TECHNOLOGICAL UNIVERSITY OF THE PHILIPPINES

College of Engineering

Electronics Engineering Department

CLASSIFICATION OF SKIN DISEASE THROUGH DEEP LEARNING VIA ANDROID SMARTPHONE

By:

ALBERIO, JEAN WILMAR G.

APUANG, JONATHAN G.

CRUZ, JOHN STEPHEN B.

GOMEZ, MARK ANGELO B.

MOLINA, BENJAMIN JR., V.

TUALA, LYNDON T.

MARCH 2019

CLASSIFICATION OF SKIN DISEASE THROUGH DEEP LEARNING VIA ANDROID SMARTPHONE

A Project Study Presented to the Faculty of

Electronics Engineering Department

College of Engineering

Technological University of the Philippines

In Partial Fulfilment of the Course Requirements for the Degree of

Bachelor of Science in Electronics Engineering

Submitted by:

Alberio, Jean Wilmar G.

Apuang, Jonathan G.

Cruz, John Stephen B.

Gomez, Mark Angelo B.

Molina, Benjamin Jr., V.

Tuala, Lyndon T.

Adviser:

Engr. Jessica S. Velasco

March 2019

APPROVAL SHEET

This project study entitled "**Classification of Skin Disease through Deep Learning via Android Smartphone**", has been prepared and submitted by the following proponents:

Alberio, Jean Wilmar G	Gomez, Mark Angelo B.
Apuang, Jonathan G.	Molina, Benjamin Jr., V.
Cruz, John Stephen B.	Tuala, Lyndon T.

In partial fulfillment of the requirements for the degree of **Bachelor of Science in Electronics Engineering** is hereby recommended for approval.

_____
ENGR. JESSICA S. VELASCO
Project Adviser

_____	_____
ENGR. JOHN CARLO V. PUNO	**ENGR. MARIA VICTORIA C. PADILLA**
Panel Member	Panel Member

_____
ENGR. NILO M. ARAGO
Panel Member

Accepted and approved in partial fulfillment of the course requirements for the degree of **Bachelor of Science in Electronics Engineering.**

_____	_____
ENGR. LEAN KARLO S. TOLENTINO	**ENGR. BENEDICTO N. FORTALEZA**
Head, ECE Department	Dean, College of Engineering

ACKNOWLEDGMENT

The proponents would like to express their gratitude first of all, to the Almighty God for giving them strength, good health and wisdom that enabled them to complete this research study without accidents occurred upon them.

To their adviser, Engr. Jessica Velasco, for her ideas and guidance in making this project study happen. To their subject advisers, Engr. August Thio-ac and Engr. Nilo Arago, for leading them the correct way of creating a study that will not only help the students in performing research but also for the benefit of their school's pride in research.

To their family for the continuing support they need in accomplishing this study specially with the financial matters involve.

To their friends for the encouragements that give them the energy to continue their works with enthusiasm.

To the humble employees of the Department of Health (DOH) for some data that showed to them.

To the hardworking doctors of the Department of Dermatologist in Ospital ng Maynila who helped on their research by validating the data they gathered.

To the honorable police officer that allowed them to gather additional data they need inside Camp Crame.

And also, to the hospitable employees of Sucat Health Center in Muntinlupa City that allowed them to deploy their research to the community in the stated barangay.

The proponents are very grateful for the support of everyone that helped in accomplishing this research study because without their support, this research may not be possible. The members thank you all. God bless us all.

ABSTRACT

Skin diseases are prevalent and recurring in the Philippines. Cases involving these diseases are often ignored and thus lead to a worse condition. This study will aid skin professionals in diagnosing skin diseases. The proponents gathered 3,406 images composed of 7 types of skin disease that was used for training and testing of different convolutional network models. The proponents used transfer learning of pretrained convolutional neural networks which include VGG16, VGG19, MobileNet, ResNet50, InceptionV3, Inception-ResNetV2, Xception, DenseNet121, DenseNet169, DenseNet201 and NASNet mobile. The skin disease classification was successfully implemented on these models. The study made the following evaluations based on confusion matrix, loading time, accuracy and weight size. Through these evaluation MobileNet achieved the highest accuracy which is 84.28% also having a lightweight weight size of 16.823MB and a loading time of 4.838 seconds. After conducting the evaluation, the MobileNet model was further enhanced using data augmentation and different sampling methods then was deployed on the developed Android application. The model with preprocessing and data augmentation provide a 94.4% accuracy on the test dataset but when tested on field it yields 68.8%. While the model with the default preprocessing of input data and splitting the dataset with 80% on train data and 20% on test data generate a 93.6% accuracy but when tested on field it results to 81.2%. In conclusion, oversampling and data augmentation provide a better accuracy on test dataset but when the classification system is tested on real-world application, the accuracy suffers.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

THE PROBLEM AND ITS BACKGROUND

1.1 Introduction

The Philippines being a tropical country has a tropical marine climate is divided into two main seasons: a rainy and a dry season. It has an average temperature of 27-33 degree Celsius as well as a relative humidity of 77-83%. These environmental factors and the coupled with age, occupation, and immune responsiveness contribute to the high prevalence of skin diseases among Filipinos.

Skin diseases are characterized as disorders that often begin inside the body or start from the skin, and outwardly show on the skin (Dayan, N., 2008). Some of them are extremely uncommon, however, others are commonly occurring. They bring the person itch, and pain, as well as emotional and social impacts because of its visibility. All things considered, dermatologists guaranteed that a large portion of the skin diseases are manageable with legitimate medications, if they are decisively diagnosed. Thus, an effective automatic skin disease detection design, which can be used by the dermatologist to reduce their workload is highly anticipated.

In recent years when big data started to boost like around 2011, the Big Data is frequently denoted by 5Vs: the wide variety of data types, the extreme volume of data, speed at which the data must be processed, the variability of the data, and the value of data. Big Data is a term used for any assembly of massive data sets whose complexity, as well

as size, exceeds the capacity of conventional data processing applications. Clinical and epidemiology offers extraordinary research opportunities using Big Data to help create scientific advancements (Wehner, M. et al., 2017). Utilization of big data and image recognition technology along with the field of dermatology could yield remarkable aid to patients, dermatologist, as well as the research community. There are plenty of skin diseases which can be diagnosed by a dermatologist through ocular inspection. Utilization of artificial intelligence coupled with the underlying technology of deep learning for diagnosis is made reasonable by the fact that each of these conditions has a unique visual feature. Cases of skin diseases that are prevalent in the Philippines and potentially can be identified by image processing technologies are chicken pox, acne, eczema, pityriasis rosea, psoriasis, tinea corporis, and vitiligo.

Researchers propose to visually infer the aforementioned skin diseases by gathering external data obtained from professional and publicly accessible websites such as dermweb.com photo atlas and a different database from the Philippine Dermatological Society (PDS) accredited institutions and classify each image into the appropriate category via transfer learning models.

1.2 Background of the Study

Among other diseases that accounts for notable fatality (i.e. HIV/AIDS, pneumonia and tuberculosis), Skin diseases are, more often than not, ignored as they are considered “small-time players” in the league of illness. Skin Problems, most commonly in tropical areas, are often seen in primary care settings due to its nature of constantly occurring. It is

seen as well in some regions where infectious diseases are endemic. One report from the World Health Organization in 2001 was the correlation of skin diseases with mortality rates in Sub-Saharan Africa. The global burden of disease was comparable to mortality rates related to hepatitis B and meningitis in the same area.

According to the data published by the World Health Organization in 2014, Skin Disease mortality in the Philippines reached 2,761 or 0.53% of total deaths, ranking the country #37 in the world. In the year 2007, chickenpox had 23,000 recorded cases and became one of the leading causes of death that year.

Dermatology is one branch of medicine associated with skin, nails, hair, and diseases associated with it. Dermatologists are medical practitioners qualified in diagnosing and treating skin disorders. They recognize skin conditions or diseases that a person might have, simply by visual observation, then after some time, if it is verified, a possible cure will be given. Analyzing a skin disease through observation is time-consuming and can sometimes lead to a wrong diagnosis. Some skin diseases like chicken pox don't need any treatment and will just heal in a few days while some other skin diseases are in need of medication. Due to this, misdiagnosis is harmful in dealing with skin diseases.

However, skin diseases begin to be a huge problem. Treating these skin diseases properly at the earlier stages has become an important thing to prevent serious damage or worse, mortality. This study would help to resolve this problem. Since this study would

allow dermatologists to determine skin diseases faster by simply capturing the area of skin infected with their smartphone.

Artificial Intelligence or AI is a branch of computer science that aspires to simulate human intelligence by machines. The processes include knowledge, problem-solving, reasoning, planning, learning, and perception. AI today is known as narrow or weak AI because it is designed to execute simple tasks like facial recognition and games. The long-term goal is to develop an almost perfect AI which could outperform humans at nearly every task.

A core part of Artificial Intelligence is machine learning. It is a simple concept machine that takes data and learns from it. The ability to identify patterns with streams of inputs, classification, and numerical regressions are what defines machine learning. Machine perception and computer visions are two different parts of machine learning. The first one uses sensory inputs to infer different aspects while the latter is the power to analyze visual information such as an object, gesture and facial recognition.

Deep Learning is a subdivision of machine learning influenced by the function and structure of the brain's neural networks. Deep learning is achieving results that were not possible before like the technology behind driver-less cars and voice control in consumer devices. This is why this technology is getting lots of attention nowadays.

In deep learning, models can achieve cutting edge accuracy, sometimes exceeding human performance. It performs classification assignments directly from a text, sound or

images. Using a massive set of data and network architecture, models are trained to achieve such accuracy.

Convolutional Neural Network (CNN is a type of deep neural network concerned with combining input data and pre-trained features to process 2D data, like images. CNN works by extracting traits directly from images so there is no need for a human to identify it manually. After training on a group of images, the relevant features are learned, which makes deep learning highly accurate in classifying objects.

1.3 Statement of the Problem

Skin infections or diseases commonly caused by bacteria and fungi, for example, cellulitis, impetigo, boils, and leprosy are all caused by bacterial infections. A normal person and sometimes a professional can cause a mistake in identifying the skin disease actually distinguishable by the naked eye by the dermatologist. But some of the dermatologists can give some inconsistencies for each diagnosis and also it gives more time for them to study the skin disease that a person might have because the dermatologist doesn’t have the device, they rely on books and the internet.

One of the challenges about this study is there are insufficient data and information about different skin diseases prevalent in the Philippines. As of now, the Department of Health Database System has insufficient information for different skin diseases prevalent in the Philippines.

1.4 Objectives

1.4.1 General Objectives

To design a GUI in an android phone that will classify different skin diseases using different Convolutional Neural Networks models.

1.4.2 Specific Objectives

1. To gather and analyze images of different skin diseases like: Chicken Pox, Acne, Eczema, Pityriasis Rosea, Psoriasis, Tinea Corporis, Vitiligo.
2. To implement a classification system using transfer learning models of different convolutional Neural Network architecture.
3. To train and test the classification system using the CNN architecture.
4. To compare and evaluate the different CNN architecture using the different criteria: Splitting of the train and test data, confusion matrix, loading time, and the weight size.
5. To design a GUI that will get information about the patient and will give an output of the classified disease.
6. To deploy and implement the project to an accredited dermatological institution.

1.5 Significance of the Study

The project study aims to be easily accessible to a dermatologist. It also helps them classify skin diseases with ease of providing faster treatment.

This project easily identifies the kind of disease in the skin that a person might have by just simply capturing a sample image of the skin of the person. In this project, many dermatologists would benefit from it, increasing knowledge and skill in primary care by adaptive learning, they only need is an android device.

This project will benefit everyone, dermatologists and patients. Effectively using big data will yield significant results in the field of dermatology. Healthcare organization, large hospital networks as well as accountable care organizations will benefit from this. In addition, this project is a machine learning in which data obtained throughout its usage will help create a more accurate device, thus proving more useful than common ways to diagnose.

1.6 Scope and Delimitations of the Study

This study is used to determine types of skin disease by image-capture using an Android phone. It is a Python-based program. Scanning device portability. It can measure accuracy through a performance test. It focuses on determining the disease of the skin of the most prevalent skin disease by using a convenient and handy android phone by capturing the image of the skin which will be processed using a program generated from Python then later develops a report based on the results and findings.

Proponents limit this research study only for android phones (cannot run on ios). Not all types of skin disease can be determined by the application due to limited reliable sources of skin disease images. A high-quality camera used to accurately scan the disease.

In capturing the image of the skin, a certain height and angle of the android phone are properly observed to regulate the lighting for more accurate results.

1.7 Operational Definition of Terms

The following are the technical terms used in the discussion of this study. Definitions were provided for easy understanding on its content.

Convolutional Neural Network (CNN) - a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. CNNs use a variation of multilayer perceptron designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

Image Processing - is the application of signal processing techniques to the domain of images — two-dimensional signals such as photographs or video. Image processing does typically involve filtering or enhancing an image using various types of functions in addition to other techniques to extract information from the images.

Android Studio - is intended to be used by development teams as small as one person or as large as global teams. The Android Studio IDE can be linked to larger teams with GIT or similar version control services for larger teams. Mature Android developers will find tools that are necessary for large teams to deliver solutions rapidly to their customers. Android solutions can be developed using either Java or C++ in Android Studio.

Transfer Learning - is a machine learning method used in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks. It is where a model developed for a task is reused as the starting point for a model on the second task. In transfer learning, we first train a base network on a base dataset and task, and then we repurpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task. (Yosinki, et al.,2014) "How transferable are features in deep neural networks?"

Deep Learning - (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised. Deep learning models are loosely related to information processing and communication patterns in a biological nervous system, such as neural coding that attempts to define a relationship between various stimuli and associated neuronal responses in the brain.

Python Program - is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, and a syntax that allows programmers to express concepts in fewer lines of code, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Tinea Corporis - (also known as ringworm, tinea circinata, and tinea glabrosa) is a superficial fungal infection (dermatophytosis) of the arms and legs, especially on glabrous skin; however, it may occur on any part of the body. It is similar to other forms of tinea.

Pityriasis Rosea - is a type of skin rash. Classically, it begins with a single red and slightly scaly area known as a "herald patch". This is then followed, days to weeks later, by a pink whole-body rash. It typically lasts less than three months and goes away without treatment. Sometime a fever may occur before the start of the rash or itchiness may be present, but often there are few other symptoms.

Eczema or Atopic Dermatitis - is a reaction pattern that the skin produces. The most common causes of eczema are Atopic dermatitis which is more prevalent in those with hay fever and asthma. It is not a single health condition, but a reaction pattern seen in a number of skin diseases. It begins as red blisters containing a clear fluid on top of plaques.

Chicken pox - caused by the herpes varicella-zoster virus, chicken pox is a viral illness characterized by a very itchy red rash and is one of the most common diseases of childhood. Though chicken pox is highly contagious, people who have had chicken pox almost always develop lifetime immunity against it. However, the virus remains dormant in the body and it can reactivate later in life and cause shingles.

Acne - also known as acne vulgaris, is a long-term skin disease that occurs when hair follicles are clogged with dead skin cells and oil from the skin. It is characterized by blackheads or whiteheads, pimples, oily skin, and possible

scarring. It primarily affects areas of the skin with a relatively high number of oil glands, including the face, upper part of the chest, and back. The resulting appearance can lead to anxiety, reduced self-esteem and, in extreme cases, depression or thoughts of suicide.

Psoriasis - is a long-lasting autoimmune disease characterized by patches of abnormal skin. These skin patches are typically red, itchy, and scaly. Psoriasis varies in severity from small, localized patches to complete body coverage. Injury to the skin can trigger psoriatic skin changes at that spot, which is known as the Koebner phenomenon.

CHAPTER 2

REVIEW OF RELATED LITERATURE

This chapter presents research findings, theories and related studies from previous research related to the proposed study.

2.1 Conceptual Literature

2.1.1 Chicken pox

Chickenpox (Varicella) is caused by the varicella-zoster virus and is very contagious. It is most common in kids, but some adults get it too. It is also an airborne disease meaning, it can easily affect others through coughs and sneezes from an infected person. The infection has a skin rash with red blisters. Symptoms of chickenpox include fever, tiredness, and headaches. For several days, the blisters pop which eventually scabs over.

Figure 1. Sample pictures of Chickenpox

Chickenpox is one of the ten most significant causes of morbidity in the Philippines (2007) with over 23,000 cases affecting almost every region in the

country. It has a rate of 26.4 (rate per 100,000 population) for the male and 25.1 for the female and is also one of the top ten causes of morbidity of every region in the country. (FHSIS Annual Report, 2007)

In 2015, Chickenpox is no longer one of the top 10 leading causes of morbidity. The Cordillera Administrative Region (CAR), Region 4A (CALABARZON), Region 6, Region 7 and Region 10 still has Chickenpox as one of their notifiable diseases. (FHSIS Annual Report, 2015)

2.1.2 Acne

Acne, or acne vulgaris, is an inflammatory skin condition and a long-term skin disease that visually appear when hair follicles are obstructed with oil and dead skin cells. It is most commonly seen in the face, chest, back, shoulder, neck and upper arms. It is not dangerous, but it can lead to anxiety and reduced self-esteem because it may lead to scarring.

Figure 2. Sample images of people affected by acne

2.1.3 Eczema

Eczema (atopic dermatitis) is a skin condition which is rarely fatal and not a contagious condition, but the rash, dryness, inflammation, and itching of the skin may be uncomfortable for those who has it. It commonly occurs in infants. Symptoms vary according to age, but they often include scaly, itchy patches of skin. There is no cure for eczema, but there are numerous things that affected people can do to alleviate symptoms, e.g. Taking lukewarm baths and applying moisturizer every day.

Figure 3. Sample images of people with eczema

2.1.4 Pityriasis Rosea

Pityriasis Rosea is a skin disease that is characterized by patches and rashes on the skin. The rash disappears on its own even without treatment. Mild itching is a problem for the people who have it. Before the Mother patch (herald patch) appears, a person with it may feel tired as though having a cold, nausea, headache, sore throat and a loss of appetite. Although treatment isn't needed, antiviral medicines such as acyclovir may help shorten the time you have the rash.

Figure 4. Sample images of people with Pityriasis Rosea

2.1.5 Psoriasis

Psoriasis is characterized by plaques of scaling skin. It has thickened red patches that are itch and painful caused by extra skin cells. The symptoms differ for each individual who has it. The most common ones are red patches of skin, cracked skin that bleed, soreness and itching, as well as swollen and stiff joints. Psoriasis is not currently curable and ongoing research is making progress on finding better treatments.

Figure 5. Sample images of people with Pityriasis Rosea

2.1.6 Tinea Corporis

Tinea Corporis or ringworm is a fungal infection caused by dermatophyte that affects the legs and arms as well as on bare skin e.g. skin regions except for the scalp, palms, soles, and groin. It is common to acquire it with animals. Direct skin contacts with someone who has it is also one of the causes. Someone with a weakened immune system are at high risk of being infected. Tinea Corporis can be treated by applying antifungal creams, but if the condition worsen, oral medication is required.

Figure 6. Sample images of Tinea Corporis

2.1.7 Vitiligo

Vitiligo is characterized by white patches of skin appearing either localized in one area, or on several different parts of the body. When the production of melanin stops functioning, vitiligo emanates. The condition is not fatal nor contagious, but it can affect self-esteem and makes you feel bad about you.

Figure 7. Sample images of Vitiligo

2.1.8 Digital Image Processing

The eyes are the most advanced sensory organ humans have. We are able to differentiate images from each other. The difference between humans and imaging machines is that a machine covers the entire EM spectrum, while the prior is limited to the visual band of the same spectrum. Imaging machines can recognize ultrasound as well as computer-generated images. Digital image processing deals with processing digital images using a digital computer. A digital image is constructed with elements called image elements, a pixel, pel and picture elements. (Gonzales, R., 2008)

2.1.9 Artificial Neural Networks

ANN's are computational algorithms intended to simulate the brain's neural network. It works the same way the human brain manages information. It is capable of machine learning and pattern recognition. Artificial Neural Network contains 3 layers called Input, Hidden and the output layer interconnected with each other. (Sharma, 2017)

2.2 Big Data

In the study “Big data analytics of IoT based Health care monitoring system”, (Dineshkumar, et al.,2016) the researchers’ advancement towards health monitoring system was generated by augmenting Big data and the Internet of Things (IoT). Physicians could not handle huge volumes of data that the IoT based health care monitoring system produce. The decision needed to make was about the patient’s health for distinction on these huge volumes of information. Thus, exercising Big Data is the way to process large volumes of data. Using hadoop’s capabilities of efficiently storing and processing this data will pave the way to how healthcare data will improve patient care at reduced costs.

2.3 Transfer Learning

Humans recognize and apply knowledge from past experiences when given a related task. The more it is connected, the faster humans can master it. Machine learning algorithms, the common ones, usually deals with simple tasks. This is what transfer learning tries to change. A more advanced algorithm called transfer learning wherein the transfer of knowledge from one or more related task is used to improve on a new task. (Torrey and Shavlik)

2.3.1 Inception V3

In the research “Inception-v3 for flower classification” (Xia, et al.) the researchers used the Inception-v3 model of the Tensor Flow platform to classify flowers. Transfer learning technology was used to obtain a boost in accuracy of flower classification. They retrain the dataset for the flower category, which

produced 95% accuracy for the Oxford-17 flower dataset proving higher than other methods.

2.3.2 MobileNet

In the study “Driver distraction detection using single convolutional neural network” (Kim, et al.,2017), the researchers used a Convolutional Neural Network model like Mobilenet to detect driver distraction. Though the system results varied greatly on how fast CPU/GPU processing time is, the results for MobileNet accuracy rather than Inception ResNet is observed to be higher.

2.3.3 Keras platform

In the study “Fully convolutional networks for segmenting images from an embedded camera” (Vinhal, et al.,2017), Keras, together with Python and Theano was used in implementing and creating a prototype for the Fully Convolutional Network (FCN). This paper Used an FCN and the keras library to segment images and provide low-level computer vision. To prototype the architecture, the proponents used keras library to speed up the process of searching an accurate network. The training as well as the validation was taken from the dataset that the robot’s imaging-acquisition collected.

