin Disease Classification Model of original images dataset

The model has trained in epochs 50, and the output obtained is training accuracy of 60% and validation accuracy of 58% in case of original images dataset. The accuracy values are not good enough here, and the model is also overfitting on both the training and validation dataset. The output of the training and validation loss is 0.83 and 0.83 respectively it has also high losses. To overcome this problem segmented images are employed to improve the accuracy.




```
Training acuracy =    0.60
Validation accuracy = 0.58
```

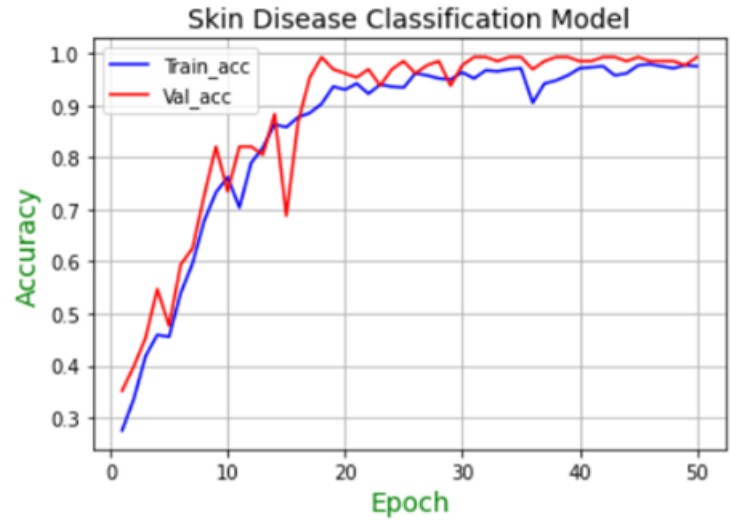



```
Training loss =   0.83
Validation loss = 0.83
```

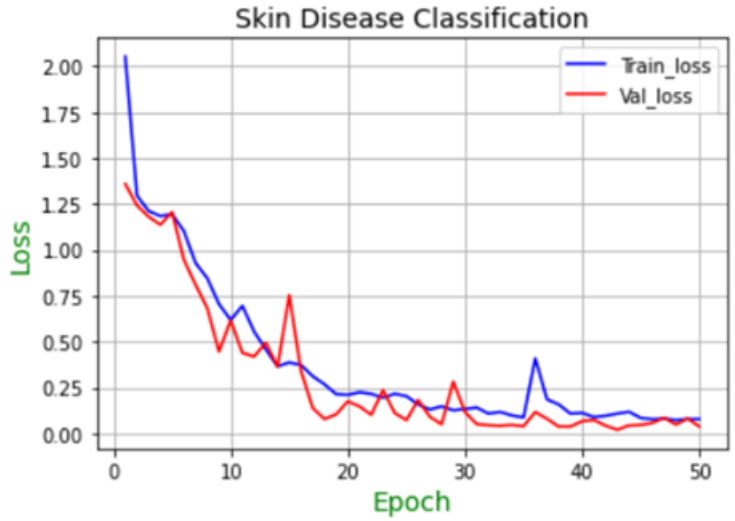
Figure 5. 1: Skin Disease Classification model accuracy and loss results of original images

5.2.2. For Skin Disease Classification Model of segmented images dataset

The model has trained in epochs 50, and the output obtained is training accuracy of 97% and validation accuracy of 99% in case of segmented images dataset. The accuracy values are better here, and the model is also good fitting both on the training and validation dataset. The output of the training and validation loss is 0.08 and 0.04 respectively. This implies that segmented images are more useful for the image detection and classification problems while our images are small in quantity.




```
Training acuracy =    0.97
Validation accuracy = 0.99
```




```
Training loss =   0.08
Validation loss = 0.04
```

Figure 5. 2: Skin Disease Classification model accuracy and loss results of segmented images

5.2.3. For Transfer learning VGG16 Model of original images dataset

The model has trained in epochs 50, and the output obtained is training accuracy of 72% and validation accuracy of 69% in case of original images dataset. This implies that the model does not converge, since training accuracy value is low and the validation accuracy is higher. The output of the training and validation loss is 0.59 and 0.61 respectively. Since the training loss is larger than the test error so the model got overfitting.

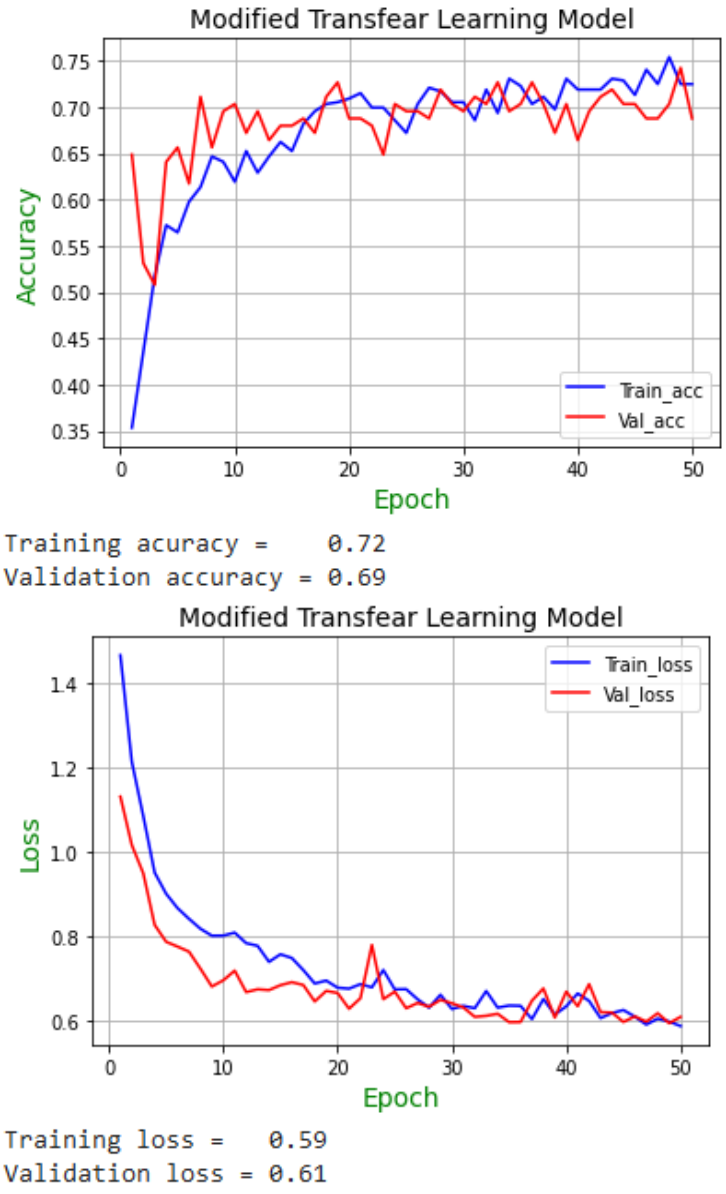
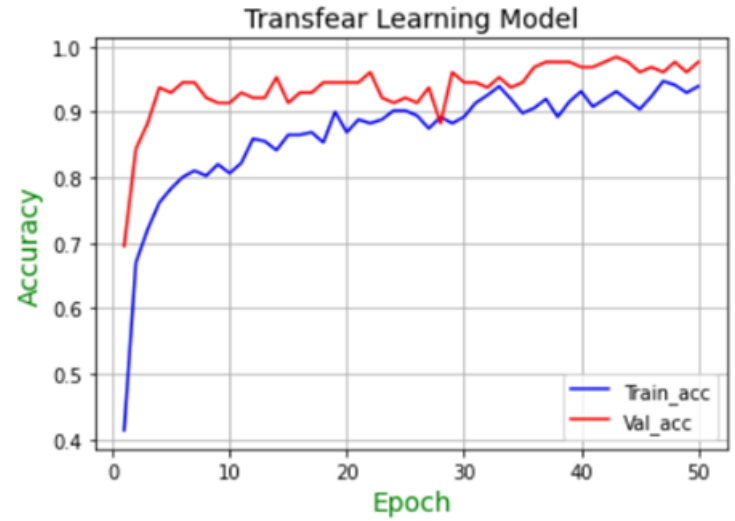