2.3.4 Inception-ResNet-V2

In the study “PolyNet: A Pursuit of Structural Diversity in Very Deep Networks”, (Zhang, et al.,2017), the researchers presented PolyInception, a new family of modules that can be inserted as replacements on the different section of a

network. PolyInception modules along with the architectural efficiency improves the power to computational cost ratio.

2.3.5 VGG-16

VGG-16 has 3 fully connected layers and 13 convolutional layers. One fully connected layer contains 1000 channels and the remaining two have 4096 channels each. Like AlexNet, VGG-16 has the same procedure except for the sampling of cropped inputs. (Kumar, et al., 2017) “A convolutional neural network for visual object recognition in marine sector”

2.3.6 VGG-19

In the study “360° view camera-based visual assistive technology for contextual scene information”, (Ali, et al.,2017), the researchers proposed A system using a convolutional Neural network to assist the visually impaired. A 360° view camera provides a contextual information through the form of audio. The researchers used a pre-trained VGG-19 network, a neural network with 19-layers, to achieve this. The network consists of convolutional layers, fully connected layers, maxpooling and an output softmax layer. These layers are sequential and are placed in a stack form.

2.3.7 AlexNet

The researchers used an improved Alexnet model since the old one is limited in image classification. Due to the large convolutional kernel, excessive compression of information was its main problem. The upgraded AlexNet trains

continuously, avoiding the problems encountered on the older one. The design still follows the principle of neural networks.

2.4 Local Related Studies

In the study "Urine Sediment Classification using Deep Learning," (Velasco, et al. 2017), the researchers Used microscopic images of urine sediments to test different convolutional networks. They used the models VGG-16, VGG-19, Xception, Resnet 50, Inception-ResnetV2, InceptionV3, and MobileNet. The study made yielded 99.4% true positives for the inception V3 and inception ResNet V2. Mobilenet had a comparable result with 98% accuracy. The output of the study contains the Confusion Matrix, feature extraction, loading time, f-scoring, recall, precision and accuracy.

"Mobile-based Medical Assistance for Diagnosing Different Types of Skin Diseases Using Case-based Reasoning with Image Processing," (Aruta, et al.,2015) confirms that developing a case-based reasoning (CBR) system is much easier compared to rule-based reasoning (RBR) in developing a computer-based diagnosis system in the medical domain. The researchers presented a mobile-based medical assistance using image processing and CBR to diagnose skin diseases. The methodology that is used in the study is CBR, used for setting up a new knowledge base and image processing technique for determining the symptoms based on a newly captured picture of the person's skin concerning his/her skin problem.

In the study "Design and Evaluation of a Multi-model, Multi-level Artificial Neural Network for Eczema Skin Lesion Detection," (De Guzman, et al., 2015) the researchers

implemented an Artificial Neural Network ANN-based single level system as well as a multi-model, multi-level system for eczema detection.

"A Primary Morphological Classifier for Skin Lesion Images" (Macatangay, et al., n.d.), is a study using k-Nearest Neighbors, Multilayer Perceptron, Decision Trees, and Support Vector Machines to classify skin lesion images into primary morphologies. K-nearest neighbor(kNN) is an algorithm that compares a given test sample described by a number of attributes to samples with similar attributes. After comparison, the Support Vector Machines (SVM) performs the best in classification problems concerning skin lesion malignancy.

In this paper, "Malignancy Detection of Candidate for Basal Cell Carcinoma Using Image Processing and Artificial Neural Network" (Sybingco, et al., 2007) proposed a system using artificial neural network as well as digital image processing in detecting BCC disease. The detection is based on special characteristics of basal cell carcinoma. The system will be capable to correctly identify the occurrence of carcinoma using the proper threshold values with percent reliability of 93.33%

In this study, "A Model for Classification of Skin Lesions using Image Processing Techniques and Neural Network" (Abu, et al., 2017), the proponents made use of the Sobel operator for the segmentation process. Three different skin diseases are selected: Psoriasis, Seborrheic Keratosis, and Pyoderma. Two feature sets were experimented on with one feature set having 86 color and texture features and the other having 4,182 color and texture features. The results are promising given that the average F-measure using 86 features was 88.67% while the average F-measure using 4,182 features was 84.81%.

2.5 Foreign Related Studies

In the study “Multi-classification of skin diseases for dermoscopy images using deep learning,” by Zhou, et. al. in 2017, the researchers used a novel multi-classification method based on the convolutional neural network (CNN) in detecting skin diseases. The designed network is trained through transfer learning. 6 kinds of diseases are classified with the trained network including nevus, psoriasis, seborrheic keratosis, seborrheic dermatitis, eczema, and basal cell carcinoma. The experiments yielded 65.8% on six-classification task and 90% on two-classification tasks.

According to the study “SKINcure: An Innovative smartphone-based application to assist in melanoma early detection and prevention,” (Abuzaghleh, et al., n.d.), the survival rates of patients depends on the stage of infection and early detection for a higher chance of cure. Melanoma’s diagnosis is challenging since the processes are prone to misdiagnosis and inaccuracies due to the doctor’s subjectivity. The study proposes a smartphone-based application for the iPhone using Dermoscopy Image Analysis that analyzes the dermoscopy skin images of users to provide instantaneous results.

“Digital dermatology: Skin disease detection model using image processing,” by Ajith, et al. in 2017, is a study in which the researchers used the three transforms, Singular Value Decomposition, Discrete Wavelet Transform, and Discrete Cosine Transform in their device to yield higher accuracy. They proposed a skin disease detection method and states the benefit of image processing techniques on this field.

In the study “The melanoma skin cancer detection and classification using support vector machine,” by Alquran, et al. in 2017, the researchers used support vector machine (SVM) to classify melanoma skin cancer. They collected dermoscopy image database, segmented it using thresholding, collected unique characteristics, calculated total dermoscopy score and then classified it using SVM. The accuracy they got was 92.1%.

In the paper “Severity Grading of Psoriatic Plaques using Deep CNN based Multi-Task Learning,” by Chaturvedi, et al. in 2016, the researchers presented new multi-task learning where three classification tasks were interdependent, and the neural net was trained accordingly. The study addressed the problem of automatic analysis-based severity scoring of psoriasis. In the severity grading, scaling, erythema and induration were considered and the task is to predict severity scores for each parameter. Experimental results show that MTL and STL both achieve good performances.

In the study “Computer-aided Diagnosis of Four Common Cutaneous Diseases Using Deep Learning Algorithm” by Zhang, et al. in 2017, the researchers concluded that deep learning algorithms are viable for diagnosing skin diseases. The aim of the study was to apply deep neural network algorithm in classification of four common skin diseases. The researchers developed the algorithm from GoogleNet Inception V3 package. They adjusted the final layer to add their own datasets using transfer learning. It had promising results having 86.54±3.63% accuracy using the first dataset and 85±4.649% for the second dataset.

In the study “Low-Quality Dermal Image Classification Using Transfer Learning”, (Elmahdy, et. al., n.d.), the proponents used used a modified AlexNet to investigate skin lesion classification problem. They had a small size dataset, so they needed to modify the

deep CNN by replacing the decision layer, adding two more layers and fine-tuning of the complete network based on stochastic gradient descent. The researchers also augemented the original dataset by flipping the images three times resulting in an increased dataset size. The performance of classifiers was comparable to the Local Binary Pattern based system having 98.6% accuracy.

Table 1. Summary of Related Studies and Literature on Skin Disease Detection

TITLE	AUTHOR/S	YEAR	REMARKS
Urine Sediment Classification using Deep Learning	Velasco, et al.	2017	The researchers obtained a 99.4% using Inception V3 and Inception-Resnet V2. MobileNet also had comparable results with 98% accuracy.
Mobile-based Medical Assistance for Diagnosing Different Types of Skin Diseases Using Case-based Reasoning with Image Processing	Aruta, et al.	2015	Considering the application's accuracy, it can overcome more than 90% accuracy for the users. The methodologies used-CBR and Image Processing-made the detection rates much more accurate than using RBR or without image processing.
Design and Evaluation of a	De Guzman, et al.	2015	This research presents the design and evaluation of a system that

Multi-model, Multi-level Artificial Neural Network for Eczema Skin Lesion Detection			implements a multi-model, multi-level system using Artificial Neural Network (ANN) architecture for Eczema detection.
A Primary Morphological Classifier for Skin Lesion Images	Macatangay, et al.	n.d.	This research k-Nearest Neighbor, Decision Trees, Multilayer Perceptron, and support vector machines in classifying skin lesion images into primary morphologies.
Malignancy Detection of Candidate for Basal Cell Carcinoma Using Image Processing and Artificial Neural Network	Sybingco, et al.	2007	This research made use of numerical values of different features such as invasion, palisading, retraction artifact, and the shape and solidity of the individual cells to detect basal cell carcinoma with percent reliability of 93%.
A Model for Classification of Skin Lesions using	Abu, et al.	2017	The researchers use, Sobel Operator, GLCM (Grey Level Co-occurrence Matrix) Texture

Image Processing Techniques and Neural Network			Features, and RGB histogram feature resulted to the average F-measure using 86 features was, 88.67%, considerably high performance compared to 4,182 features (88.4%).
Multi-classification of skin diseases for dermoscopy images using deep learning	Zhou, et. Al	2017	This study obtained an accuracy of 90% on two-classification task and 65.8% on six-classification task using convolutional neural network.
SKINcure: An Innovative smart phone-based application to assist in melanoma early detection and prevention	Abuzaghleh, et al.	n.d.	This research proposes an innovative and fully functional smart-phone based application to analyze dermoscopy skin images for melanoma early detection.
Digital dermatology: Skin	Ajith, et al.	2017	This paper yielded a higher accuracy using three transforms,

disease detection model using image processing			Discrete Cosine Transform, Discrete Wavelet Transform, and Singular Value Transform.
The melanoma skin cancer detection and classification using support vector machine	Alquran, et al.	2017	The researchers used Support Vector Machine in detecting melanoma skin cancer and achieved an accuracy of 92.1%
Severity Grading of Psoriatic Plaques using Deep CNN based Multi-Task Learning	Chaturvedi, et al.	2016	The researchers address the problem of automatic machine analysis-based severity scoring of psoriasis skin disease.

Computer-aided Diagnosis of Four Common Cutaneous Diseases Using Deep Learning Algorithm	Zhang, et al.	2017	The researchers applied a deep neural network algorithm to classify dermoscopic images into four common skin diseases.
Low Quality Dermal Image Classification Using Transfer Learning	Elmahdy, et. al.	n.d.	The proponents used transfer learning using AlexNet deep Convolutional Neural Network (CNN) to investigate three class skin lesion classification problem of a low quality and small size dataset.

CHAPTER 3

METHODOLOGY

This chapter provides the procedures and methods used in the development and application of the project. This includes design flow, program algorithms, and testing procedures used to make the system.

3.1 Research Design

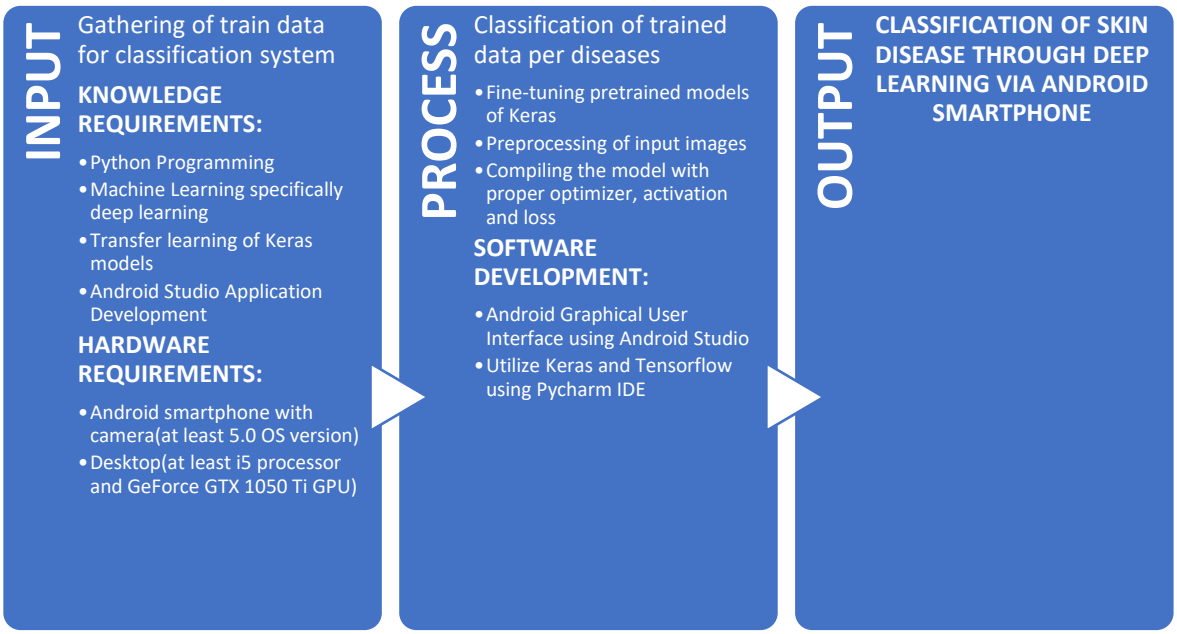


Figure 8. Input – Process – Output (IPO) diagram for the system

Figure 8 shows the Input – Process – Output diagram of classification of skin disease through Deep Learning via Android Smartphone. In the input, the system requires the following Android phone with camera and desktop with at least i5 Processor with GPU; since GPU provide a parallel computing capability to lessen the required time to train the model with the skin disease dataset. These input devices are used to develop the

classification system. Apart from it, the researchers are required to have the knowledge of deep learning using Python programming. For the process block, knowledge of fine-tuning of pretrained Keras model, different preprocessing of input data image methods and compiling model with proper optimizer, activation and loss is needed. The Android application system uses its camera to acquire the skin lesion and undergo deep learning to provide diagnosis. Once analyze, the output of the detected skin disease will be displayed. The project study has the following stages:

- Conceptualization of the Design
- Gathering of the Related Facts/Information
- Design of the Overall System

3.1.1 Conceptualization of the Design

The developed system identifies different skin diseases that is prevalent in the Philippines, specifically chicken pox, acne, eczema, Pityriasis rosea, psoriasis, Tinea corporis and vitiligo. It uses classification system using transfer learning models of different Convolutional Neural Network architecture available in Keras.

In determining the said skin diseases, the trained model choice based on the provided criteria which was trained using the computer with GPU will analyze the affected skin lesion captured using Android smartphone. The result will be displayed on the Android application after the model was done classifying the input image of the skin lesion.

3.1.2 Gathering of Related Facts/Information

- Research for local and international studies related to detection of different skin diseases.
- Interview with related institutions in which necessary queries can be answered reliably. The institutions are accredited from Philippine Dermatological Society (PDS), UP-PGH Section of Dermatology, Ospital ng Maynila Medical Center (OMMC) Department of Dermatology, Jose R. Reyes Memorial Medical Center - Department of Dermatology.
- Research the process of detection of skin diseases through different image processing techniques.
- Research for methods of detecting skin diseases.
- Research the proper procedures of diagnosing and treating skin diseases.
- Research for application development and interface with Android smartphone.
- Research about image classification using deep learning and transfer learning of convolutional neural network.

3.1.3 Design of the Overall System

The development of an Android smartphone application that can identify skin diseases using deep learning through Convolutional Neural Network has three main process. These are capturing the skin lesion, deep learning analysis, and displaying the result, as shown in Figure 9.

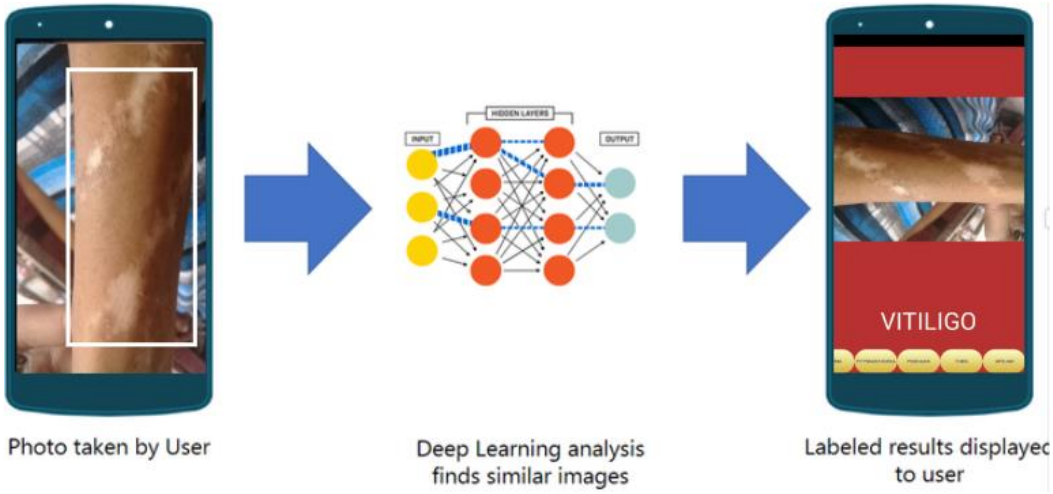


Figure 9. The Overall Conceptual Design of the System

3.2 Project Development

3.2.1 Dataset

The dataset comes from a combination of public accessible dermatology repositories, color photo atlas of dermatology and taken manually. The images gathered from online public access dermatology repositories are validated by dermatologist. Data images gathered consist of acne, eczema, Pityriasis rosea, psoriasis, Tinea corporis, varicella(chickenpox) and vitiligo.

3.2.2 Data Preparation

The dataset contains sets of images in jpeg extension which some were taken manually corresponding to the same skin lesion but from multiple viewpoint or multiple set of images acquired on the same person. Images were carefully selected and should be in colored and larger than 224x224 in size. Figure 10 show some example of images for each 7-skin disease.

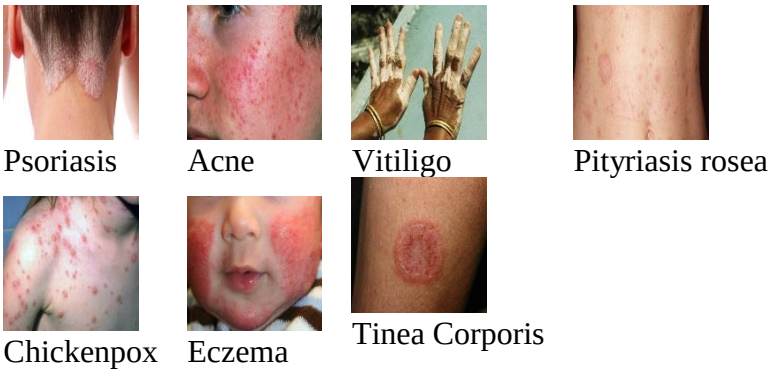


Figure 10. Sample Images of Dataset

3.2.3 Sample Selection

The dataset is partition to train and test data for each category of skin disease. The train data consist of 80% of the dataset while the test data is 20%. The validation data was gathered either in the train data or test data with the same number of images compared to test data as shown in Table 2.

Table 2. Dataset

Skin disease	Number of Images	Train Data	Test Data	Validation Data
Acne	805	644	161	161
Eczema	599	479	120	120
Chickenpox	314	251	63	63
Pityriasis rosea	347	278	69	69
Psoriasis	633	506	127	127
Tinea Corporis	385	308	77	77
Vitiligo	323	258	65	65
Total	**3,406**	**2,724**	**682**	**682**

3.2.4 Transfer Learning of Different Models

The program will be implemented using Pycharm IDE. Keras platform will be used with Tensorflow backend in coding the system. The system can detect 7 different skin diseases specifically: acne, eczema, psoriasis, vitiligo, Tinea corporis, chickenpox, and Pityriasis rosea. This is done by using transfer learning models of convolutional neural network, such as, VGG 16, VGG 19, Inception, Xception, ResNet50, DenseNet and Mobilenet.

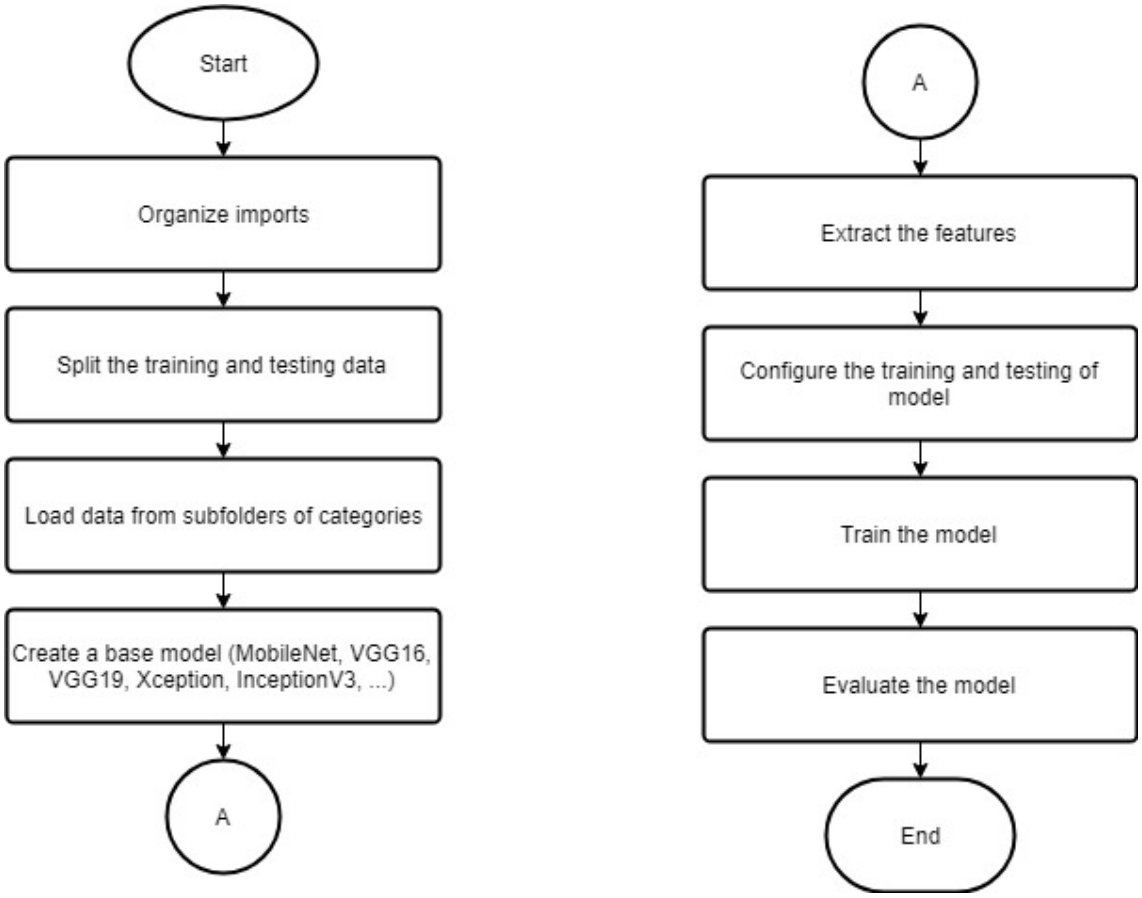


Figure 11. Program flowchart of the Python Code

The program starts at organizing imports such as numpy, keras, scikit-learn and matplotlib. Then, split the training and testing data by configuring the dataset into specific directories (training, testing and validation). Third is to load skin disease images from subfolders of category. Next, create a base model of different pretrained convolutional neural networks. Then, acquire the features by preprocessing the data. Keras has utilities to take care of this automatically. Next is to configure the training and testing of the model. Lastly, after training, different architectures will be evaluated and compared based on model accuracy, confusion matrix, loading time, and weight size to verify what is the best architecture for the skin disease classification.