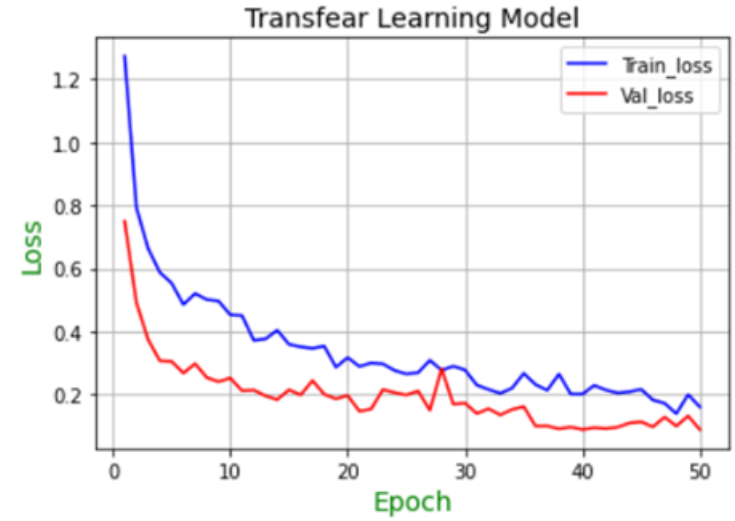
Figure 5. 3: Transfer-learning model accuracy and loss results of original images

5.2.4. For Transfer learning VGG16 Model of segmented images dataset

The model has trained in epochs 50, and the output obtained is training accuracy of 94% and validation accuracy of 98% in case of segmented images dataset. Here also the model does not converge, since the training accuracy value is lower than the validation accuracy. The output of the training and validation loss is 0.18 and 0.09 respectively. Since the training loss is larger than the test error so the model here also got overfitting.



Training acuracy = 0.94
Validation accuracy = 0.98



Training loss = 0.16
Validation loss = 0.09

Figure 5. 4: Transfer-learning model accuracy and loss results of segmented images

5.2.5. For Finetuned VGG16 Model of original images dataset

The model has trained in epochs 50, and the output obtained is training accuracy of 83% and validation accuracy of 68% in case of original images dataset. Here the model is noisy. The training accuracy value is low but the validation accuracy is better. The output of the training and validation loss is 0.46 and 1.09 respectively. Since the training loss is larger than the test error so the model here also has overfitting.

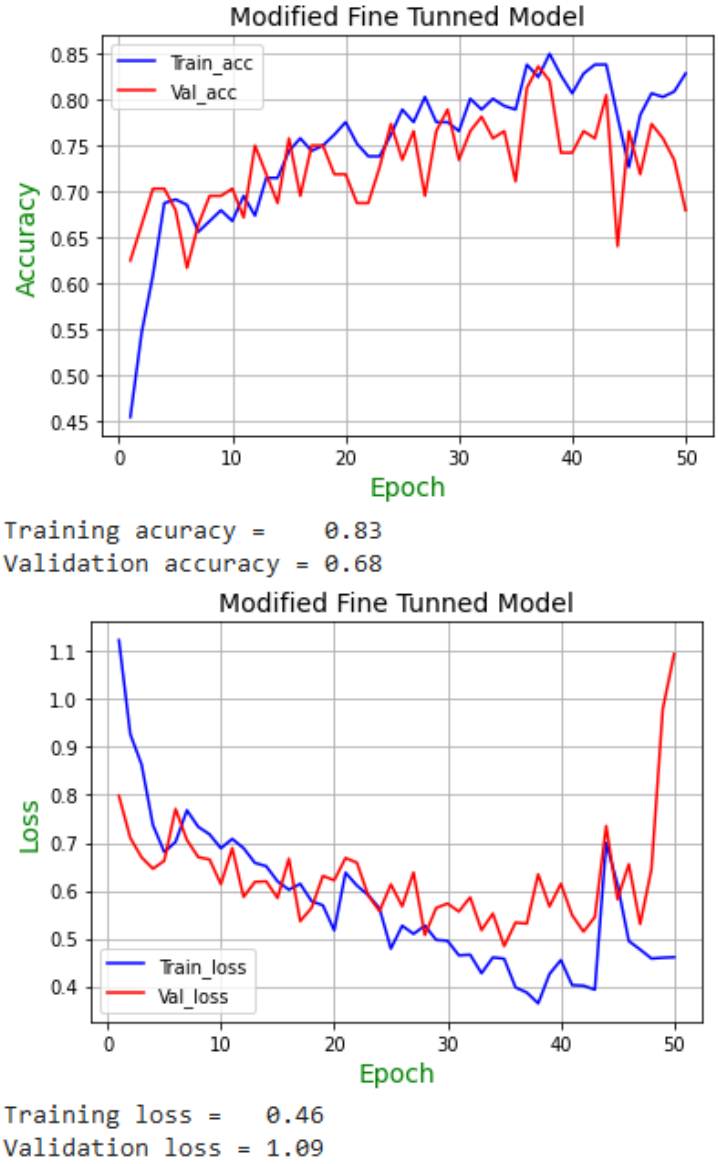
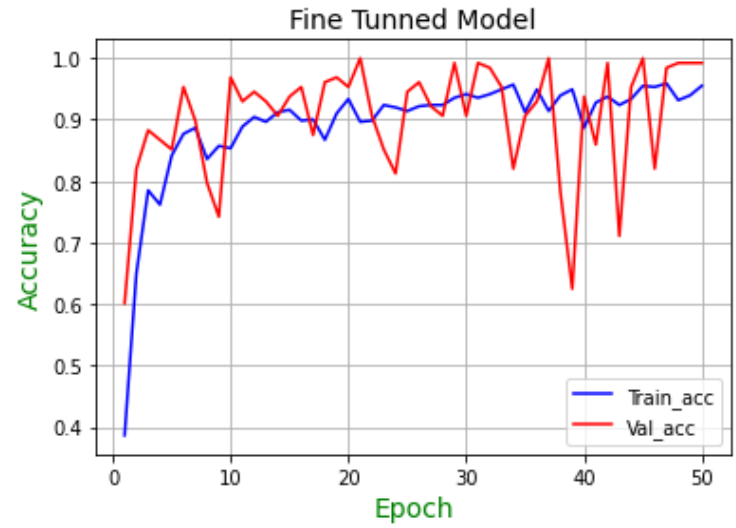


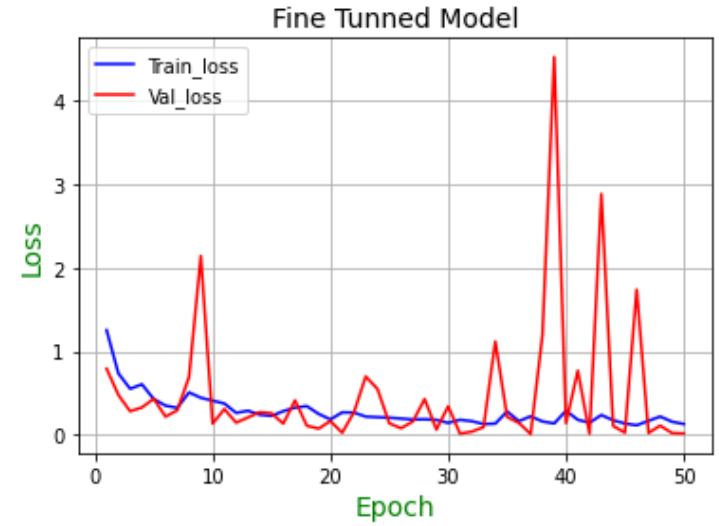
Figure 5. 5: Finetuned model accuracy and loss results of original images

5.2.6. For Finetuned VGG16 Model of segmented images dataset

The model has trained in epochs 50, and the output obtained is training accuracy of 96% and validation accuracy of 99% in case of original images dataset. Here the model is also noisy. The training accuracy value is also low but the validation accuracy is better. The output of the training and validation loss is 0.13 and 0.02 respectively. Since the training loss is larger than the test error so the model here also has overfitting.




```
Training acuracy =    0.96
Validation accuracy = 0.99
```




```
Training loss =   0.13
Validation loss = 0.02
```

Figure 5. 6: Finetuned model accuracy and loss results of segmented images

5.3. Confusion matrices results:

The performance of the proposed architecture is evaluated based on the confusion matrix report how the skin diseases are classified correctly.

5.3.1. For Skin Disease Classification Model

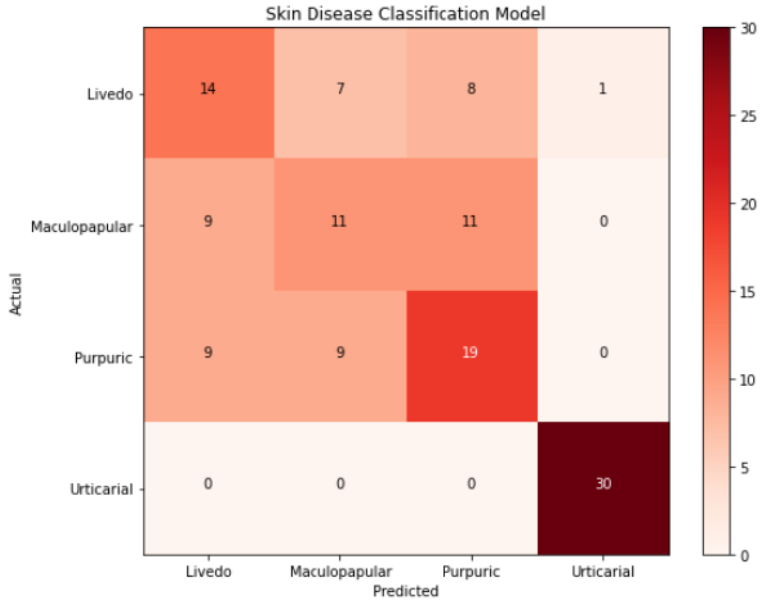


Figure 5. 7: Skin disease classification model confusion matrix graph of original images

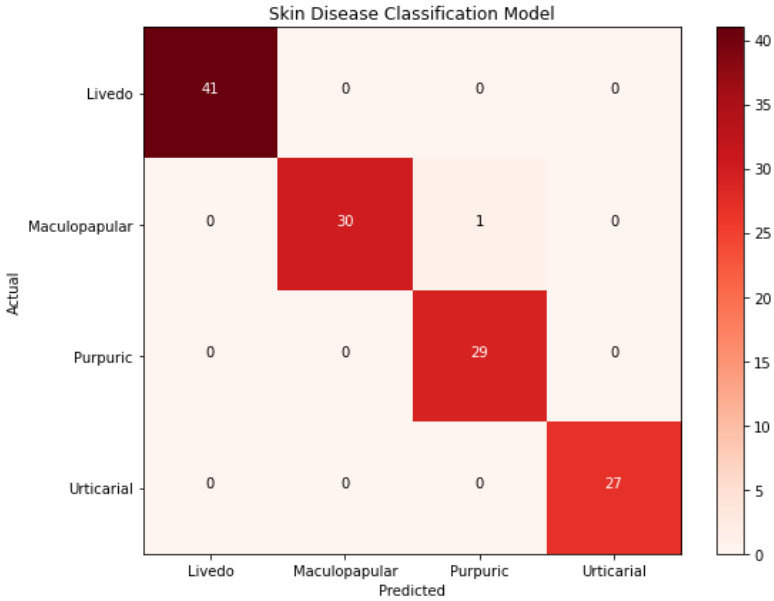


Figure 5. 8: Skin disease classification model confusion matrix graph of segmented images