Using pretrained convolutional networks, size of the input image differs for each model. The input image is equal to the size of the image (width and height) and the number of channels. Table 3 shows the fixed size of the input image for each model.

Table 3. Input for each model

Model	Input Image
MobileNet	224x224x3
VGG16	224x224x3
VGG19	224x224x3
Xception	299x299x3
ResNet50	224x224x3
InceptionV3	299x299x3
InceptionResNetV2	299x299x3
DenseNet121	224x224x3
DenseNet169	224x224x3
DenseNet201	224x224x3
NASNet Mobile	224x224x3

3.2.5 Training Algorithm

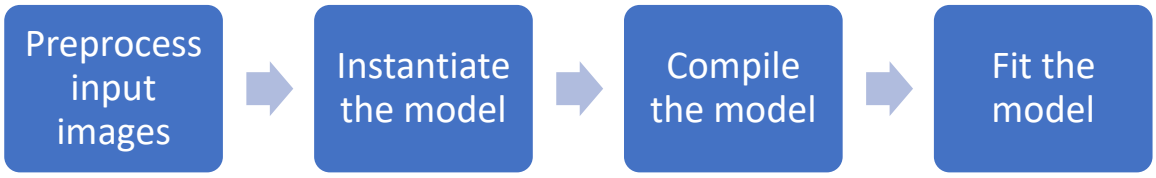


Figure 12. Training algorithm of the models

The training of the models starts from preprocessing the input images using the preprocessing function provided by Keras by following the input size required from each model shown in Table 2. Then, instantiate the model where the final classification layer or the last layers of each model was removed, and the other remaining layer was freeze. After removing the last layers, new layers were introduced like global average pooling and dense layer to modify the model to our classification system, this process is called as fine-tuning the model.

3.2.5.1 Loss

In order to manipulate the model or the neural network, there should be a measurement to be done on how far the expected output to the result. The loss of the neural network provides the computation on the distance of the true target of the neural network to the predictions made by your model. Figure 13 shows the overview concept of the loss function.

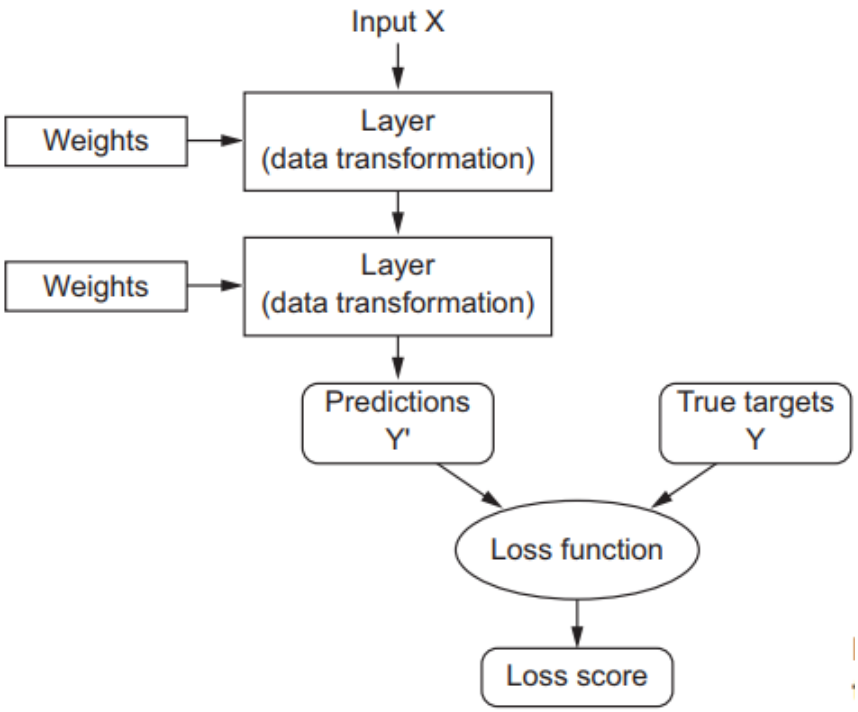


Figure 13. Loss function in deep learning

3.2.5.2 Optimizer

In every deep learning, the initial weights of the model compose of random values. The after a one epoch or iteration the model will compute for the loss score in order to minimize the value of the loss score the optimizer is introduced. Optimizer update the weights of the model in order to lessen the loss score gradually directing the model to the correct direction, this is called the backpropagation algorithm. Figure 14 visualize how the role of optimizer in a deep learning model.

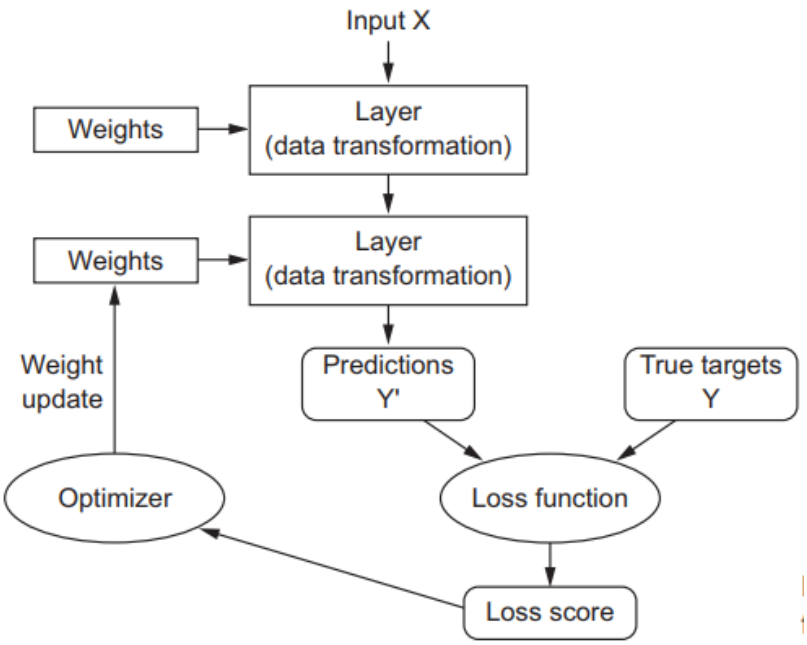


Figure 14. Optimizer function in deep learning

To compile the model the following parameters was used: the learning rate is 0.0001, the activation is softmax, the loss is categorical crossentropy, and the optimizer is Adam. Then, fitting the model with 30 epochs.

3.2.6 Inference Algorithm

After training the model, the model weight and architecture can be saved using the saved_model function of Keras with .h5 file extension. To properly deploy the CNN model to an Android application the .h5 file should be converted to a protobuff file with a file extension of .pb. Tensorflow is needed in this process where you freeze the graph producing a .pb file and a txt file containing the labels of the created model. To fully use the model for inference, a Tensorflow utility is used which is the optimize_for_inference function and with this the model can be successfully loaded to your Android application. Figure 15 shows the block diagram of necessary steps in deploying the model.

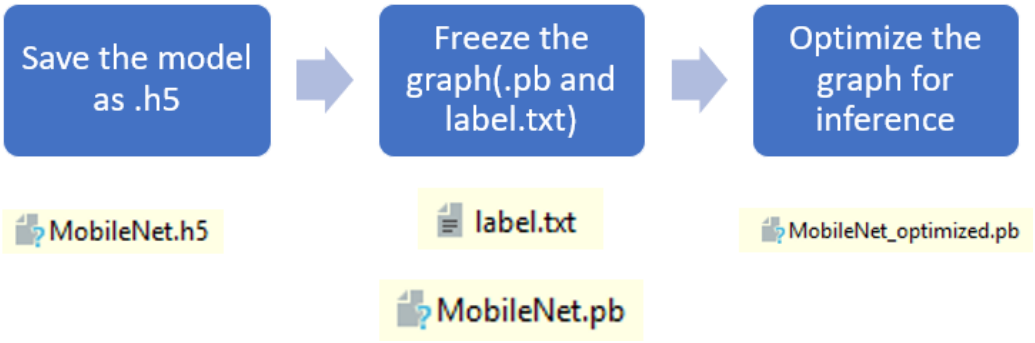


Figure 15. Deploying the model

3.2.7 Android Application

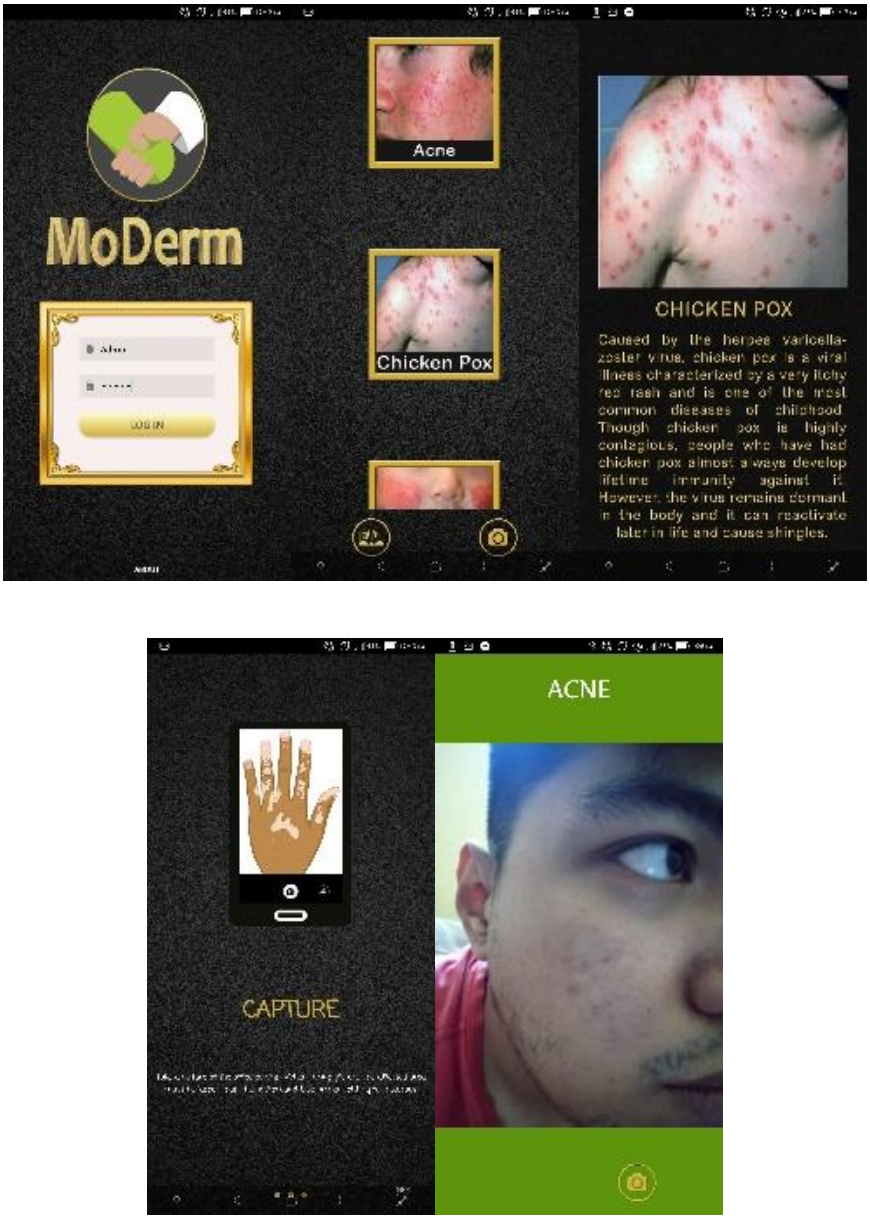


Figure 16. Shows the GUI of the Android application

3.2.8 MODERM Logo

Figure 17. Shows the logo of the MODERM application

3.2.9 Hardware

3.2.9.1 Samsung

Figure 18. Samsung Galaxy J2 Prime

Figure 18 shows the android smartphone used for the study. It has an android operating system version 6.0 (Marshmallow), 1.5GB RAM, 8 MP main camera and 5 MP front camera with autofocus and LED flash.

3.2.9.2 Asus

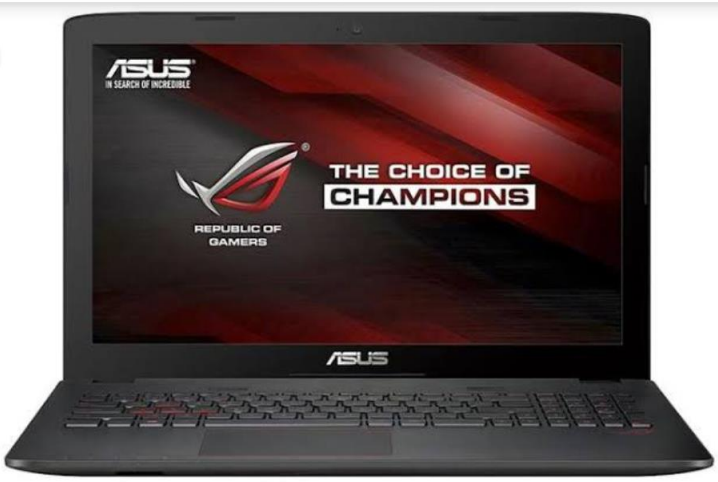


Figure 19. Asus ROG GL552VW

Figure 19 shows the laptop used to train using Pycharm. It has CPU Intel Core i7-6700HQ 32, GPU NVIDIA GeForce GTX 960M (2GB GDDR5) 55, and a 16 GB DDR4 RAM.

3.3 Evaluation Criteria of Different Pretrained Convolutional Neural Network

3.3.1 Confusion Matrix

Confusion matrix acts as the truth table of a classifier, basically different evaluation metrics can be derived by this alone to fully understand the model performance. The model is performing good if the diagonals yield the greatest number of frequencies in the table.

3.3.2 Accuracy

Accuracy is evaluated as the total number of correct predictions of the classifier over the test dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} = \frac{TP + TN}{P + N} \quad (\text{Eq. } 1)$$

Where

TP = true positives

FP = false positives

TN = true negatives

FN = false negative

3.3.2.1 Precision

To evaluate the precision of the classifier, the total number of the correct predictions made over the total number of the positive predictions made. This evaluation metric is also called as the positive predictive value of the classifier act as the value of the exactness of the model.

$$Precision = \frac{TP}{TP + FP} \quad (\text{Eq. } 2)$$

Where

TP = true positives

FP = false positives

3.3.2.2 Recall

Recall is defined as the total number of the correct positive predictions made by the classifier over the total number of true positives and false negatives, act as the completeness measurement of the classifier.

$$REC = \frac{TP}{TP + FN} = \frac{TP}{P} \quad (\text{Eq. } 3)$$

Where

TP = true positives

FN = false negative

P = total number of positives

3.3.2.3 F1-score

F1-score is evaluated as the average of recall and precision, where it is a metric that show the classifier balance between precision and recall.

$$F1 = \frac{2x(PREC * REC)}{PREC + REC} \quad \text{(Eq. 4)}$$

3.3.3 Weight size

After training the model. The model is then saved with the saved-model function under Keras. This function saved the model architecture as well as the weights of the model after training. The file produced is in .h5 extension, this file is the weight size of the model.

3.3.4 Loading Time

Saving the model provide a way to just load the model to deploy it. In order to compute for the loading time of each model, load-model function was used to load the model and calculate the time it takes.

$$Loading\ Time = Start\ Time - End\ Time \quad \text{(Eq. 5)}$$

3.4 Gantt Chart

Table 4. Gantt chart

TOPIC/TASK	December 2017	January 2018	February 2018	March 2018	April 2018	May 2018	June 2018	July 2018	August 2018	September 2018	October 2018	November 2018	December 2018	January 2019	February 2019	March 2019
Conceptualization of Topic for Project Study	■															
Research of Problem for Topic Defense	■															
Drafting and Finalization of Presentation for Topic Defense	■															
Topic Defense		■														
Project Research and Consultation for Title Proposal			■	■												
Title Defense				■												
Finalization of Chapter 1: The Problem and Its Background				■												
Finalization of Chapter 2: Review of Related Literature				■												
Finalization of Chapter 3: Methodology				■	■	■										
Planning and Designing				■	■	■	■									
Studying and Programming using Python and Android Studio						■	■	■	■	■	■					
Progress Defense								■								
Testing and Evaluation of the Project											■	■	■			
Pre-Final Defense											■					
Documentation of the Project for Final Defense														■	■	■
Final Defense															■	
Finalization of the Project Document and Bookbinding																■

The Gantt chart shown in Table 4 is a visual framework of project timetable to be achieved. This is quite useful for managing and keeping the proponents on track, providing a visual timeline for starting and finishing the research.

CHAPTER 4

RESULTS AND DISCUSSION

This section shows the results of the data gathered from different tests and samples throughout the whole duration of the study. Different analyses were drawn from the data and experiments made.

4.1 Project Technical Description

MODERM application is a skin disease classification made available on Android smartphone that used transfer learning on pretrained MobileNet model with Python, utilizing Keras and Tensorflow. Then, the model was deployed on Android application with the help of Android Studio on providing the Graphical User Interface and Tensorflow for the inference of the trained model.

Keras provides a lot of open source pretrained models that can be used in transfer learning. Using transfer learning on the models, provide the researchers to train the skin disease image dataset collected from different publicly accessible dermatological online repositories, photo atlas of dermatology and some taken manually. After training, different models were evaluated based on different criteria. With the help of Tensorflow, the chosen model is then saved and converted to a protobuff file to be able to deploy on the Android application. Android studio was used to develop the Graphical User Interface of the application which now connects the user to the image classification system deployed on the Android smartphone.

4.2 Project Limitations and Capabilities

The following are the limitations of this study.

1. The MODERM Application is only available on the Android smartphone.
2. The developed application can only detect 7 skin diseases namely: acne, eczema, Pityriasis rosea, Psoriasis, Tinea corporis, Varicella(chickenpox) and vitiligo.
3. The application will always display result with the highest confidence level based on the 7 skin diseases mentioned above thus, testing the application on diseases not included on the dataset of the application will yield absolute misclassification.

4.3 Results of Evaluation with Different models

4.3.1 Confusion Matrices

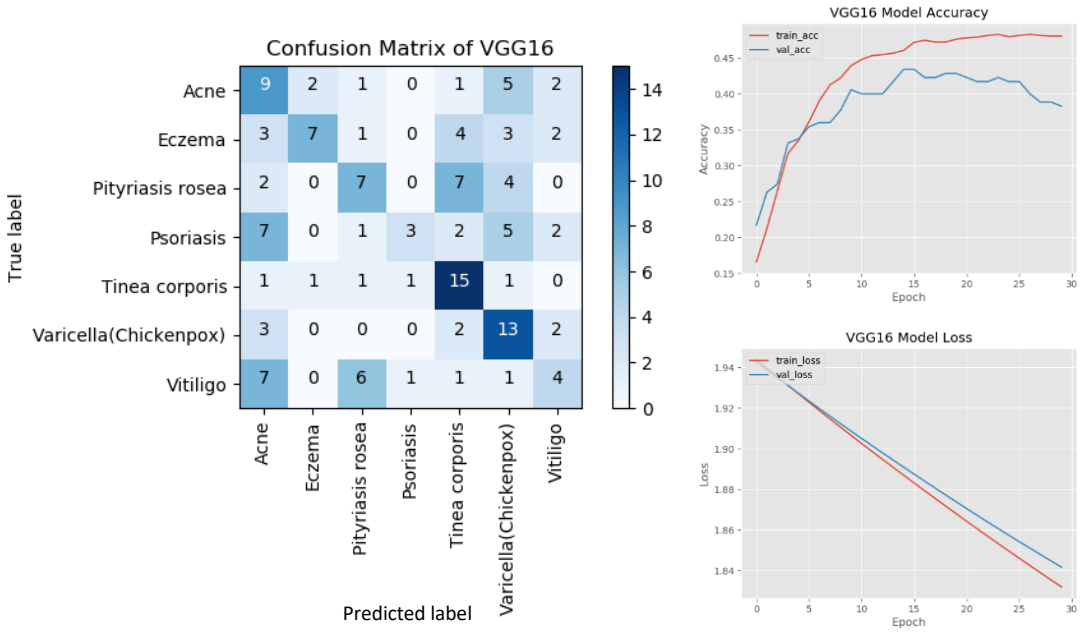


Figure 20. VGG16 confusion matrix, model accuracy, and model Loss

The VGG16 model classifier produced desirable results on the Tinea corporis with 75% accuracy but clearly fail to classify psoriasis and vitiligo test

images which have an accuracy of 15% and 20% respectively, while other classes also showed a very poor classification producing an accuracy of 41.4%.