5.3.2. For Transfer Learning VGG16 Model

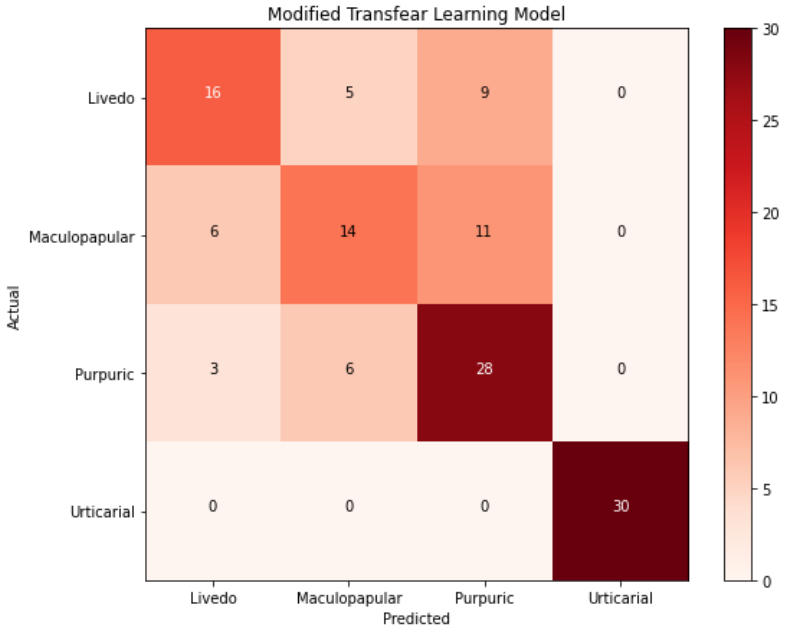


Figure 5. 9: Transfer-learning model confusion matrix graph of original images

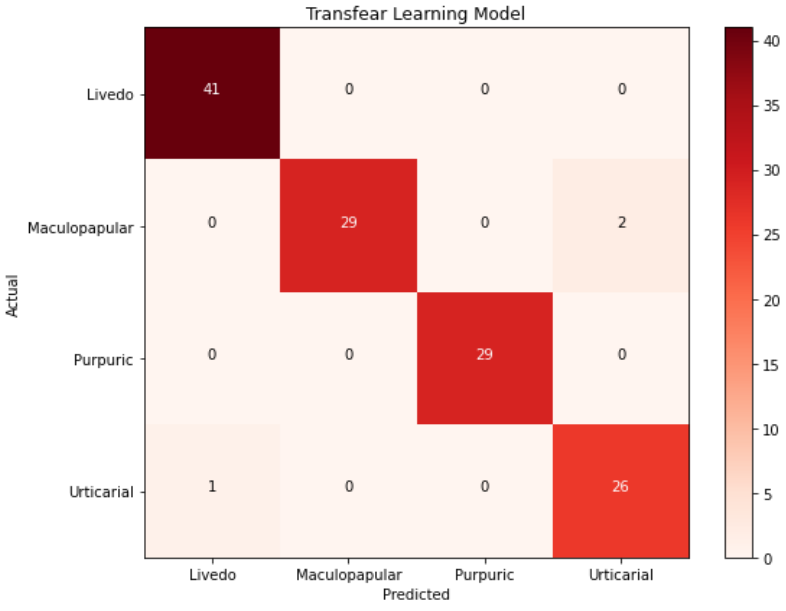


Figure 5. 10: Transfer-learning model confusion matrix graph of segmented images

5.3.3. For Finetuned VGG16 Model

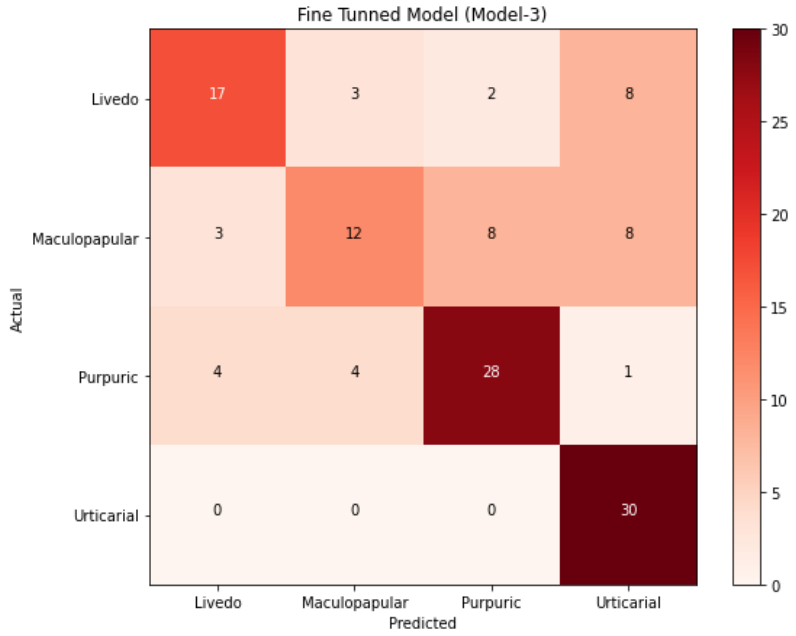


Figure 5. 11: Finetuned model confusion matrix graph of original images

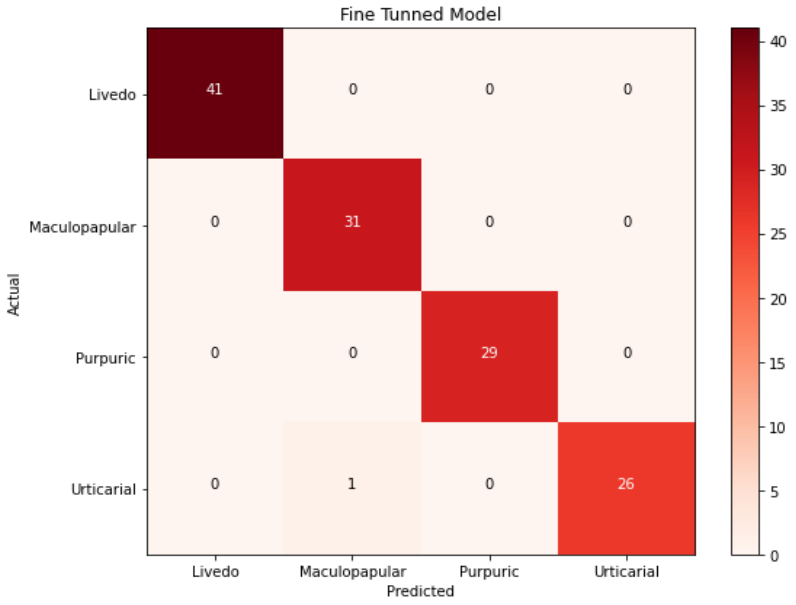


Figure 5. 12: Finetuned model confusion matrix graph of segmented images

5.4. Summary of confusion matrices classification report:

5.4.1. For Skin Disease Classification Model

	precision	recall	f1-score	support
Livedo	0.44	0.47	0.45	30
Maculopapular	0.41	0.35	0.38	31
Purpuric	0.50	0.51	0.51	37
Urticarial	0.97	1.00	0.98	30
accuracy			0.58	128
macro avg	0.58	0.58	0.58	128
weighted avg	0.57	0.58	0.57	128

Figure 5. 13: Skin Disease Classification model classification report of original images

	precision	recall	f1-score	support
Livedo	1.00	1.00	1.00	41
Maculopapular	1.00	0.97	0.98	31
Purpuric	0.97	1.00	0.98	29
Urticarial	1.00	1.00	1.00	27
accuracy			0.99	128
macro avg	0.99	0.99	0.99	128
weighted avg	0.99	0.99	0.99	128

Figure 5. 14: Skin Disease Classification model classification report of segmented images

5.4.2. For Transfer Learning VGG16 Model

	precision	recall	f1-score	support
Livedo	0.64	0.53	0.58	30
Maculopapular	0.56	0.45	0.50	31
Purpuric	0.58	0.76	0.66	37
Urticarial	1.00	1.00	1.00	30
accuracy			0.69	128
macro avg	0.70	0.69	0.69	128
weighted avg	0.69	0.69	0.68	128

Figure 5. 15: Transfer-learning confusion matrix classification report of original images

	precision	recall	f1-score	support
Livedo	0.98	1.00	0.99	41
Maculopapular	1.00	0.94	0.97	31
Purpuric	1.00	1.00	1.00	29
Urticarial	0.93	0.96	0.95	27
accuracy			0.98	128
macro avg	0.98	0.97	0.98	128
weighted avg	0.98	0.98	0.98	128

Figure 5. 16: Transfer-learning classification report of segmented images

5.4.3. For Fine-tuned VGG16 Model

	precision	recall	f1-score	support
Livedo	0.71	0.57	0.63	30
Maculopapular	0.63	0.39	0.48	31
Purpuric	0.74	0.76	0.75	37
Urticarial	0.64	1.00	0.78	30
accuracy			0.68	128
macro avg	0.68	0.68	0.66	128
weighted avg	0.68	0.68	0.66	128

Figure 5. 17: Fine-tuned classification report of original images

	precision	recall	f1-score	support
Livedo	1.00	1.00	1.00	41
Maculopapular	0.97	1.00	0.98	31
Purpuric	1.00	1.00	1.00	29
Urticarial	1.00	0.96	0.98	27
accuracy			0.99	128
macro avg	0.99	0.99	0.99	128
weighted avg	0.99	0.99	0.99	128

Figure 5. 18: Fine-tuned classification report of segmented images

Table 4. 2: Results summary of the models accuracy

Models	Dataset	Training accuracy (%)	Validation accuracy (%)	Epochs
Skin Disease Classification Model	Original images	60	58	50
	Segmented images	**97**	**99**	**50**
Transfer-learning VGG16 Model	Original images	84	95	50
	Segmented images	94	98	50
Fine-tuned VGG16 Model	Original images	83	68	50
	Segmented images	96	99	50

Table 4. 3**:** Results summary of the model loss

Models	Dataset	Training loss	Validation loss	Epochs
Skin Disease Classification Model	Original images	0.83	0.83	50
	Segmented images	**0.08**	**0.04**	**50**
Transfer-learning VGG16 Model	Original images	0.59	0.61	50
	Segmented images	0.16	0.09	50
Fine-tuned VGG16 Model	Original images	0.46	1.09	50
	Segmented images	0.13	0.02	50

Discussion

The models evaluation was performed using the same partitioning of training and testing dataset and the number of epochs for each model was 50. The result obtained from the segmented-image dataset achieved the best accuracy of 99% for the proposed model. So the above result shows that the proposed CNN model with segmented image dataset give us the best result than the other two modified transfer learning and fine-tuning VGG16 models. These image segments better represent the lesion in the skin disease than do the original skin images. It also decreased training and validation losses. The result of these segmented images changed the characteristics of the image into more meaningful ones, thus facilitated the interpretation and classification processes. Since segmentation is the process of defining homogeneous pixels into similar image segments, Segment-based classification approach gave us better accuracies than that of the traditional pixel-based classification methods.

CHAPTER SIX

6. CONCLUSION AND FUTURE WORK

6.1. Conclusion

In this thesis, the proposed work was compared to evaluate the performance with original and segmented images datasets for the COVID-19-related skin disease classification problem, and all the images are segmented by using ImageJ software. Three models implemented for evaluation of the classification performance ware: the proposed model (Skin Disease Classification CNN model), transfer learning and fine-tuned VGG16 CNN models.

In this experiment, the result showed how the proposed model with segmented-image dataset improved the accuracy and training speed of the model. The experiment included the COVID-19-related skin diseases of original and segmented image datasets in four classes. We implemented transfer learning and fine-tune VGG16 model for comparison with the proposed model. Training the whole layers of the pre-trained CNN can be challenging in terms of time and resources. Fine-tuning the whole model not only gives better result but also helps the model converge faster than fine-tuning the top layers only. One basic problem observed during the training is over-fitting and convergence problems. Both the transfer learning and finetuned models shows on the experiments over-fit the training data. Many methods are used to minimize over-fitting, but we minimized the over-fitting problem using image augmentation and dropout in this work.