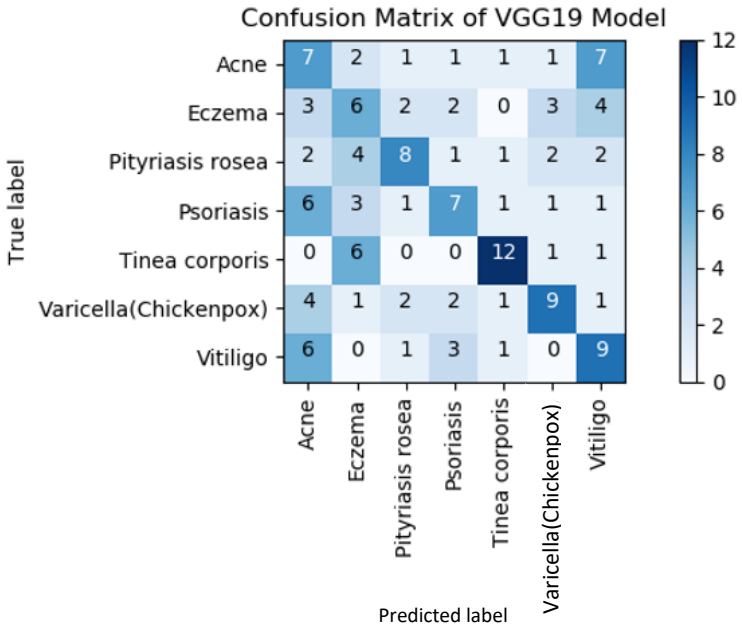


Figure 21. VGG19 confusion matrix

VGG 19 model classifier show a substandard result on the test dataset where the highest accuracy achieved is only 60% on the Tinea corporis class. The VGG19 attained an accuracy of 41.42%.

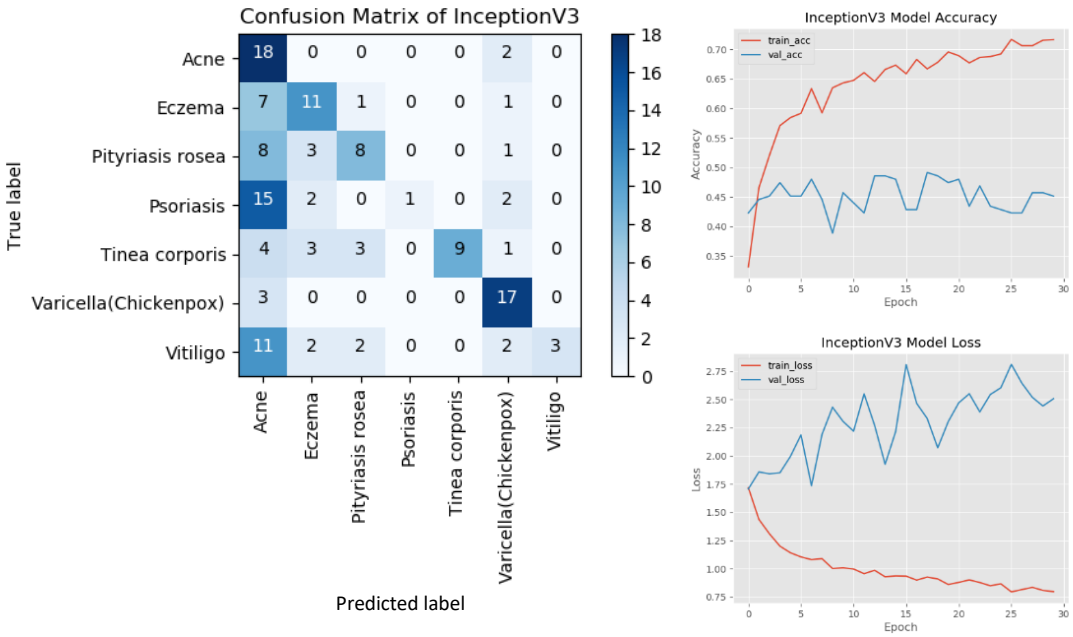


Figure 22. InceptionV3 confusion matrix, model accuracy, and model loss

The InceptionV3 model classifier mostly predicts the images in the test dataset as acne and varicella(chickenpox) resulting to a poor overall accuracy of the classifier. Based on this result, the model generalized the dataset as either acne or varicella(chickenpox) and achieved a 48.6% accuracy.

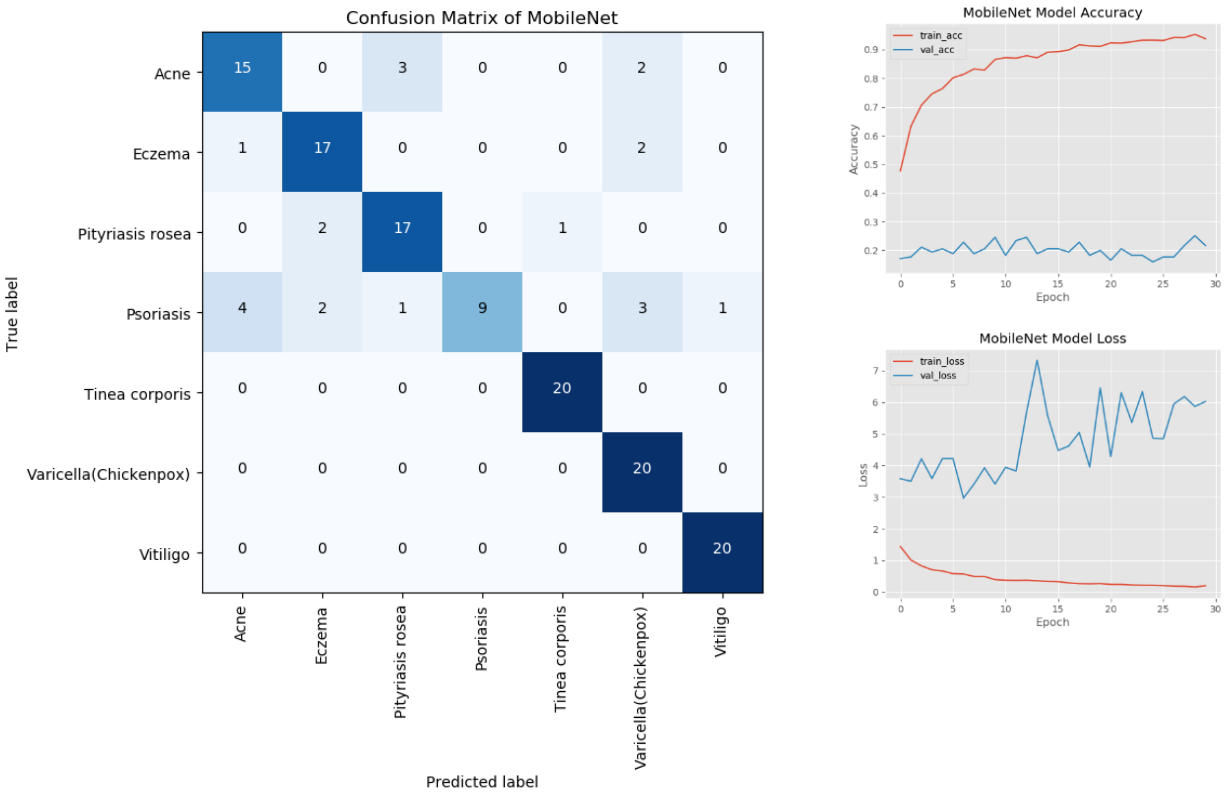


Figure 23. MobileNet confusion matrix, model accuracy, and model loss

The MobileNet model classifier generated an outstanding result on the test dataset wherein it attained an 84.28% accuracy. The result exhibits a slight misclassification on the other classes except acne and psoriasis. The result shows that the classifier had a bad performance classifying the psoriasis images where it only has an 45% accuracy.

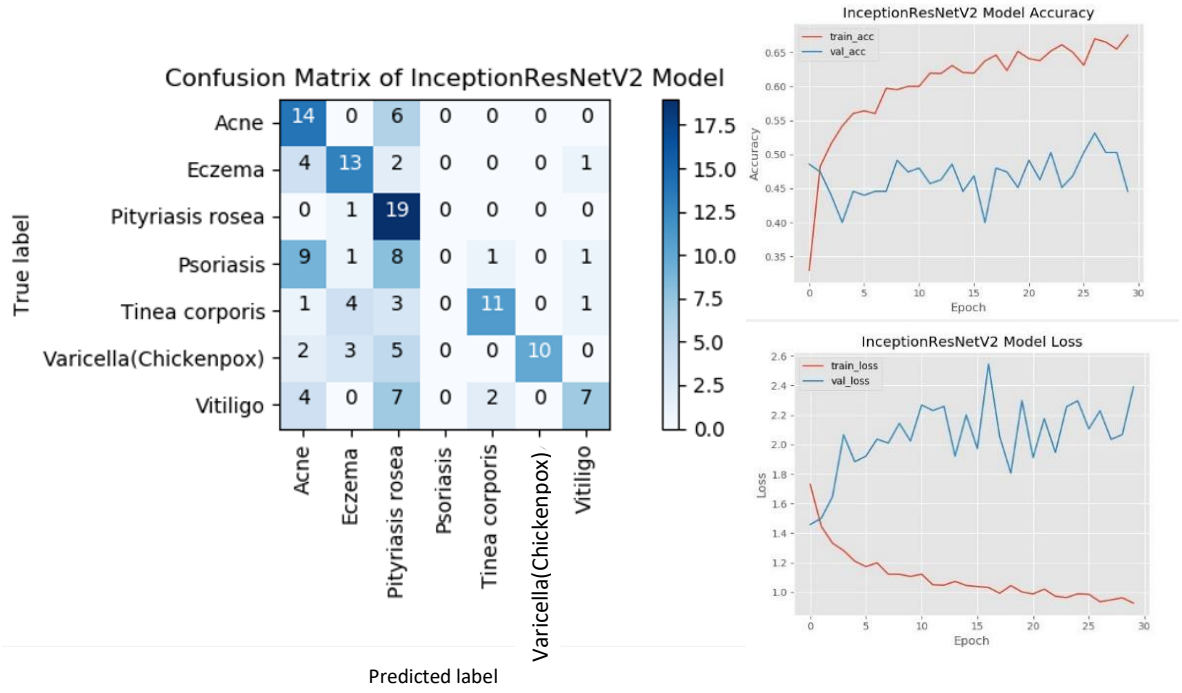


Figure 24. InceptionResNetV2 confusion matrix, model accuracy, and model loss

As observed in Figure 24, the InceptionResNetV2 failed to classify the psoriasis test dataset where the accuracy is 0%, and the model mostly predict Pityriasis rosea resulting in a poor overall classification accuracy with an 52.86%.

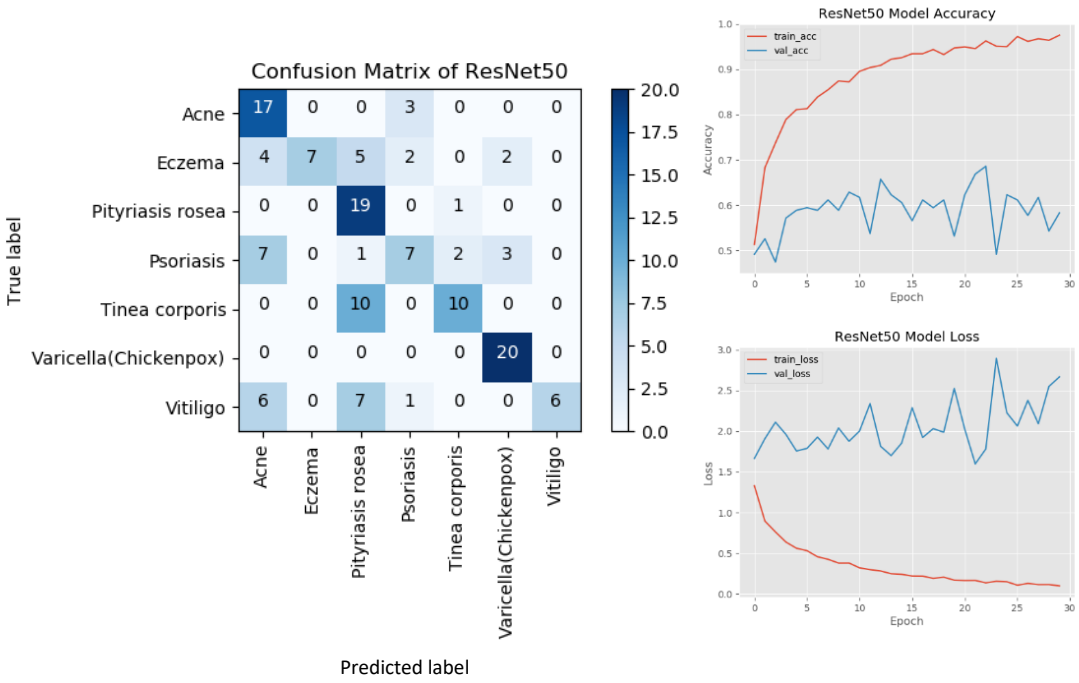


Figure 25. ResNet50 confusion matrix, model accuracy, and model loss

ResNet50 clearly showed a favorable result classifying varicella(chickenpox) with an accuracy of 100%, but the model provides a very poor classification because the model mostly predicts the images as Pityriasis rosea and can be clearly seen on the test dataset of Tinea corporis and vitiligo resulting to an 61.4% accuracy of the classifier.

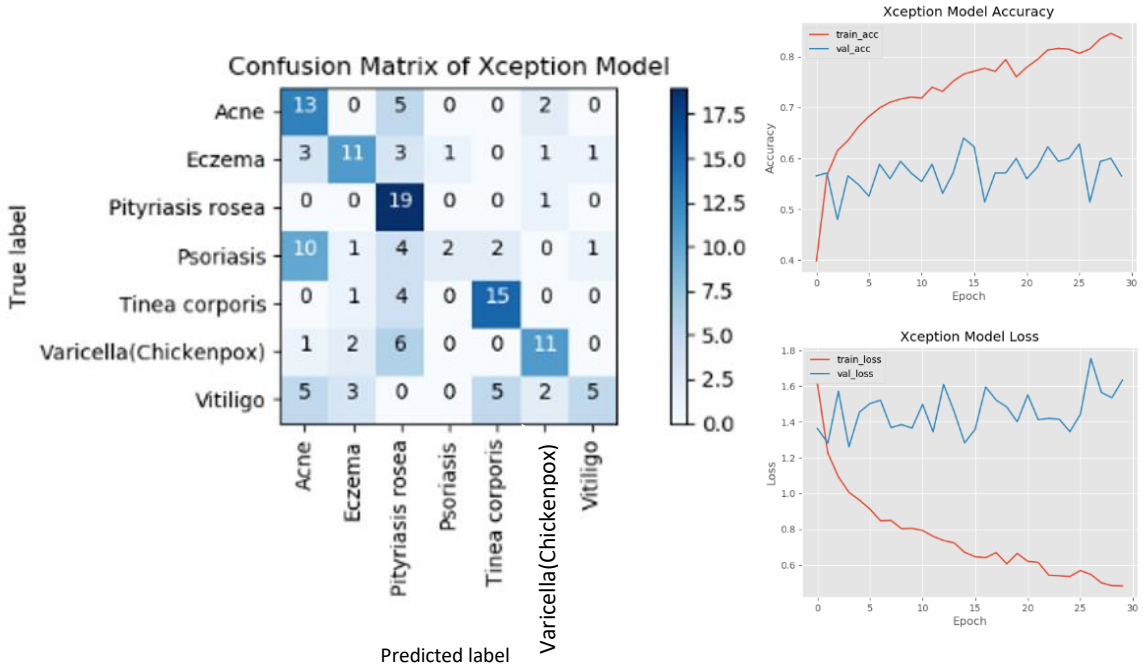


Figure 26. Xception confusion matrix, model accuracy and model loss

Xception model classifier attained a very poor classification on psoriasis where its accuracy is 10% as well as on Pityriasis rosea because its false positives is 16% of the dataset. Thus, the model has a poor overall classification accuracy of 54.29%.

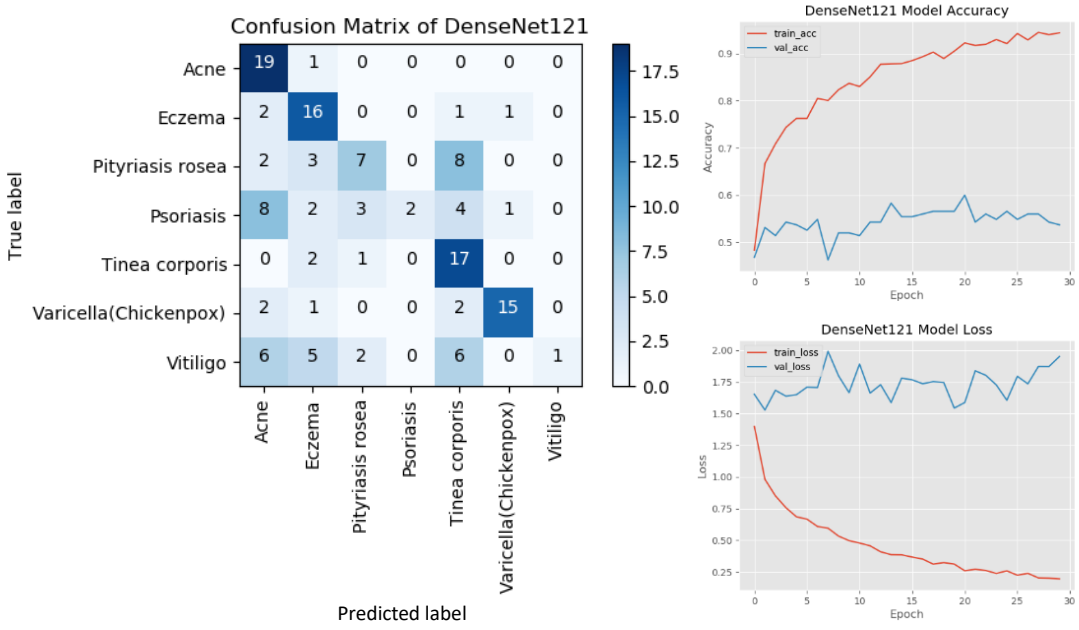


Figure 27. DenseNet121 confusion matrix, model accuracy, and model loss

DenseNet 121 model classifier showed an unreliable prediction on psoriasis and vitiligo, which generate a result of 10% and 5% accuracy respectively. The model mostly generalized the test dataset as either acne or Tinea corporis where the false positives of each class reach 14% of the dataset. As a result, the model provides a very poor classification accuracy of 64.3%.

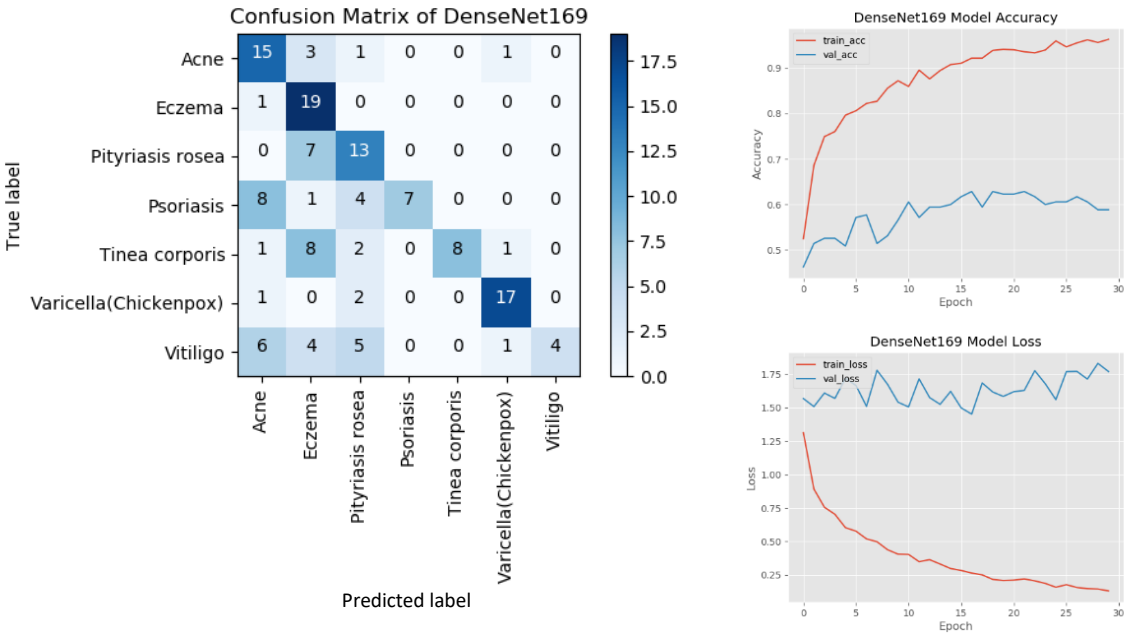


Figure 28. DenseNet169 confusion matrix, model accuracy and model loss

DenseNet169 classifier model provide an undesirable result on vitiligo with 20% accuracy and the model generalized the dataset as eczema wherein it generated a 16% of the dataset as false positives but the model shows good result in classifying varicella(chickenpox) with an 85% accuracy. But overall, the model provided poor classification resulting to an 59.3% accuracy.

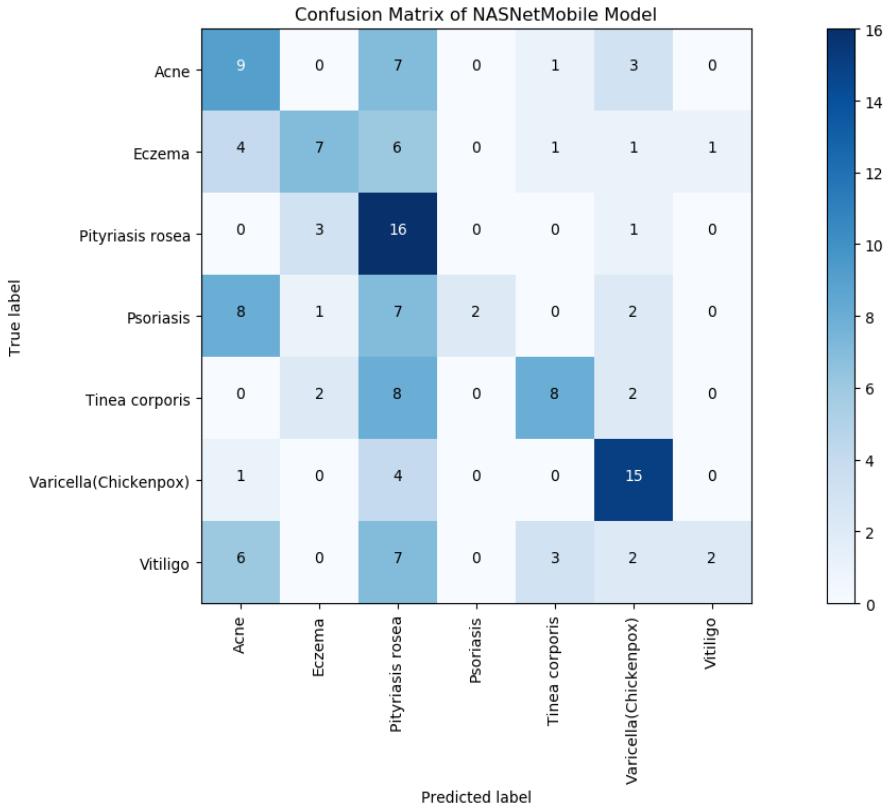


Figure 29. NASNet Mobile confusion matrix

NASNet Mobile model classifier clearly shows that the model generalized the dataset as Pityriasis rosea resulting to 28% false positives and yield an unreliable prediction result on psoriasis and vitiligo which have an accuracy of 10%. As a result, the model yields a low classification accuracy on the dataset which is 42.14%.

4.3.2 Accuracy

Accuracy is an evaluation metrics on classification models. It is defined as the ratio of the total correct prediction and the total number of predictions made. Figure 30 shows the accuracy of each model trained on the skin disease dataset. As can be seen, MobileNet provide the highest accuracy among the models which is 84.28%.

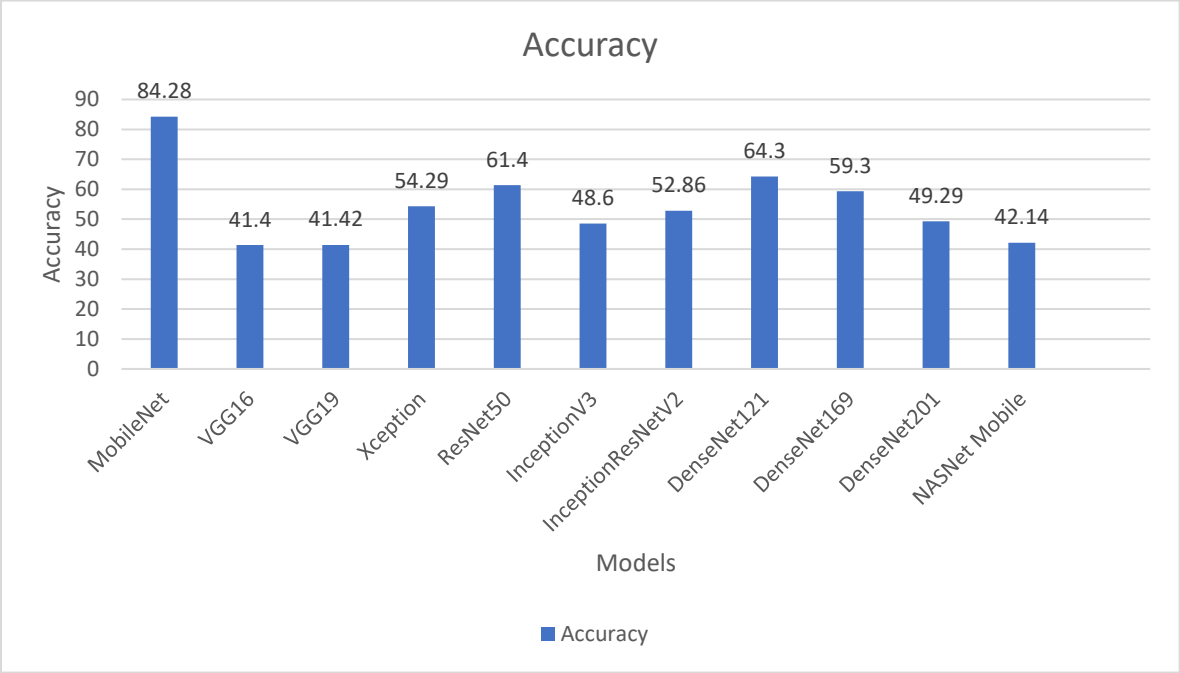


Figure 30. A comparison of accuracy results of each model

4.3.3 Weight Size

The model trained on the skin disease dataset was saved including the model architecture as well as the weights of the model on the skin dataset after 30 epochs. This saved model was then compared based on its weight size. The MobileNet is very lightweight having a 16.823MB size followed by NASNet mobile which is

23.66MB. While the VGG16 and VGG19 had a significant weight size of 540.496MB and 561.242MB respectively.

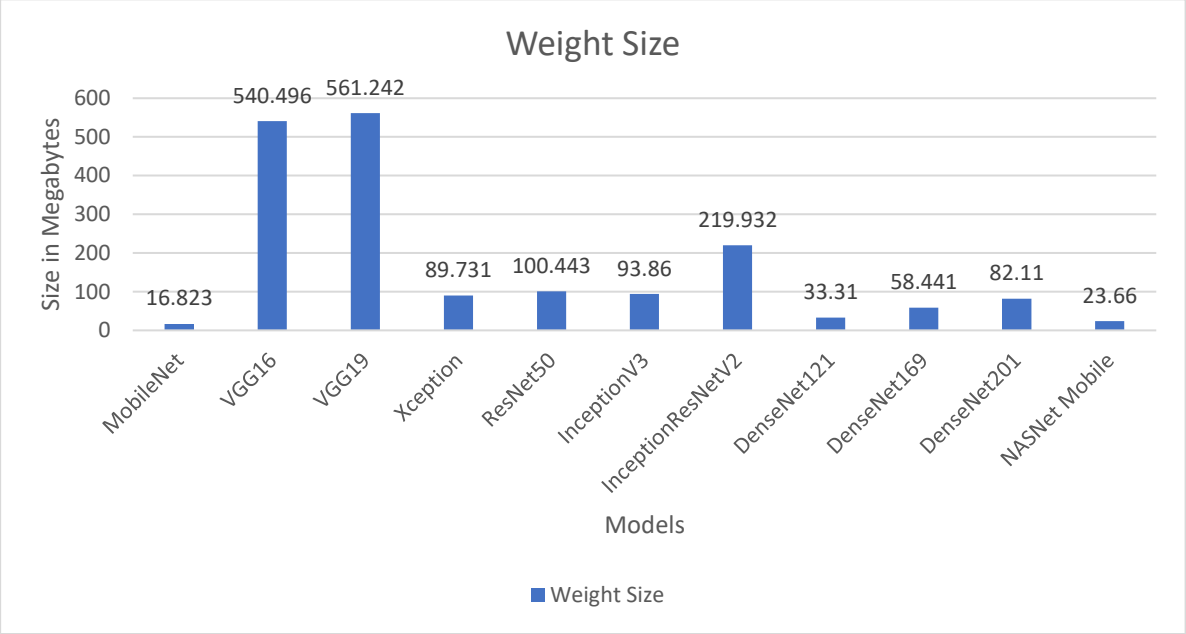


Figure 31. A comparison of weight sizes of each trained model

4.3.4 Loading Time

After saving the model, the trained model can now be easily access by loading the model. The proponents compute for its loading time. Figure 32 shows the loading time of each model using the load_model function made available by Keras. The fastest model to load is the VGG16 which only need 3.453 seconds followed by VGG19's loading time of 3.809 seconds. While the slowest model to load is the InceptionResNetV2 that takes 57.227 seconds.

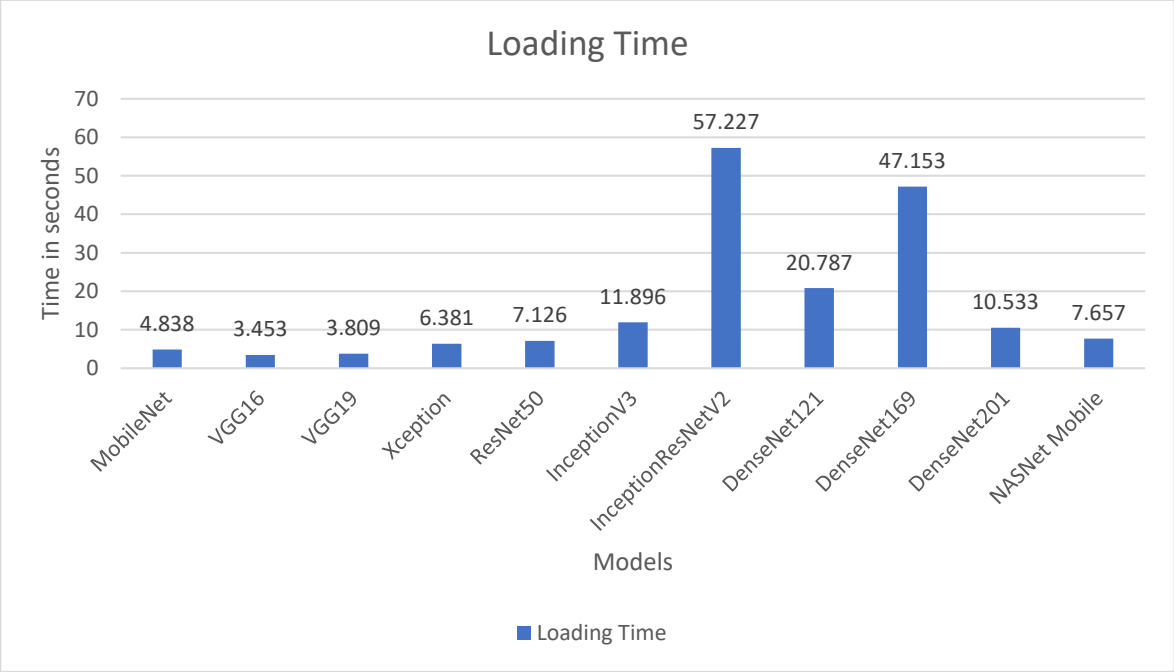


Figure 32. A comparison of loading time of each saved model

Table 5. Model evaluation summary

Model	Weight size	Loading time (seconds)	Accuracy (Percent)
MobileNet	16.823MB	4.838	84.28
VGG16	540.496MB	3.543	41.4
VGG19	561.242MB	3.809	41.42
Xception	89.731MB	6.381	54.29
ResNet50	100.443MB	7.126	61.4
InceptionV3	93.860MB	11.896	48.6
InceptionResNetV2	219.932MB	57.227	52.86
DenseNet121	33.317MB	20.787	64.3
DenseNet169	58.441MB	47.153	59.3
DenseNet201	82.110MB	10.533	49.29
NASNet Mobile	23.660MB	7.657	42.14

The results show that MobileNet model was the best classifier among the models, has an accuracy of 84.28%. This model is also lightweight having a weight size of 16.823MB. The loading time is 4.838 seconds making it the third fastest

model to load. But still having a promising accuracy and weight size is a doable trade-off to its loading time considering it is only a 1.295 seconds difference to the fastest model trained.

4.4 Results of MobileNet model

4.4.1 On Test Dataset

4.4.1.1 MobileNet_optimized

The MobileNet_optimized version is done with the default preprocessing function of Keras. The number of images per each skin disease is partition into 80% for the train data and 20% for the test data. The dataset used is considered as an imbalanced dataset. Evaluating the model on the test dataset generated an accuracy of 93.6%.

Table 6. Dataset used in MobileNet_optimized

Skin Disease	Number of Images	Train Data 80%	Test Data 20%	Validation Data
Acne	805	644	161	161
Eczema	599	479	120	120
Pityriasis rosea	347	278	69	69
Psoriasis	633	506	127	127
Tinea Corporis	385	308	77	77
Varicella(chickepox)	314	251	63	63
Vitiligo	323	258	65	65
Total	3406	2724	638	638

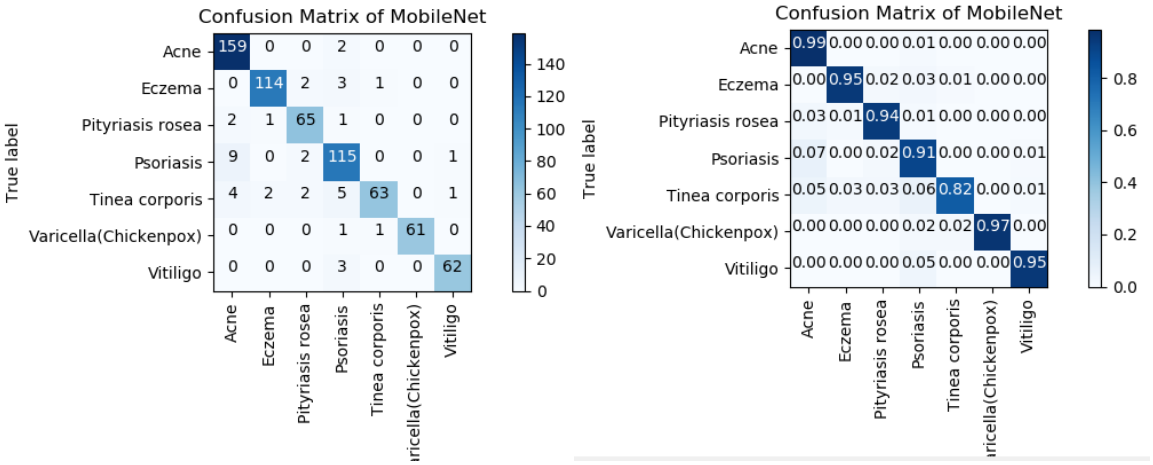


Figure 33. Mobilenet_optimized confusion matrix and normalized confusion matrix (Rank-1 accuracy: 93.6%)

4.4.1.2 Mobilenetv5

The MobileNetv5 is done by using the oversampling method where the imbalanced dataset is made to be balance by selecting the class with the highest number of input data and the remaining class will be adjusted based on this number of input data. This is done by providing a multiple copy of the input data of the smaller classes to have an identical number of input data per each class. The default preprocessing function of Keras was still used and the model attained an 91.8% accuracy.

Table 7. Dataset used in MobileNetv5

Skin Disease	Number of Images	Train Data 80%	Test Data 20%	Validation Data
Acne	805	644	161	161
Eczema	599	644	161	161
Pityriasis rosea	347	644	161	161
Psoriasis	633	644	161	161
Tinea Corporis	385	644	161	161
Varicella(chickepox)	314	644	161	161
Vitiligo	323	644	161	161
Total	3406	4508	1127	1127

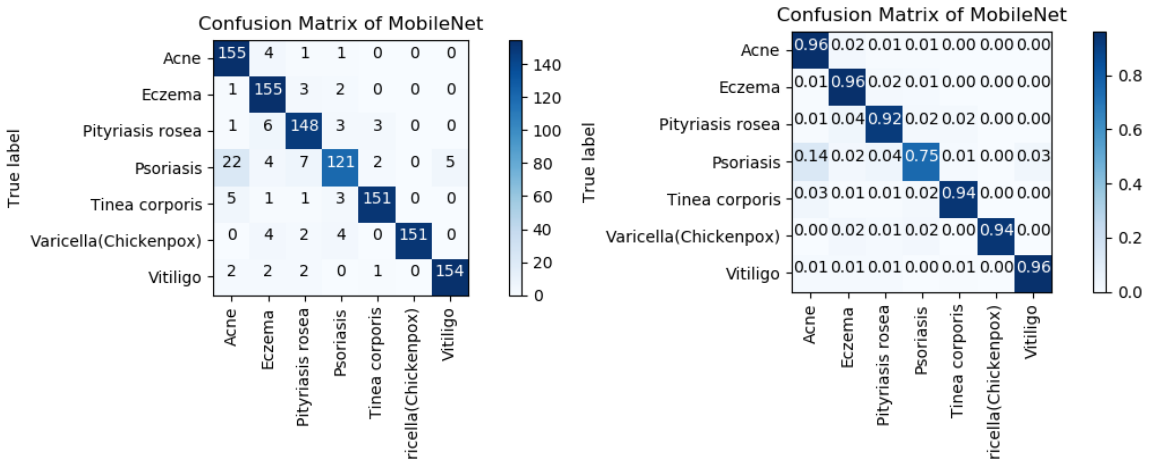


Figure 34. Mobilenetv5 confusion matrix and normalized confusion matrix (Rank-1 accuracy: 91.8%)

4.4.1.3 Mobilenetv7

MobileNetv7 still uses the oversampling method to deal with the imbalanced dataset but data augmentation was introduced in preprocessing the input data where in the model attained a 94.4% accuracy. Table 8 shows the following parameters observed in the data augmentation process.

Table 8. Parameters used in data augmentation

Process	Value
Rescale	1./255
Rotation range	40
Width shift range	0.2
Height shift range	0.2
Shear range	0.2
Zoom range	0.2
Horizontal flip	True
Fill mode	Nearest

Table 9. Dataset used in MobileNetv7

Skin Disease	Number of Images	Train Data 80%	Test Data 20%	Validation Data
Acne	805	644	161	161
Eczema	599	644	161	161
Pityriasis rosea	347	644	161	161
Psoriasis	633	644	161	161
Tinea Corporis	385	644	161	161
Varicella(chickepox)	314	644	161	161
Vitiligo	323	644	161	161
Total	3406	4508	1127	1127

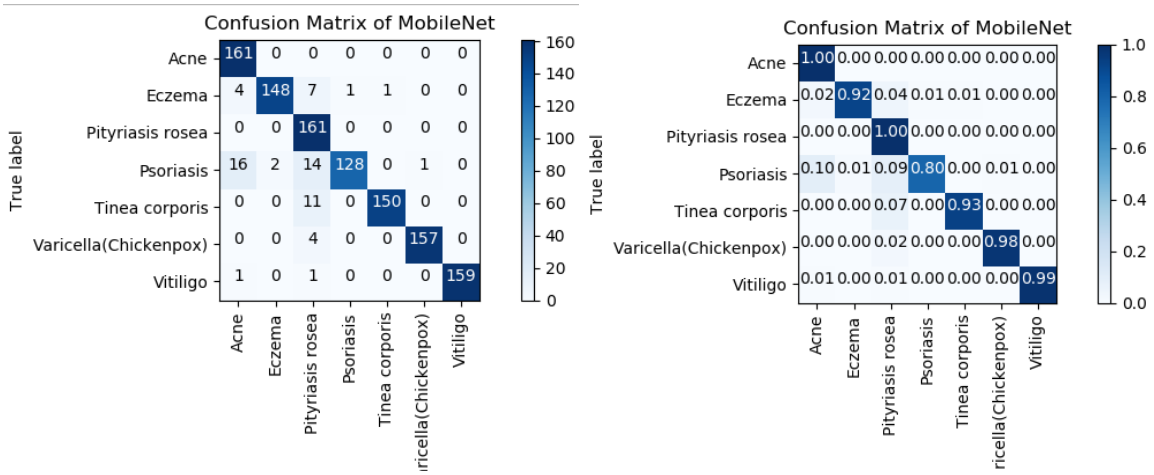


Figure 35 Mobilenetv7 confusion matrix and normalized confusion matrix (Rank-1 accuracy: 94.4%)

Table 10. Classification report of MobileNetv7

Rank-1 accuracy: 0.944

	precision	recall	f1-score	support
Acne	0.88	1.00	0.94	161
Eczema	0.99	0.92	0.95	161
Pityriasis rosea	0.81	1.00	0.90	161
Psoriasis	0.99	0.80	0.88	161
Tinea corporis	0.99	0.93	0.96	161
Varicella(Chickenpox)	0.99	0.98	0.98	161
Vitiligo	1.00	0.99	0.99	161
micro avg	0.94	0.94	0.94	1127
macro avg	0.95	0.94	0.94	1127
weighted avg	0.95	0.94	0.94	1127

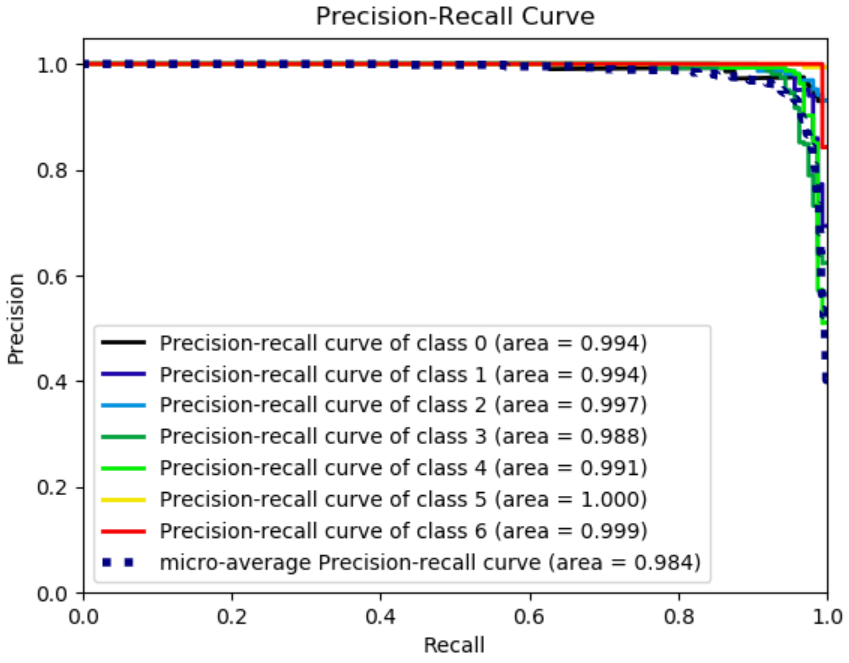


Figure 36. Precision-recall curve of MobileNetv7

Precision-Recall curve depicts the plot of the precision versus the recall of the model. It shows the model trade-off in prediction based on its true positive rate and positive predicted value. Figure 36 shows the Precision-Recall curve of the MobileNetv7 model

where, class 0 = acne, class 1 = eczema, class 2 = Pityriasis rosea, class 3 = psoriasis, class 4 = Tinea corporis, class 5 = varicella(chickenpox), and class 6 = vitiligo.