The main contributions of this thesis can be summarized as follow:

- It included a newly proposed dataset of COVID-19-related skin disease.
- It enhanced the accuracy of the proposed model by using segmented skin dataset.
- Since we used segmented images, it minimized the computational power and increased the training speed of the model,

6.2. Recommendation

This research can be used for other biomedical images to attain further performance advancement in the medical field for image recognition and classification. ImageJ tool was used for the image segmentation of skin diseases. In the case of segmented images, the proposed model achieved better results than the pretrained VGG16 model for skin disease classification problem, since ImageNet data are not specialized in skin data.

Some gaps are observed from this proposed work such as over-fitting, so we need to increase the dataset to minimize the gaps. The proposed work needs improvement to increase the performance. Therefore, the following are some recommendations of the study:

- In this research the amount of dataset used are not much. It is better to increase dataset since deep learning methods need huge amount of data.
- Since image segmentation has a major role for the performance of classification, check other segmentation techniques to increase the performance.
- Compare the model with other pre-trained CNN architectures.

6.3. Future Work

To improve the skin disease classification model result, we use ensemble learning technique to build more effective model by combining fine-tuned VGG16 and the proposed models. One major problem observed during training the model is over-fitting, so we should find more images for this dataset for the future work to avoid over-fitting. Better training strategy will help the models to have best result. Do fine-tuning the whole model with more data instead of fine-tuning the top layers only to get better result and it also increases the model accuracy.

REFERENCES

A.V. MarzanoiD, N. Cassano, et al. (2020). *Cutaneous manifestations in patients with COVID-19*: a preliminary review of an emerging issue: British Journal of Dermatology 183, pp431–442

Giovanni Genovese, Chiara Moltrasio et al. (2020). *Skin Manifestations Associated with COVID-19*: Current Knowledge and Future Perspectives; Dermatology Unit, S. Karger AG, Basel

Giulia Daneshgaran, Danielle P. Dubin et al. (2020). *Cutaneous Manifestations of COVID-19*: An Evidence-Based Review; Springer Nature Switzerland AG

Ling-Fang Li, Xu Wang et al. (2020). 'Deep Learning in Skin Disease Image Recognition: A Review', *IEEE*.

Chun Hei and Michael Chan (2020) *Convolutional Neural Networks: Why are they so good for image related learning?* [Online]. Available: http://towardsdatascience.com/convolutional-neural-networks-why-are-they-so-good-for-image-related-learning-2b202a25d757

Prafful Mishra (2019). *Why are Convolutional Neural Networks good for Image classification*, [Online]. Available: https://medium.datadriveninvestors.com/why-are-convolutional-neural-networks-good-for-image-classification-146ec6e865e8

Ren LL, Wang YM et al. (2020). *Identification of a novel coronavirus causing severe pneumonia in human: a descriptive study*. Chin Med J.:133(9):1015–24.

World Health Organization (WHO) (2020) WHO director-general's opening remarks at the media briefing on COVID-19. 11 Mar 2020. [Online]. Available: https://www.who.int/dg/speeches/detail/who-director-general-s-opening-remarks-at-the-mediabriefing-on-covid-19%2D%2D-11-march-2020.

Livingston E. and Bucher K. (2020) *Coronavirus disease 2019 (COVID-19) in Italy*, JAMA; 323(14):1335

Drew DA, Nguyen LH et al. (2020). *Rapid implementation of mobile technology for real-time epidemiology of COVID-19*, Science; 368: 1362–7.

Marzano AV, Cassano N, et al. (2020). *Cutaneous manifestations in patients with COVID-19*: A preliminary review of an emerging issue. *Br J Dermatol* published online June 1. DOI:10.1111/bjd.19264.

"First case of COVID-19 confirmed in Ethiopia." (2020) Available: https://www.afro.who.int/news/first-case-covid-19-confirmed-ethiopia

Andargachew Mulu, Amsalu Bekele, et al. (2021). *The challenges of COVID-19 testing in Africa*: the Ethiopian experience: PanAfrican Medical Journal, 05 Jan 2021.

"Skin explained" (2019). *Sinclair Dermatology, department of Health*. Available: https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/skin#bhc-content

François Chollet (2018). *Deep Learning with Python*, Manning Publications Co., Shelter Island, NY 11964.

Alin Laurentiu, Tatu et al. (2021). *A Working Hypothesis on Vesicular Lesions Related to COVID-19 Infection*, Koebner Phenomena Type V, and a Short Review of Related Data: Clinical, Cosmetic and Investigational Dermatology :14 419–423

A. Rao (2019) *Convolutional Neural Network Tutorial (CNN) – Developing An Image Classifier In Python Using TensorFlow*, [Online]. Available: https://www.edureka.co/blog/convolutional-neural-network/

Cornelisse D. (2018) *An Intuitive Guide to Convolutional Neural Networks*, [Online]. Available: https://medium.com/free-code-camp/an-intuitive-guide-to-convolutionalneural-networks-260c2de0a050

Prabhu (2018) *Understanding of Convolutional Neural Network (CNN)*, Deep Learning. [Online]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutionalneural-network-cnn-deep-learning-99760835f148

Corvil.com (2019) What is the difference between 'SAME' and 'VALID' padding in tf.nn.max_pool of tensorflow? [Online]. Available: https://www.corvil.com/kb/what-is-the-difference-between-same-and-validpadding-in-tf-nn-max-pool-of-tensorflow

Dertat, A. et al. (2017). *Applied Deep Learning Part 4: Convolutional Neural Networks*. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Saha, S. et al. (2018). *A Comprehensive Guide to Convolutional Neural Networks*, The ELI5 way. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutionalneural-networks-the-eli5-way-3bd2b1164a53

Dertat, A. et al. (2017). *Applied Deep Learning Part 1: Artificial Neural Networks*. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-1-artificialneural-networks-d7834f67a4f6?#860e

Deshpande, A. et al. (2016). *A Beginner's Guide to Understanding Convolutional Neural Networks*. [Online]. Available: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-UnderstandingConvolutional-Neural-Networks/

Coursera (2017) *Normalizing inputs-deeplearning.ai* [online]. Available: https://www.coursera.org/lecture/deep-neural-network/normalizinginputs.-lXv6U

Daniel López et al. (2018). *Deep neural networks and transfer learning applied to multimedia web mining*, https://www.researchgate.net/publication/318171618

D. S. Gupta (2017), *Transfer learning and the art of using Pre-trained Models in Deep Learning*. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/06/transfer- learning-theart-of-fine-tuning-a-pre-trained-model/

H. C. Shin et al. (2016). *Deep Convolutional Neural Networks for Computer-aided Detection:* CNN Architectures, Dataset Characteristics, and Transfer Learning, in IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1285-1298.