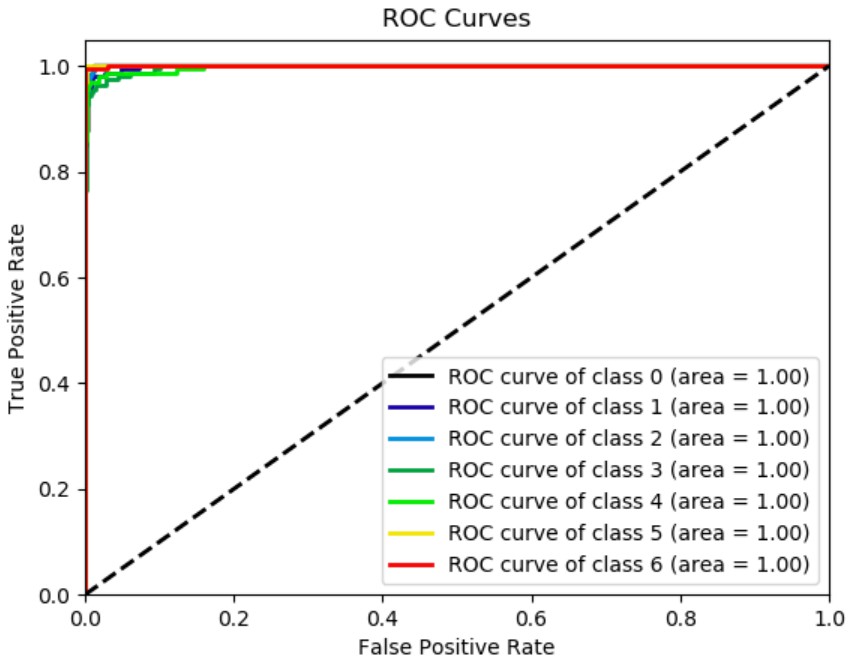


Figure 37. Receiver Operating Characteristic Curve or ROC Curve of MobileNetv7

ROC curve shows the plot between the false positive rate and true positive rate. This graph clearly described how the model function at predicting the positive class when the actual result is positive. Figure 37 shows the ROC curve of the MobileNetv7 model where each class provide a area under the curve of 1. Where, class 0 = acne, class 1 = eczema, class 2 = Pityriasis rosea, class 3 = psoriasis, class 4 = Tinea corporis, class 5 = varicella(chickenpox), and class 6 = vitiligo.

4.4.2 Field Testing

4.4.2.1 Comparison of Different MobileNet Versions

Table 11. Data gathered using the different MobileNet versions deployed on Android application.

Captured Image	MobileNet_optimized	MobileNetv5	MobileNetv7
Acne	Acne	Acne	Acne
Acne	Acne	Pityriasis rosea	Acne
Vitiligo	Vitligo	Vitiligo	Vitiligo
Psoriasis	Eczema	Tinea Corporis	Acne
Vitiligo	Vitiligo	Vitiligo	Vitligo
Pityriasis rosea	Pityriasis rosea	Pityriasis rosea	Pityriasis rosea
Eczema	Eczema	Eczema	Eczema
Tinea Corporis	Tinea corporis	Tinea corporis	Tinea corporis
Vitiligo	Vitiligo	Vitiligo	Acne
Vitiligo	Vitligo	Vitiligo	Vitiligo
Vitiligo	Vitiligo	Vitiligo	Vitiligo
Varicella(chickenpox)	Varicella(chickenpox)	Varicella(chickenpox)	Pityriasis rosea
Tinea corporis	Tinea corporis	Tinea corporis	Tinea corporis
Psoriasis	Pityriasis rosea	Eczema	Acne
Psoriasis	Psoriasis	Eczema	Pityriasis rosea
Varicella(chickenpox)	Acne	Varicella(chickenpox)	Varicella(chickenpox)

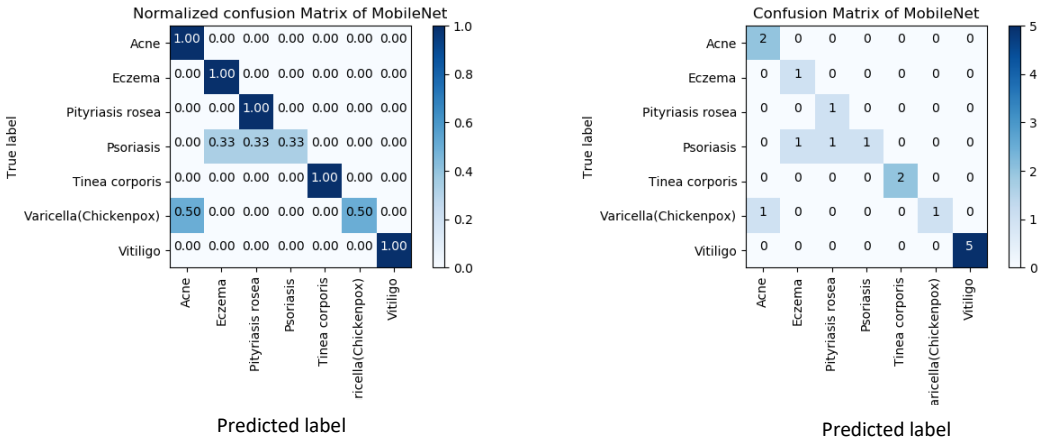


Figure 38. Confusion matrix and normalized confusion matrix for MobileNet_optimized (Rank-1 accuracy: 81.2%)

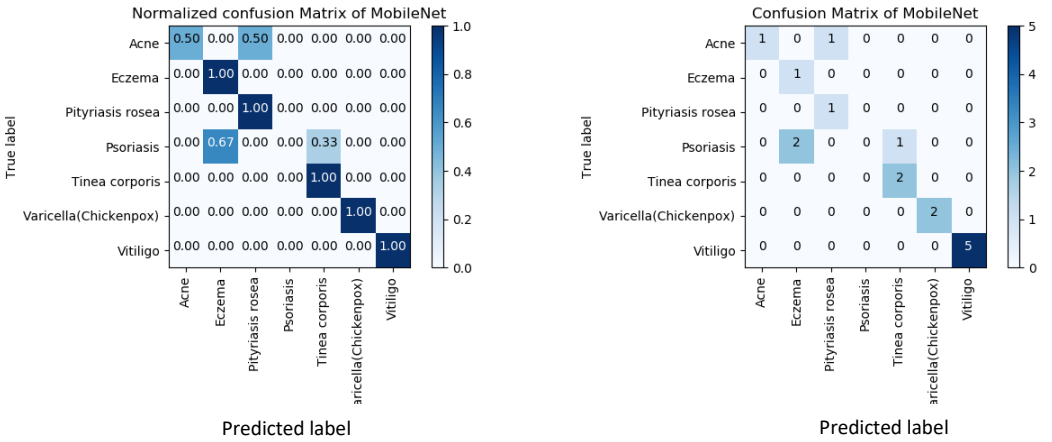


Figure 39. Confusion matrix and normalized confusion matrix for MobileNetv5 (Rank-1 accuracy: 75%)

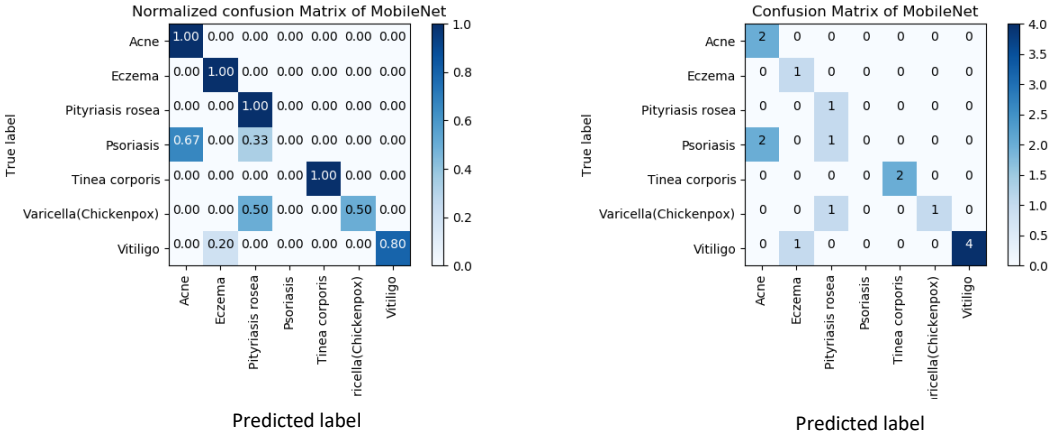


Figure 40. Confusion matrix and normalized confusion matrix for MobileNetv7 (Rank-1 accuracy: 68.8%)

4.5 Saliency Map

Saliency maps can be used to visualize how the model predict each class with a given input. The generated heatmap provide a way to intuitively visualize the location of the pixels where the model fixates most of its attention for diagnosis. As can be seen, the model most set it focus where the lesion is located.

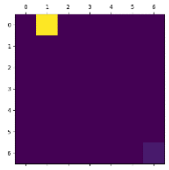
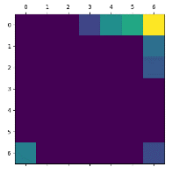
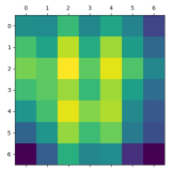
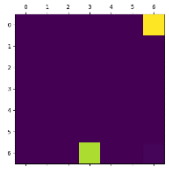
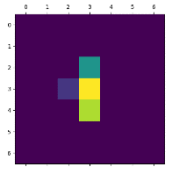
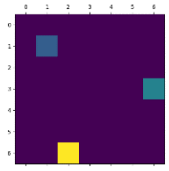
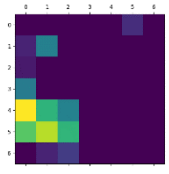
ACTUAL PICTURE	HEATMAP	IMPOSED HEATMAP
 **Acne**		
 **Eczema**		
 **Pityriasis rosea**		
 **Psoriasis**		
 **Tinea corporis**		
 **Varicella(chickenpox)**		
 **Vitiligo**		

Figure 41. Saliency maps for 7 skin disease sample images

CHAPTER 5

SUMMARY, CONCLUSION, AND RECOMMENDATION

This chapter includes the summary of the whole project, the conclusion derived from the results discussed in chapter 4, and recommendation for future development of the study.

5.1 Summary of Findings

The proponents were able to utilize transfer learning on different pretrained models made available by Keras. Evaluating the model based on different criteria like: accuracy, weight size, confusion matrix, and loading time provide the researchers to use the MobileNet model which has a promising result in skin disease classification, wherein it achieved a 84.28% accuracy with a lightweight size of 16.823MB and a loading time of 4.838 seconds, thus proving it to be a suitable model for deploying the trained Convolution Neural Network to an Android application. Then, it is further enhanced by different sampling methods and preprocessing of input images.

The proponents discovered that imbalanced dataset can be a major problem in the image classification system. Different sampling methods were experimented on the chosen MobileNet model. Undersampling provides a low accuracy on the test dataset with an accuracy of 84.28%. Splitting the dataset to 80% for train and 20% for test provide a 93.6% accuracy on the test dataset. When tested on the data gathered on field, it yields an 81.2%

accuracy, providing the overall most accurate result compared to the different techniques applied to further enhance the classification system. While MobileNetv5 generated a 91.8% accuracy on the test dataset but attained a 75% accuracy on field testing and the MobileNetv7 provide a 94.4% accuracy on the test dataset but generated an accuracy of 68.8% on field testing. On preprocessing the input data, the proponents explored the capability of data augmentation to improve the system. MobileNetv5 attained a 91.8% it is further improved by introducing a data augmentation process in preprocessing the input data and this version is MobileNetv7 that showed a significant accuracy improvement of 94.4% but when the system was applied on the field testing it produced the lowest accuracy. Dataset collection and field testing were the most challenging part of the research, the proponents tried different hospitals and dermatological institutions requesting to acquire a digital copy of the 7 skin diseases but were declined. Consequently, most of the images gathered are collected from open source dermatological repositories, and photo atlases of dermatology composed of 3,406 images.

5.2 Conclusion

From the conducted research study, the proponents conclude that:

1. Using transfer learning on available pretrained model provided by Keras was used on the skin disease dataset gathered and was implemented successfully. The proponent used the PyCharm IDE in training the models using Python programming language and different modules and libraries for deep learning like Keras, Tensorflow, scikit-learn, numpy and matplotlib.

2. Different fine-tuning techniques were used to train the model. Adam optimizer, categorical crossentropy for loss, and softmax for activation were the parameters used in compiling all the models.
3. The Graphical User Interface (GUI) of the application was develop using Android Studio. To fully deploy the model on the Android application the saved model should be converted to Tensorflow which is a protobuff file composed of model architecture and the labels.
4. Using data augmentation in preprocessing the input images yield a better accuracy compared to the conventional preprocessing of input images provided by Keras.
5. Sampling method like undersampling and oversampling provide a better accuracy on training the system with imbalanced dataset. Oversampling provides a much better accuracy but when the classification system is tested on real-world application, the accuracy suffers.

5.3 Recommendation

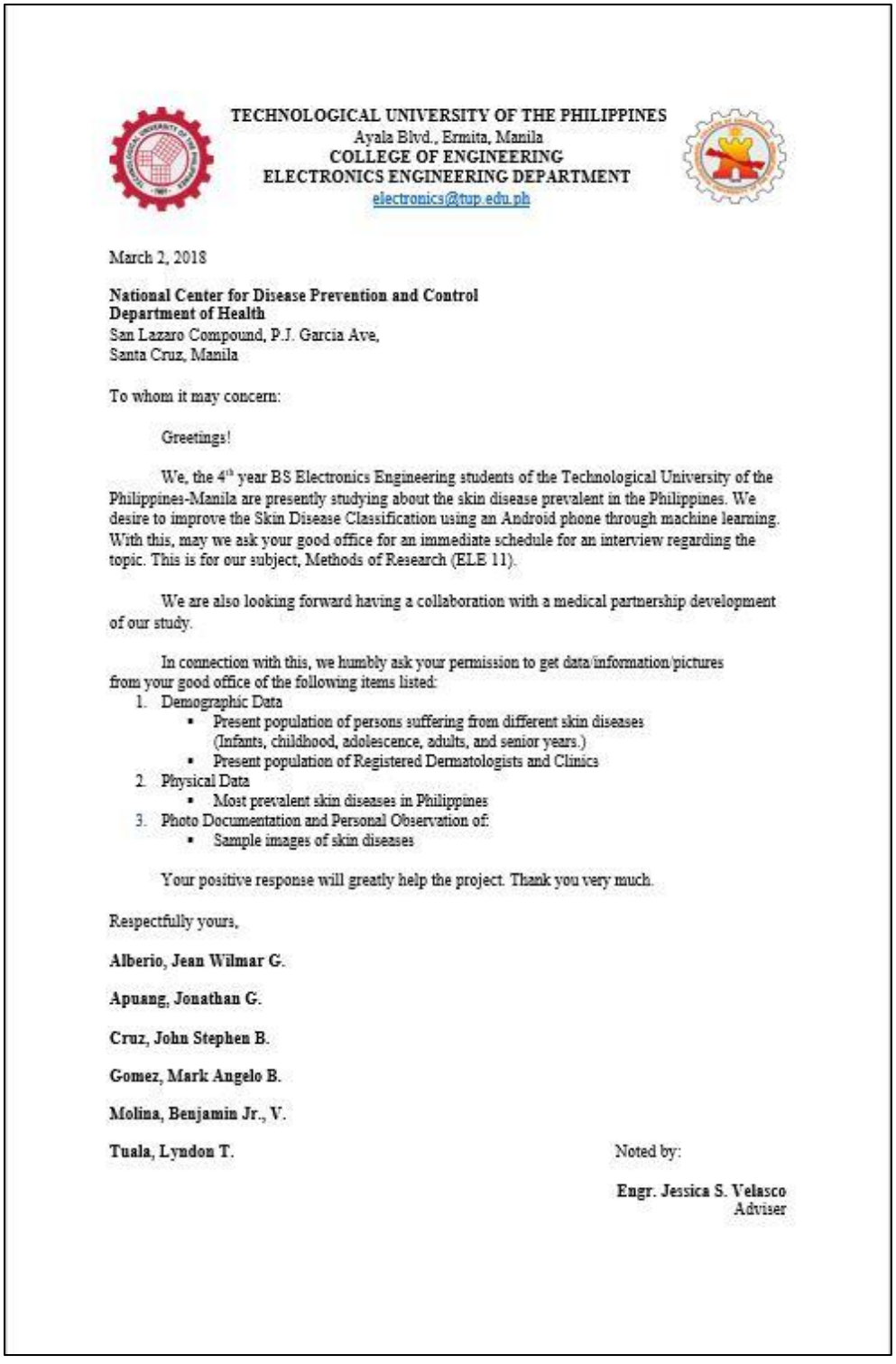
For further development of the study, these are the following recommendations:

1. This study can be developed by obtaining more data and classifying other types of skin disease.
2. Having partnership with a dermatological institution is highly recommended.
3. Confidence level of the prediction should be displayed as well in line with the skin disease in making an inference.
4. Implement ensemble learning to further enhance the classification system.

APPENDIX A

Evaluation Sheets and Certification

1. Letter of request for interview (DOH)



TECHNOLOGICAL UNIVERSITY OF THE PHILIPPINES
Ayala Blvd., Ermita, Manila
COLLEGE OF ENGINEERING
ELECTRONICS ENGINEERING DEPARTMENT
electronics@tup.edu.ph

March 2, 2018

National Center for Disease Prevention and Control
Department of Health
San Lazaro Compound, P.J. Garcia Ave,
Santa Cruz, Manila

To whom it may concern:

Greetings!

We, the 4th year BS Electronics Engineering students of the Technological University of the Philippines-Manila are presently studying about the skin disease prevalent in the Philippines. We desire to improve the Skin Disease Classification using an Android phone through machine learning. With this, may we ask your good office for an immediate schedule for an interview regarding the topic. This is for our subject, Methods of Research (ELE 11).

We are also looking forward having a collaboration with a medical partnership development of our study.

In connection with this, we humbly ask your permission to get data/information/pictures from your good office of the following items listed:

1. Demographic Data
   - Present population of persons suffering from different skin diseases (Infants, childhood, adolescence, adults, and senior years.)
   - Present population of Registered Dermatologists and Clinics
2. Physical Data
   - Most prevalent skin diseases in Philippines
3. Photo Documentation and Personal Observation of:
   - Sample images of skin diseases

Your positive response will greatly help the project. Thank you very much.

Respectfully yours,

Alberio, Jean Wilmar G.

Apuang, Jonathan G.

Cruz, John Stephen B.

Gomez, Mark Angelo B.

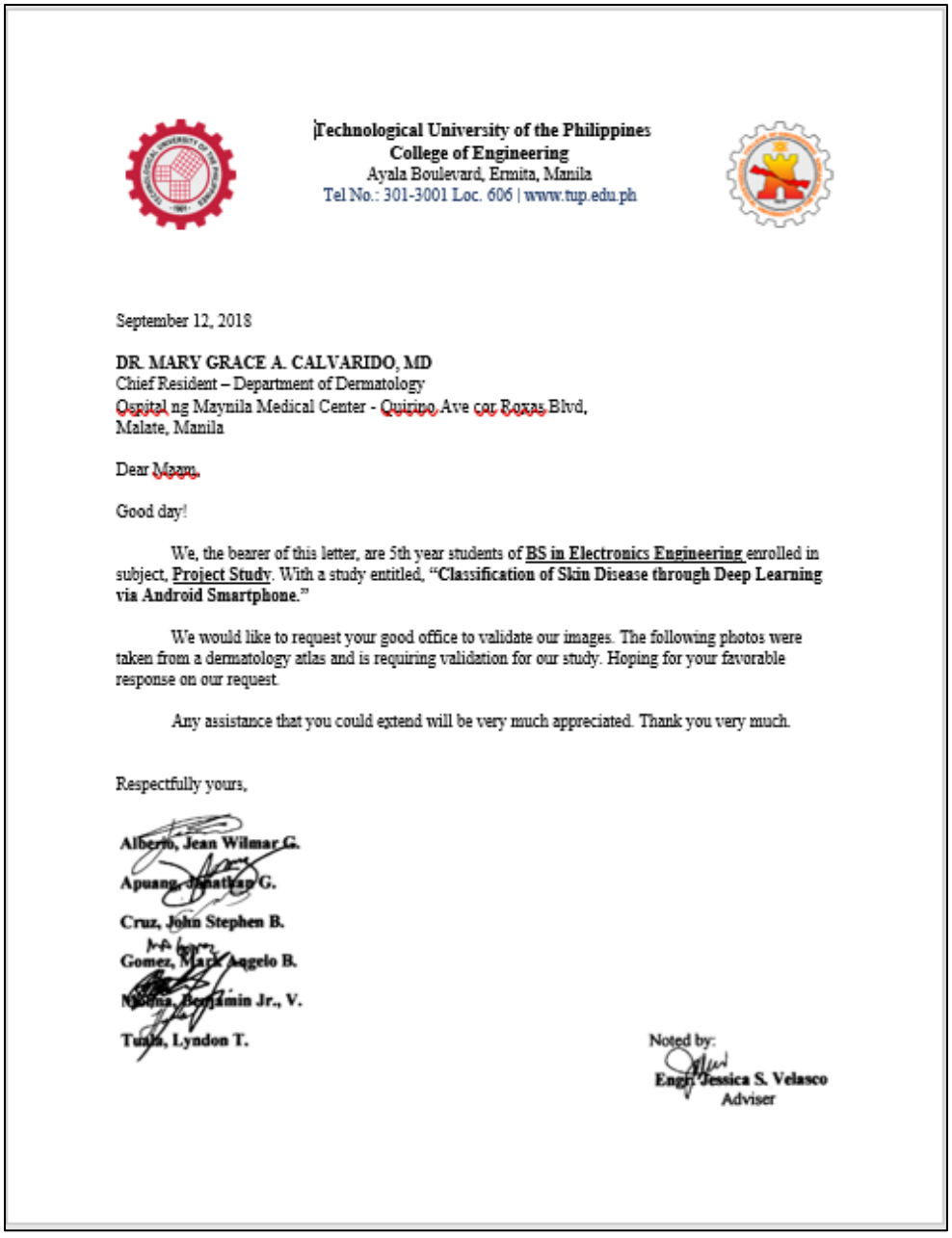
Molina, Benjamin Jr., V.

Tuala, Lyndon T.

Noted by:

Engr. Jessica S. Velasco
Adviser

2. Letter for validation of datasets (Ospital ng Maynila)



Technological University of the Philippines
College of Engineering
Ayala Boulevard, Ermita, Manila
Tel No.: 301-3001 Loc. 606 | www.tup.edu.ph

September 12, 2018

DR. MARY GRACE A. CALVARIDO, MD
Chief Resident – Department of Dermatology
Ospital ng Maynila Medical Center - Quirino Ave cor Roxas Blvd,
Malate, Manila

Dear Maam,

Good day!

We, the bearer of this letter, are 5th year students of BS in Electronics Engineering enrolled in subject, Project Study. With a study entitled, "**Classification of Skin Disease through Deep Learning via Android Smartphone.**"

We would like to request your good office to validate our images. The following photos were taken from a dermatology atlas and is requiring validation for our study. Hoping for your favorable response on our request.

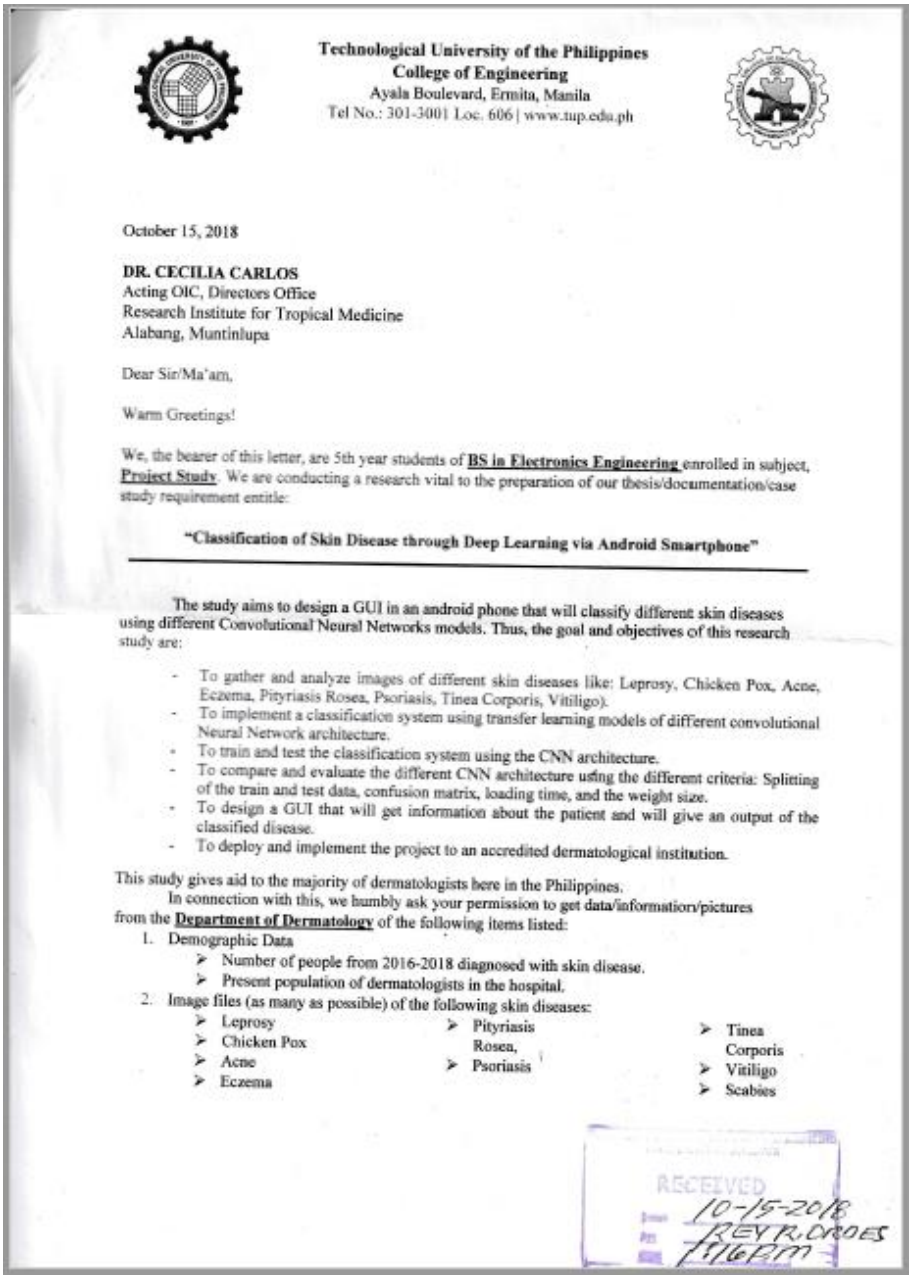
Any assistance that you could extend will be very much appreciated. Thank you very much.

Respectfully yours,

Alberto, Jean Wilmar G.
Apuang, Jonathan G.
Cruz, John Stephen B.
Gomez, Mark Angelo B.
Nocona, Benjamin Jr., V.
Tuazo, Lyndon T.

Noted by:
Engr. Jessica S. Velasco
Adviser

3. Letter of request for data (RITM)



Technological University of the Philippines
College of Engineering
Ayala Boulevard, Ermita, Manila
Tel No.: 301-3001 Loc. 606 | www.tup.edu.ph

October 15, 2018

DR. CECILIA CARLOS
Acting OIC, Directors Office
Research Institute for Tropical Medicine
Alabang, Muntinlupa

Dear Sir/Ma'am,

Warm Greetings!

We, the bearer of this letter, are 5th year students of **BS in Electronics Engineering** enrolled in subject, **Project Study**. We are conducting a research vital to the preparation of our thesis/documentation/case study requirement entitle:

"Classification of Skin Disease through Deep Learning via Android Smartphone"

The study aims to design a GUI in an android phone that will classify different skin diseases using different Convolutional Neural Networks models. Thus, the goal and objectives of this research study are:

- To gather and analyze images of different skin diseases like: Leprosy, Chicken Pox, Acne, Eczema, Pityriasis Rosea, Psoriasis, Tinea Corporis, Vitiligo).
- To implement a classification system using transfer learning models of different convolutional Neural Network architecture.
- To train and test the classification system using the CNN architecture.
- To compare and evaluate the different CNN architecture using the different criteria: Splitting of the train and test data, confusion matrix, loading time, and the weight size.
- To design a GUI that will get information about the patient and will give an output of the classified disease.
- To deploy and implement the project to an accredited dermatological institution.

This study gives aid to the majority of dermatologists here in the Philippines.

In connection with this, we humbly ask your permission to get data/information/pictures from the **Department of Dermatology** of the following items listed:

1. Demographic Data
   - Number of people from 2016-2018 diagnosed with skin disease.
   - Present population of dermatologists in the hospital.
2. Image files (as many as possible) of the following skin diseases:
   - Leprosy
   - Chicken Pox
   - Acne
   - Eczema
   - Pityriasis Rosea,
   - Psoriasis
   - Tinea Corporis
   - Vitiligo
   - Scabies

RECEIVED
10-15-2018
REY RODRIGUES
1:16PM

4. Letter of request for data (PNP Camp Crame)

Technological University of the Philippines
College of Engineering
Ayala Boulevard, Ermita, Manila
Tel No.: 301-3001 Loc. 606 | www.tup.edu.ph

January 29, 2019

PSSUPT REIMOUND C. SALES
Chief
PNP General Hospital, Health Service.
Camp Crame, Q.C

Dear Sir,

Warm Greetings!

We, the bearer of this letter, are 5th year students of **BS in Electronics Engineering** enrolled in subject, **Project Study**. We are conducting a research vital to the preparation of our thesis/documentation/case study requirement entitle:

"Classification of Skin Disease through Deep Learning via Android Smartphone"

The study aims to design a GUI in an android phone that will classify different skin diseases using different Convolutional Neural Networks models. Thus, the goal and objectives of this research study are:

- To gather and analyze images of different skin diseases like: Leprosy, Chicken Pox, Acne, Eczema, Pityriasis Rosea, Psoriasis, Tinea Corporis, Vitiligo).
- To implement a classification system using transfer learning models of different convolutional Neural Network architecture.
- To train and test the classification system using the CNN architecture.
- To compare and evaluate the different CNN architecture using the different criteria: Splitting of the train and test data, confusion matrix, loading time, and the weight size.
- To design a GUI that will get information about the patient and will give an output of the classified disease.
- To deploy and implement the project to an accredited dermatological institution.

This study gives aid to the majority of dermatologists here in the Philippines.

In connection with this, we humbly ask your permission to get data/information/pictures from the **Department of Dermatology** of the following items listed:

1. Demographic Data
   - Number of people from 2016-2018 diagnosed with skin disease.
   - Present population of dermatologists in the hospital.
2. Image files (as many as possible) of the following skin diseases:
   - Leprosy
   - Chicken Pox
   - Acne
   - Eczema
   - Pityriasis Rosea,
   - Psoriasis
   - Tinea Corporis
   - Vitiligo
   - Scabies

5. Sample waiver for patients

TECHNOLOGICAL UNIVERSITY OF THE PHILIPPINES
Ayala Blvd., Ermita, Manila
COLLEGE OF ENGINEERING
ELECTRONICS ENGINEERING DEPARTMENT
electronics@tup.edu.ph

Consent Form for Image Data Capturing
Pahintulot sa Pagkuha ng Datos na Imahe/Litrato

Ako, _____, ____ taong gulang, mula sa _____, ay nagbibigay pahintulot sa mga mananaliksik ng unibersidad na kuhanan ng litrato /imahe ang aking balat para gamitin sa kanilang proyekto sa pananaliksik.

Sumasang-ayon ako sa kanilang gagawin na pagkuha sakin ng litrato ngunit sa ilang mga kondisyong ito:
1. Na mananatiling pribado ang aking pagkakakilanlan
2. Na gagamtin lang ang aking litrato ng balat para sa proyekto lamang at hindi na kung saan-saan pa.

_____
Lagda

Sasagutin ng aming grupo ang alinmang katanungan patungkol sa proyekto at nangangakong igagalang ang mga nasabing kondisyon. Maaaring kontakin ang aming adviser sa anumang katanungan patungkol sa gawaing ito.

Mga mananaliksik:

Alberio, Jean Wilmar G.
Apuang, Jonathan G.
Cruz, John Stephen B.
Gomez, Mark Angelo B.
Ngaona, Benjamin Jr., V.
Tuzla, Lyndon T.

Adviser ng proyekto:

Engr. Jessica S. Velasco
09152815325

6. Deployment letter (Sucat Health Center)

TECHNOLOGICAL UNIVERSITY OF THE PHILIPPINES
Ayala Blvd., Ermita, Manila
COLLEGE OF ENGINEERING
ELECTRONICS ENGINEERING DEPARTMENT
electronics@tup.edu.ph

January 29, 2019

Dra. Rowena R. Tan
General Practitioner
Sucat Health Center
Muntinlupa City

Dear Sir,

Warm Greetings!

We the bearer of this letter, are students of **BS in Electronics Engineering** enrolled in **Project Study 2, Lab (PS 2L)**. We are conducting a research vital to the preparation of our thesis requirement entitle:

"Classification of Skin Disease through Deep Learning via Android Smartphone"

The study aims to design and develop an Android application that will classify different skin diseases. Thus, the goal and objectives of this research study are:

- Gather and analyze images of different skin diseases like: Chickenpox, Acne, Eczema, Pityriasis rosea, Psoriasis, Tinea corporis, Vitiligo.
- Implement a classification system using transfer learning models of different convolutional Neural Network architecture.
- Train and test the classification system using the CNN architecture.
- Compare and evaluate the different CNN architecture using the different criteria: Splitting of the train and test data, confusion matrix, loading time, and the weight size.
- Design a GUI that will get information about the patient and will give an output of the classified disease.
- Deploy and implement the project to an accredited dermatological institution.

This second semester, we are about to start the fourth chapter of our study which is the result and discussion. To enable this, we are humbly requesting your expertise for the validation of our classification system. In connection with this, we humbly ask your permission to conduct activities from your good office listed below:

1. Validation of the Accuracy of the Classification System: [1]
   - May the researchers request dermatologists to evaluate the accuracy of the classification system.
2. Evaluation of the Android Application: [2]
   - Respondents are kindly asked to accomplish a feedback form.
3. Field Deployment: [3]
   - Deploy the Android application on your good office to field test the system.

Any assistance that you could extend will be very much appreciated. Please refer to the attached documents for your kind perusal. Thank you very much.

Respectfully yours,

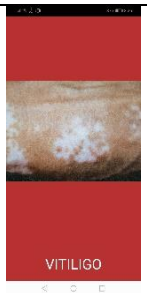
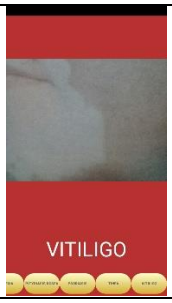
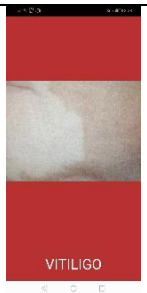
Alberto, Jean Wilmar G.
Apuang, Jonathan G.
Cruz, John Stephen B.
Gomez, Mark Anggelo B.
Nicolas, Benjamin Jr., V.
Tuala, Lyndon T.

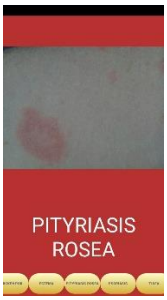
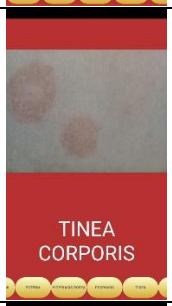
Noted by:
Engr. Jessica S. Velasco
Adviser

APPENDIX B

Tables

1. Comparison of Different MobileNet Versions

Captured Image	MobileNet_optimized	MobileNetv5	MobileNetv7
 Acne			
 Acne			
 Vitiligo			
 Eczema			
 Vitiligo			

 **Pityriasis rosea**			
 **Eczema**			
 **Tinea Corporis**			
 **Vitiligo**			
 **Vitiligo**			

 **Vitiligo**			
 **Chickenpox**			
 **Tinea corporis**			
 **Psoriasis**			
 **Psoriasis**			

Chickenpox

2. Datasets

Disease	Number of images
Acne	809
Psoriasis	633
Pityriasis rosea	347
Varicella(chickenpox)	314
Vitiligo	323
Eczema	599
Tinea corporis	385
TOTAL	3411

APPENDIX C

Bill of Materials

Bill of Materials

Summary of Expenses

Expenses		Amount (Php)
Doctor's Appointment Fee	Manila Doctor's Hospital	500
	Manila Medical Center	500
Data Gathering/Deployment Token (for volunteer patient)		3,000
Cellphone Device		6,000
Total:		10,000

APPENDIX D

Source Codes

MobileNet

```
import numpy as np
import keras
from keras.layers import Dense
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

print ("[INFO] program started")

train_path='Skin_diseaseV5/train'                    #
valid_path='Skin_diseaseV5/validation' test_path='Skin_diseaseV5/test'
test_path2='Skin_diseaseV5/test2'

class_labels=['Acne',
              'Eczema',
              'Pityriasis rosea',
              'Psoriasis',
              'Tinea corporis',
              'Varicella(Chickenpox)',
              'Vitiligo']

train_batches=ImageDataGenerator(rescale=1./255,
                                 rotation_range=40,
                                 width_shift_range=0.2,
                                 height_shift_range=0.2,
                                 shear_range=0.2,
                                 zoom_range=0.2,
```

```
                    horizontal_flip=True,
                    fill_mode='nearest',

preprocessing_function=keras.applications.mobilenet.preprocess_input)\
   .flow_from_directory(train_path, target_size=(224,224),
                    classes=class_labels, batch_size=5)
valid_batches=ImageDataGenerator(preprocessing_function=keras.applications.mobilene
t.preprocess_input)\
   .flow_from_directory(valid_path, target_size=(224,224),
                    classes=class_labels, batch_size=5)

test_batches=ImageDataGenerator(rescale=1./255 ,

preprocessing_function=keras.applications.mobilenet.preprocess_input)\
   .flow_from_directory(test_path, target_size=(224,224),
                    classes=class_labels, batch_size=5, shuffle=False)
test_batches2=ImageDataGenerator(rescale=1./255 ,

preprocessing_function=keras.applications.mobilenet.preprocess_input)\
   .flow_from_directory(test_path2, target_size=(224,224),
                    classes=class_labels, batch_size=5, shuffle=False)

base_model = keras.applications.mobilenet.MobileNet()
x=base_model.layers[-6].output
predictions = Dense(7, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)
N=30
model.compile(Adam(lr=.0001), loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
history=model.fit_generator(train_batches, steps_per_epoch=902,
validation_data=valid_batches,
                validation_steps=226, epochs=N, verbose=1) print ("[INFO] evaluating
model...")
test_labels=test_batches.classes
test_labels2=test_batches2.classes
predictions=model.predict_generator(test_batches, steps=226, verbose=1)
predictions2=model.predict_generator(test_batches2, steps=16, verbose=1)
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)
```

```
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

print ("[INFO] confusion matrix")

cm=confusion_matrix(test_labels, predictions.argmax(axis=1))
plt.figure()
plot_confusion_matrix(cm, class_labels,normalize=False, title='Confusion Matrix of MobileNet')
plt.savefig('Confusion_Matrix_of_MobileNet.png')

cm=confusion_matrix(test_labels2, predictions2.argmax(axis=1))
plt.figure()
plot_confusion_matrix(cm, class_labels,normalize=False, title='Confusion Matrix of MobileNet w/ Deployment Dataset')
plt.savefig('Confusion_Matrix_of_MobileNet_with_Deployment_Test_Dataset.png')

cm=confusion_matrix(test_labels, predictions.argmax(axis=1))
plt.figure()
plot_confusion_matrix(cm, class_labels,normalize=True, title='Confusion Matrix of MobileNet')
plt.savefig('Confusion_Matrix_of_MobileNet_Normalized.png')
```

```
cm=confusion_matrix(test_labels2, predictions2.argmax(axis=1))
plt.figure()
plot_confusion_matrix(cm, class_labels,normalize=True, title='Confusion Matrix of MobileNet w/ Deployment Dataset')
plt.savefig('Confusion_Matrix_of_MobileNet_Normalized_with_Deployment_Test_Dataset.png')

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

score=accuracy_score(test_labels, predictions.argmax(axis=1))
print("Rank-1 accuracy: %0.3f" % score)
print(classification_report(test_labels, predictions.argmax(axis=1),
target_names=class_labels))

score=accuracy_score(test_labels2, predictions2.argmax(axis=1))
print("Rank-1 accuracy with Deployment Test Dataset: %0.3f" % score)
print(classification_report(test_labels2, predictions2.argmax(axis=1),
target_names=class_labels))

print ("[INFO] plotting the results...")
plt.figure()
plt.style.use("ggplot")
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.title("MobileNet Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(loc="upper left")
```

```
plt.show()
plt.savefig('MobileNet_Model_Loss.png')
```

NASnet Mobile

```
import numpy as np
import keras
from keras.layers import Dense, GlobalAveragePooling2D
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

print ("[INFO] program started")

train_path='Skin_DiseasesV2/train'
valid_path='Skin_DiseasesV2/validation'
test_path='Skin_DiseasesV2/test'

class_labels=['Acne',
              'Eczema',
              'Pityriasis rosea',
              'Psoriasis',
              'Tinea corporis',
              'Varicella(Chickenpox)',
              'Vitiligo']
```

```
train_batches=ImageDataGenerator(preprocessing_function=keras.applications.nasnet.preprocess_input)\
    .flow_from_directory(train_path, target_size=(224,224), classes=class_labels, batch_size=5)
valid_batches=ImageDataGenerator(preprocessing_function=keras.applications.nasnet.preprocess_input)\
    .flow_from_directory(valid_path, target_size=(224,224), classes=class_labels, batch_size=5)
test_batches=ImageDataGenerator(preprocessing_function=keras.applications.nasnet.preprocess_input)\
    .flow_from_directory(test_path, target_size=(224,224), classes=class_labels, batch_size=5, shuffle=False)

base_model = keras.applications.nasnet.NASNetMobile(include_top=False, input_shape=(224,224,3))

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(7, activation='softmax')(x)

for layer in base_model.layers:
    layer.trainable = False

model = Model(inputs=base_model.input, outputs=predictions)
model.summary()

print ("[INFO] successfully loaded base model and model...")
print ("[INFO] training started...")
N=30
```

```
model.compile(Adam(lr=.0001), loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit_generator(train_batches, steps_per_epoch=412, validation_data=valid_batches,
                    validation_steps=35, epochs=N, verbose=1)

print ("[INFO] evaluating model...")

test_labels=test_batches.classes
predictions=model.predict_generator(test_batches, steps=28, verbose=1)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
```

```
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print ("[INFO] confusion matrix")

cm=confusion_matrix(test_labels, predictions.argmax(axis=1))
plt.figure()
plot_confusion_matrix(cm, class_labels,normalize=False, title='Confusion Matrix of NASNetMobile')
plt.savefig('Confusion_Matrix_of_NASNetMobile.png')

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

score=accuracy_score(test_labels, predictions.argmax(axis=1))
print("Rank-1 accuracy: %0.3f" % score)
print(classification_report(test_labels, predictions.argmax(axis=1),
target_names=class_labels))
```

```
print ("[INFO] plotting the results...")

plt.figure()
plt.style.use("ggplot")
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.title("NASNetMobile Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(loc="upper left")
plt.show()
plt.savefig('NASNetMobile_Model_Loss.png')
```