Sparavigna (2020) On the Use of ImageJ Segmentation, *ResearchGate, 12 January 2020*.

Nikhil B. (2018) Image Data Pre-Processing for Neural Networks [online]. Available: https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258.

S.Gopal Krishna Patro, Kishore Kumar sahu (2017) 'Normalization: A Preprocessing Stage'. https:// normalization.ai/image-data-pre-processing-for-neural-networks-6725821

Haofu et al. (2016) 'A deep learning approaches to universal skin disease classification.' *University of Rochester Department of Computer Science, CSC*.

Ulzii-Orshikh et al. (2018) *The skin cancer classification using deep convolutional neural networks*, Multimedia Tools and Applications77.8, 9909-9924.

Habib Safigholi, Rezvantalab, et al. (2018). 'Dermatologist level dermoscopy skin cancer classification using different deep learning convolutional neural networks algorithms.' *arXiv preprint arXiv*: 1810.10348.

P. Marcelino (2018) *Transfer learning from pre-trained models*, https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751.

Q. Guan (2019). *Deep convolutional neural network VGG-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images: a pilot study*, Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6775529/.

Keras (2020) 'Keras Applications', *Last accessed*: June 2020, Available: https://keras.io/api/applications/.

D. Sarkar (2018). *A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning*, Available: https://towardsdatascience.com/acomprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-indeep-learning-212bf3b2f27a.

Harris (2020) ENVIConfustionMatrix:*ConfustionMatrix* , Available: https://www.harrsisgeospatial.com/docs/ENVIConfustionMatrix_ConfustionMatrix.html.

Wikipedia, Wikimedia Foundation, last accessed (2020) *Confusion Matrix*.: June 2020. Available: https://en.wikipedia.org/wiki/Confusion_matrix.

Nitish Srivastava, Geoffrey Hinton et al. (2014). 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting'. *Journal of Machine Learning Research 15*

A. C. Sparavigna (*2020*) 'On the Use of ImageJ Segmentation,' *ResearchGate, 12 January 2020*.

Esteva, Andre, et al. (2017). 'Dermatologist-level classification of skin cancer with deep neural networks,' *Nature* 542.7639: 115-118.

Codella, Noel et al. (2017) 'Deep learning ensembles for melanoma recognition in dermoscopy images,' *IBM Journal of Research and Development* 61.4/5: 5-1.

Liao and Haofu (2016) 'A deep learning approaches to universal skin disease classification.' *University of Rochester Department of Computer Science, CSC*(2016)

Dorj, Ulzii-Orshikh et al. (2018). 'The skin cancer classification using deep convolutional neural networks,' *Multimedia Tools and Applications* 77.8: 9909-9924.

Rezvantalab, Amirreza et al. (2018). "Dermatologist level dermoscopy skin cancer classification using different deep learning convolutional neural networks algorithms." *arXiv preprint arXiv*:1810.10348.

Mohd Sanad and Zaki Rizvi (2020) *Learn Image Classification on 3 Datasets using Convolutional Neural Networks (CNN),* https://www.analyticsvidhya.com/blog/2020/02/learn-image-classification-cnn-convolutional-neural-networks-3-datasets/

Coursera (2018) *Normalizing inputs-deeplearning.ai* [online], Available: https://www.coursera.org/lecture/deep-neural-network/normalizinginputs.-lXv6U

Nikhil B (2019). *Image Data Pre-Processing for Neural Networks* [online], Available: https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258.

Andre et al. (2017) 'Dermatologist-level classification of skin cancer with deep neural networks,' *Nature* 542.7639: 115-118.

Noel CF et al. (2017). 'Deep learning ensembles for melanoma recognition in dermoscopy images,' *IBM Journal of Research and Development* 61.4/5: 5-1.

Ilya Sutskever, Ruslan Salakhutdinov et al, (2014). Overfitting: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 2014

Confusion Matrix: (2020) *Wikipedia*, Wikimedia Foundation, last accessed June 2020. Available: https://en.wikipedia.org/wiki/Confusion_matrix.

APPENDIXES

Appendix A: Sample source Codes

1. Image generator code:

```
train_datagen = ImageDataGenerator(
                        rotation_range=20,
                        width_shift_range=0.1,
                        height_shift_range=0.1,
                        shear_range=0.1,
                        zoom_range=0.2,
                        horizontal_flip=True,
                        vertical_flip=True,
                        rescale=1./255,
                        fill_mode='nearest',
                        dtype='float32'
                        )
train_datagen.fit(X_train)
val_datagen = ImageDataGenerator()
val_datagen.fit(X_test)
```

2. Code to print and plot the confusion matrix for true vs predicted labels

```
def plot_confusion_matrix(cm, classes,
                    name,
                    normalize=False,
                    title='Confusion matrix',
                    cmap=plt.cm.Reds):
   plt.figure(figsize=(8,6))
   plt.imshow(cm, interpolation='nearest', cmap=cmap)
   plt.title(name)
   plt.colorbar()
   tick_marks = np.arange(len(classes))
   plt.xticks(tick_marks, classes, rotation=0)
   plt.yticks(tick_marks, classes)
   if normalize:
      cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
   thresh = cm.max() / 2.
   for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
      plt.text(j, i, cm[i, j],
```

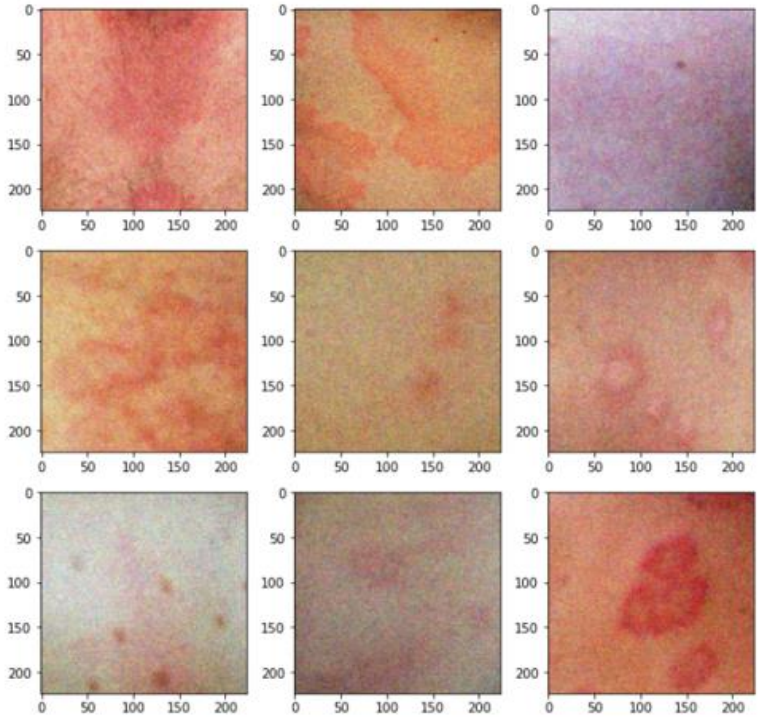
```
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    fig = plt
```

3. Code to plot learning curves: for Accuracy and Loss

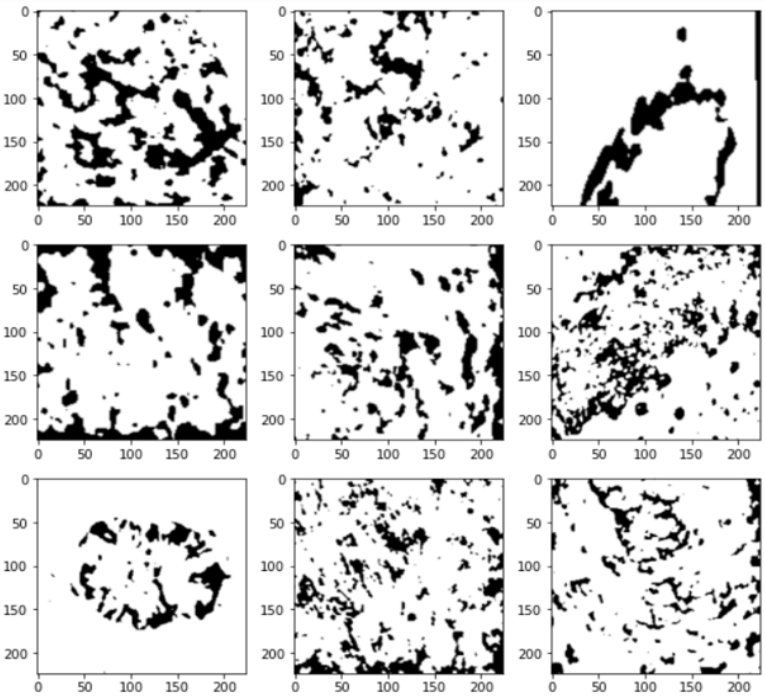
```
acc_train=history.history['accuracy']
acc_val=history.history['val_accuracy']
epochs=range(1,7)
plt.plot(epochs, acc_train, 'b', label='Train_acc')
plt.plot(epochs, acc_val, 'r', label='Val_acc')
plt.title('Training vs Validation Accuracy', fontsize=14)
plt.xlabel('Epoch',color='green',fontsize=14)
plt.ylabel('Accuracy',color='green',fontsize=14)
plt.grid(True)
plt.legend()
plt.show()
print("Training acuracy =    {:.2f}".format(history.history["accuracy"][-1]))
print("Validation accuracy = {:.2f}".format(history.history["val_accuracy"][-1]))

loss_train=history.history['loss']
acc_val=history.history['val_loss']
epochs=range(1,7)
plt.plot(epochs, acc_train, 'b', label='Train_loss')
plt.plot(epochs, acc_val, 'r', label='Val_loss')
plt.title('Training vs Validation Loss', fontsize=14)
plt.xlabel('Epoch',color='green',fontsize=14)
plt.ylabel('Loss',color='green',fontsize=14)
plt.grid(True)
plt.legend()
plt.show()
print("Training loss =   {:.2f}".format(history.history["loss"][-1]))
print("Validation loss = {:.2f}".format(history.history["val_loss"][-1]))
```

Appendix B: Sample skin disease images

1. Sample original images



a. Sample segmented images



Appendix C: Model summary

1. The Skin Disease Classification model summary

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_12 (Conv2D)          (None, 224, 224, 32)      896

 max_pooling2d_12 (MaxPooling2D)  (None, 112, 112, 32)  0

 conv2d_13 (Conv2D)          (None, 112, 112, 64)      18496

 max_pooling2d_13 (MaxPooling2D)  (None, 56, 56, 64)    0

 dropout_21 (Dropout)        (None, 56, 56, 64)        0

 conv2d_14 (Conv2D)          (None, 56, 56, 128)       73856

 max_pooling2d_14 (MaxPooling2D)  (None, 28, 28, 128)   0

 dropout_22 (Dropout)        (None, 28, 28, 128)       0

 flatten_4 (Flatten)         (None, 100352)            0

 dense_7 (Dense)             (None, 128)               12845184

 dropout_23 (Dropout)        (None, 128)               0

 dense_8 (Dense)             (None, 4)                 516

=================================================================
Total params: 12,938,948
Trainable params: 12,938,948
Non-trainable params: 0
```

2. The Transfer Learning VGG16 model summary

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_8 (InputLayer)        [(None, 224, 224, 3)]     0
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168
 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
 flatten (Flatten)           (None, 25088)             0
 fc1 (Dense)                 (None, 512)               12845568
 dropout_10 (Dropout)        (None, 512)               0
 fc2 (Dense)                 (None, 512)               262656
 dropout_11 (Dropout)        (None, 512)               0
 output (Dense)              (None, 4)                 2052
=================================================================
Total params: 27,824,964
Trainable params: 13,110,276
Non-trainable params: 14,714,688
_________________________________________________________________
```

3. The Finetuned VGG16 model summary

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_7 (InputLayer)        [(None, 224, 224, 3)]     0
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168
 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
 flatten (Flatten)           (None, 25088)             0
 fc1 (Dense)                 (None, 512)               12845568
 dropout_8 (Dropout)         (None, 512)               0
 fc2 (Dense)                 (None, 512)               262656
 dropout_9 (Dropout)         (None, 512)               0
 output (Dense)              (None, 4)                 2052
=================================================================
Total params: 27,824,964
Trainable params: 26,089,476
Non-trainable params: 1,735,488
_________________________________________________________________
```