Xception

```
import numpy as np
import keras
from keras.layers import Dense, GlobalAveragePooling2D
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

print ("[INFO] program started")

train_path='Skin_DiseasesV2/train'
valid_path='Skin_DiseasesV2/validation'
```

```
test_path='Skin_DiseasesV2/test'

class_labels=['Acne',
              'Eczema',
              'Pityriasis rosea',
              'Psoriasis',
              'Tinea corporis',
              'Varicella(Chickenpox)',
              'Vitiligo']

train_batches=ImageDataGenerator(preprocessing_function=keras.applications.xception.
preprocess_input)\
   .flow_from_directory(train_path, target_size=(299,299), classes=class_labels,
batch_size=5)
valid_batches=ImageDataGenerator(preprocessing_function=keras.applications.xception.
preprocess_input)\
   .flow_from_directory(valid_path, target_size=(299,299), classes=class_labels,
batch_size=5)
test_batches=ImageDataGenerator(preprocessing_function=keras.applications.xception.p
reprocess_input)\
   .flow_from_directory(test_path, target_size=(299,299), classes=class_labels,
batch_size=5, shuffle=False)

base_model = keras.applications.xception.Xception(include_top=False)

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(7, activation='softmax')(x)
```

```
for layer in base_model.layers:
    layer.trainable = False

model = Model(inputs=base_model.input, outputs=predictions)



model.summary()

print ("[INFO] successfully loaded base model and model...")
print ("[INFO] training started...")

N=30

model.compile(Adam(lr=.0001), loss='categorical_crossentropy',
metrics=['accuracy'])
history=model.fit_generator(train_batches, steps_per_epoch=412,
validation_data=valid_batches,
                validation_steps=35, epochs=N, verbose=1)

print ("[INFO] evaluating model...")

test_labels=test_batches.classes
predictions=model.predict_generator(test_batches, steps=28, verbose=1)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
```

```
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print ("[INFO] confusion matrix")

cm=confusion_matrix(test_labels, predictions.argmax(axis=1))
```

```
plt.figure()
plot_confusion_matrix(cm, class_labels,normalize=False, title='Confusion Matrix of Xception')
plt.savefig('Confusion_Matrix_of_Xception.png')

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

score=accuracy_score(test_labels, predictions.argmax(axis=1))
print("Rank-1 accuracy: %0.3f" % score)
print(classification_report(test_labels, predictions.argmax(axis=1),
target_names=class_labels))

print ("[INFO] plotting the results...")

plt.figure()
plt.style.use("ggplot")
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.title("Xception Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(loc="upper left")
plt.show()
plt.savefig('Xception_Model_Loss.png')
```

VGG16

```
import numpy as np
import keras
from keras.layers import Dense
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model, Sequential
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

print ("[INFO] program started")

train_path='Skin_DiseasesV2/train'
valid_path='Skin_DiseasesV2/validation'
test_path='Skin_DiseasesV2/test'



class_labels=['Acne',
              'Eczema',
              'Pityriasis rosea',
              'Psoriasis',
              'Tinea corporis',
              'Varicella(Chickenpox)',
              'Vitiligo']

train_batches=ImageDataGenerator(preprocessing_function=keras.applications.vgg16.pr
eprocess_input)\
   .flow_from_directory(train_path, target_size=(224,224), classes=class_labels,
batch_size=5)
```

```
valid_batches=ImageDataGenerator(preprocessing_function=keras.applications.vgg16.preprocess_input)\
   .flow_from_directory(valid_path, target_size=(224,224), classes=class_labels, batch_size=5)
test_batches=ImageDataGenerator(preprocessing_function=keras.applications.vgg16.preprocess_input)\
   .flow_from_directory(test_path, target_size=(224,224), classes=class_labels, batch_size=5, shuffle=False)

vgg16_model=keras.applications.vgg16.VGG16()
model = Sequential()
for layer in vgg16_model.layers:
   model.add(layer)

model.layers.pop()

for layer in model.layers:
   layer.trainable=False

model.add(Dense(7, activation='softmax'))

N=30

model.compile(Adam(lr=.0001), loss='categorical_crossentropy', metrics=['accuracy'])
history=model.fit_generator(train_batches, steps_per_epoch=412,
validation_data=valid_batches,
                validation_steps=35, epochs=N, verbose=1)

print ("[INFO] evaluating model...")
```

```
test_labels=test_batches.classes
predictions=model.predict_generator(test_batches, steps=28, verbose=1)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
```

```
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print ("[INFO] confusion matrix")

cm=confusion_matrix(test_labels, predictions.argmax(axis=1))
plt.figure()
plot_confusion_matrix(cm, class_labels,normalize=False, title='Confusion Matrix of
VGG16')
plt.savefig('Confusion_Matrix_of_VGG16.png')

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

score=accuracy_score(test_labels, predictions.argmax(axis=1))
print("Rank-1 accuracy: %0.3f" % score)
print(classification_report(test_labels, predictions.argmax(axis=1),
target_names=class_labels))

print ("[INFO] plotting the results...")
plt.figure()
plt.style.use("ggplot")
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.title("VGG16 Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
```

```
plt.legend(loc="upper left")
plt.show()
plt.savefig('VGG16_Model_Loss.png')
```

VGG19

```
import numpy as np
import keras
from keras.layers import Dense, GlobalAveragePooling2D
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

print ("[INFO] program started")

train_path='Skin_DiseasesV2/train'
valid_path='Skin_DiseasesV2/validation'
test_path='Skin_DiseasesV2/test'

vgg19_model=keras.applications.vgg19.VGG19()

class_labels=['Acne',
              'Eczema',
```

```
            'Pityriasis rosea',
            'Psoriasis',
            'Tinea corporis',
            'Varicella(Chickenpox)',
            'Vitiligo']

train_batches=ImageDataGenerator(preprocessing_function=keras.applications.vgg19.preprocess_input)\
    .flow_from_directory(train_path, target_size=(224,224), classes=class_labels, batch_size=5)
valid_batches=ImageDataGenerator(preprocessing_function=keras.applications.vgg19.preprocess_input)\
    .flow_from_directory(valid_path, target_size=(224,224), classes=class_labels, batch_size=5)
test_batches=ImageDataGenerator(preprocessing_function=keras.applications.vgg19.preprocess_input)\
    .flow_from_directory(test_path, target_size=(224,224), classes=class_labels, batch_size=5, shuffle=False)

model = Sequential()
for layer in vgg19_model.layers:
    model.add(layer)

model.layers.pop()

for layer in model.layers:
    layer.trainable=False

model.add(Dense(7, activation='softmax'))
model.summary()
```

```
print ("[INFO] successfully loaded base model and model...")
print ("[INFO] training started...")
N=30

model.compile(Adam(lr=.0001), loss='categorical_crossentropy',
metrics=['accuracy'])
history=model.fit_generator(train_batches, steps_per_epoch=412,
validation_data=valid_batches,
                validation_steps=35, epochs=N, verbose=1)

print ("[INFO] evaluating model...")
test_labels=test_batches.classes
predictions=model.predict_generator(test_batches, steps=28, verbose=1)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)
```

```
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print ("[INFO] confusion matrix")

cm=confusion_matrix(test_labels, predictions.argmax(axis=1))
plot_confusion_matrix(cm, class_labels,normalize=False, title='Confusion Matrix of VGG19')
plt.figure()
plt.savefig('Confusion_Matrix_of_VGG19.png')

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

score=accuracy_score(test_labels, predictions.argmax(axis=1))
```

```
print("Rank-1 accuracy: %0.3f" % score)
print(classification_report(test_labels, predictions.argmax(axis=1),
target_names=class_labels))

print ("[INFO] plotting the results...")
plt.figure()
plt.style.use("ggplot")
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.title("VGG19 Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(loc="upper left")
plt.show()
plt.savefig('VGG19_Model_Loss.png')
```

ResNet50

```
import numpy as np
import keras
from keras.layers import Dense, GlobalAveragePooling2D
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

print ("[INFO] program started")
```

```
train_path='Skin_DiseasesV2/train'
valid_path='Skin_DiseasesV2/validation'
test_path='Skin_DiseasesV2/test'

class_labels=['Acne',
              'Eczema',
              'Pityriasis rosea',
              'Psoriasis',
              'Tinea corporis',
              'Varicella(Chickenpox)',
              'Vitiligo']

train_batches=ImageDataGenerator(preprocessing_function=keras.applications.resnet50.
preprocess_input)\
    .flow_from_directory(train_path, target_size=(224,224), classes=class_labels,
batch_size=5)
valid_batches=ImageDataGenerator(preprocessing_function=keras.applications.resnet50.
preprocess_input)\
    .flow_from_directory(valid_path, target_size=(224,224), classes=class_labels,
batch_size=5)
test_batches=ImageDataGenerator(preprocessing_function=keras.applications.resnet50.p
reprocess_input)\
    .flow_from_directory(test_path, target_size=(224,224), classes=class_labels,
batch_size=5, shuffle=False)

base_model = keras.applications.resnet50.ResNet50(include_top=False)

x = base_model.output
x = GlobalAveragePooling2D()(x)
```

```
x = Dense(1024, activation='relu')(x)
predictions = Dense(7, activation='softmax')(x)

for layer in base_model.layers:
    layer.trainable = False

model = Model(inputs=base_model.input, outputs=predictions)
model.summary()

print ("[INFO] successfully loaded base model and model...")
print ("[INFO] training started...")
N=30

model.compile(Adam(lr=.0001), loss='categorical_crossentropy',
metrics=['accuracy'])
history=model.fit_generator(train_batches, steps_per_epoch=412,
validation_data=valid_batches,
                validation_steps=35, epochs=N, verbose=1)
print ("[INFO] evaluating model...")

test_labels=test_batches.classes
predictions=model.predict_generator(test_batches, steps=28, verbose=1)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
```

```
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print ("[INFO] confusion matrix")

cm=confusion_matrix(test_labels, predictions.argmax(axis=1))
plt.figure()
```

```
plot_confusion_matrix(cm, class_labels,normalize=False, title='Confusion Matrix of ResNet50')
plt.savefig('Confusion_Matrix_of_ResNet50.png')

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

score=accuracy_score(test_labels, predictions.argmax(axis=1))
print("Rank-1 accuracy: %0.3f" % score)
print(classification_report(test_labels, predictions.argmax(axis=1),
target_names=class_labels))

print ("[INFO] plotting the results...")

plt.figure()
plt.style.use("ggplot")
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.title("ResNet50 Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(loc="upper left")
plt.show()
plt.savefig('ResNet50_Model_Loss.png')
```

InceptionV3

```
import numpy as np
import keras
from keras.layers import Dense, GlobalAveragePooling2D
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

print ("[INFO] program started")

train_path='Skin_DiseasesV2/train'
valid_path='Skin_DiseasesV2/validation'
test_path='Skin_DiseasesV2/test'

class_labels=['Acne',
              'Eczema',
              'Pityriasis rosea',
              'Psoriasis',
              'Tinea corporis',
              'Varicella(Chickenpox)',
              'Vitiligo']

train_batches=ImageDataGenerator(preprocessing_function=keras.applications.inception
_v3.preprocess_input)\
   .flow_from_directory(train_path, target_size=(299,299), classes=class_labels,
batch_size=5)
valid_batches=ImageDataGenerator(preprocessing_function=keras.applications.inception
_v3.preprocess_input)\
```

```
    .flow_from_directory(valid_path, target_size=(299,299), classes=class_labels,
batch_size=5)
test_batches=ImageDataGenerator(preprocessing_function=keras.applications.inception_
v3.preprocess_input)\
    .flow_from_directory(test_path, target_size=(299,299), classes=class_labels,
batch_size=5, shuffle=False)

base_model = keras.applications.inception_v3.InceptionV3(include_top=False)

x = model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(7, activation='softmax')(x)

for layer in base_model.layers:
    layer.trainable = False

model = Model(inputs=base_model.input, outputs=predictions)
model.summary()

print ("[INFO] successfully loaded base model and model...")
print ("[INFO] training started...")
N=30

model.compile(Adam(lr=.0001), loss='categorical_crossentropy',
metrics=['accuracy'])
history=model.fit_generator(train_batches, steps_per_epoch=412,
validation_data=valid_batches,
                validation_steps=35, epochs=N, verbose=1)
print ("[INFO] evaluating model...")
```

```
test_labels=test_batches.classes
predictions=model.predict_generator(test_batches, steps=28, verbose=1)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
```

```
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print ("[INFO] confusion matrix")

cm=confusion_matrix(test_labels, predictions.argmax(axis=1))
plt.figure()
plot_confusion_matrix(cm, class_labels,normalize=False, title='Confusion Matrix of
InceptionV3')
plt.savefig('Confusion_Matrix_of_InceptionV3.png')

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

score=accuracy_score(test_labels, predictions.argmax(axis=1))
print("Rank-1 accuracy: %0.3f" % score)
print(classification_report(test_labels, predictions.argmax(axis=1),
target_names=class_labels))

print ("[INFO] plotting the results...")

plt.figure()
plt.style.use("ggplot")
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.title("InceptionV3 Model Loss")
```

```
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(loc="upper left")
plt.show()
plt.savefig('InceptionV3_Model_Loss.png')
```

InceptionResNetV2

```
import numpy as np
import keras
from keras.layers import Dense, GlobalAveragePooling2D
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

print ("[INFO] program started")
train_path='Skin_DiseasesV2/train'
valid_path='Skin_DiseasesV2/validation'
test_path='Skin_DiseasesV2/test'

class_labels=['Acne',
              'Eczema',
              'Pityriasis rosea',
              'Psoriasis',
              'Tinea corporis',
              'Varicella(Chickenpox)',
```

```
        'Vitiligo']

train_batches=ImageDataGenerator(preprocessing_function=keras.applications.inception
_resnet_v2.preprocess_input)\
   .flow_from_directory(train_path, target_size=(299,299), classes=class_labels,
batch_size=5)
valid_batches=ImageDataGenerator(preprocessing_function=keras.applications.inception
_resnet_v2.preprocess_input)\
   .flow_from_directory(valid_path, target_size=(299,299), classes=class_labels,
batch_size=5)
test_batches=ImageDataGenerator(preprocessing_function=keras.applications.inception_
resnet_v2.preprocess_input)\
   .flow_from_directory(test_path, target_size=(299,299), classes=class_labels,
batch_size=5, shuffle=False)

base_model
=keras.applications.inception_resnet_v2.InceptionResNetV2(include_top=False)

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(7, activation='softmax')(x)

for layer in base_model.layers:
   layer.trainable = False

model = Model(inputs=base_model.input, outputs=predictions)
model.summary()

print ("[INFO] successfully loaded base model and model...")
```

```
print ("[INFO] training started...")
N=30

model.compile(Adam(lr=.0001), loss='categorical_crossentropy',
metrics=['accuracy'])
history=model.fit_generator(train_batches, steps_per_epoch=412,
validation_data=valid_batches,
                validation_steps=35, epochs=N, verbose=1)

print ("[INFO] evaluating model...")
test_labels=test_batches.classes
predictions=model.predict_generator(test_batches, steps=28, verbose=1)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
```

```
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print ("[INFO] confusion matrix")

cm=confusion_matrix(test_labels, predictions.argmax(axis=1))
plt.figure()
plot_confusion_matrix(cm, class_labels,normalize=False, title='Confusion Matrix of InceptionResNetV2')
plt.savefig('Confusion_Matrix_of_InceptionResNetV2.png')

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

score=accuracy_score(test_labels, predictions.argmax(axis=1))
print("Rank-1 accuracy: %0.3f" % score)
print(classification_report(test_labels, predictions.argmax(axis=1),
```

```
target_names=class_labels))

print ("[INFO] plotting the results...")

plt.figure()
plt.style.use("ggplot")
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.title("InceptionResNetV2 Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(loc="upper left")
plt.show()
plt.savefig('InceptionResNetV2_Model_Loss.png')
```

DenseNet121

```
import numpy as np
import keras
from keras.layers import Dense, GlobalAveragePooling2D
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

print ("[INFO] program started")
```

```
train_path='Skin_DiseasesV2/train'
valid_path='Skin_DiseasesV2/validation'
test_path='Skin_DiseasesV2/test'

class_labels=['Acne',
              'Eczema',
              'Pityriasis rosea',
              'Psoriasis',
              'Tinea corporis',
              'Varicella(Chickenpox)',
              'Vitiligo']

train_batches=ImageDataGenerator(preprocessing_function=keras.applications.densenet.
preprocess_input)\
    .flow_from_directory(train_path, target_size=(224,224), classes=class_labels,
batch_size=5)
valid_batches=ImageDataGenerator(preprocessing_function=keras.applications.densenet.
preprocess_input)\
    .flow_from_directory(valid_path, target_size=(224,224), classes=class_labels,
batch_size=5)
test_batches=ImageDataGenerator(preprocessing_function=keras.applications.densenet.p
reprocess_input)\
    .flow_from_directory(test_path, target_size=(224,224), classes=class_labels,
batch_size=5, shuffle=False)

base_model = keras.applications.densenet.DenseNet121(include_top=False,
input_shape=(224,224,3))

x = base_model.output
```

```
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(7, activation='softmax')(x)

for layer in base_model.layers:
    layer.trainable = False

model = Model(inputs=base_model.input, outputs=predictions)

print ("[INFO] successfully loaded base model and model...")
print ("[INFO] training started...")
N=30

model.compile(Adam(lr=.0001), loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit_generator(train_batches, steps_per_epoch=412, validation_data=valid_batches,
                    validation_steps=35, epochs=N, verbose=1)
print ("[INFO] evaluating model...")

test_labels=test_batches.classes
predictions=model.predict_generator(test_batches, steps=28, verbose=1)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
```

```
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print ("[INFO] confusion matrix")

cm=confusion_matrix(test_labels, predictions.argmax(axis=1))
plt.figure()
plot_confusion_matrix(cm, class_labels,normalize=False, title='Confusion Matrix of
```

```
DenseNet121')
plt.savefig('Confusion_Matrix_of_DenseNet121.png')

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

score=accuracy_score(test_labels, predictions.argmax(axis=1))
print("Rank-1 accuracy: %0.3f" % score)
print(classification_report(test_labels, predictions.argmax(axis=1),
target_names=class_labels))

print ("[INFO] plotting the results...")

plt.figure()
plt.style.use("ggplot")
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.title("DenseNet121 Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(loc="upper left")
plt.show()
plt.savefig('DenseNet121_Model_Loss.png')
```

DenseNet169

```
import numpy as np
import keras
```

```
from keras.layers import Dense, GlobalAveragePooling2D
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

print ("[INFO] program started")

train_path='Skin_DiseasesV2/train'
valid_path='Skin_DiseasesV2/validation'
test_path='Skin_DiseasesV2/test'

class_labels=['Acne',
              'Eczema',
              'Pityriasis rosea',
              'Psoriasis',
              'Tinea corporis',
              'Varicella(Chickenpox)',
              'Vitiligo']

train_batches=ImageDataGenerator(preprocessing_function=keras.applications.densenet.
preprocess_input)\
   .flow_from_directory(train_path, target_size=(224,224), classes=class_labels,
batch_size=5)
valid_batches=ImageDataGenerator(preprocessing_function=keras.applications.densenet.
preprocess_input)\
   .flow_from_directory(valid_path, target_size=(224,224), classes=class_labels,
batch_size=5)
```

```
test_batches=ImageDataGenerator(preprocessing_function=keras.applications.densenet.preprocess_input)\
    .flow_from_directory(test_path, target_size=(224,224), classes=class_labels, batch_size=5, shuffle=False)

base_model = keras.applications.densenet.DenseNet169(include_top=False, input_shape=(224,224,3))

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(7, activation='softmax')(x)

for layer in base_model.layers:
    layer.trainable = False

model = Model(inputs=base_model.input, outputs=predictions)
model.summary()

print ("[INFO] successfully loaded base model and model...")
print ("[INFO] training started...")
N=30

model.compile(Adam(lr=.0001), loss='categorical_crossentropy', metrics=['accuracy'])
model.fit_generator(train_batches, steps_per_epoch=412, validation_data=valid_batches,
                    validation_steps=35, epochs=N, verbose=1)

print ("[INFO] evaluating model...")
```

```
test_labels=test_batches.classes
predictions=model.predict_generator(test_batches, steps=28, verbose=1)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
```

```
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print ("[INFO] confusion matrix")

cm=confusion_matrix(test_labels, predictions.argmax(axis=1))
plt.figure()
plot_confusion_matrix(cm, class_labels,normalize=False, title='Confusion Matrix of
DenseNet169')
plt.savefig('Confusion_Matrix_of_DenseNet169.png')

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

score=accuracy_score(test_labels, predictions.argmax(axis=1))
print("Rank-1 accuracy: %0.3f" % score)
print(classification_report(test_labels, predictions.argmax(axis=1),
target_names=class_labels))

print ("[INFO] plotting the results...")

plt.figure()
plt.style.use("ggplot")
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.title("DenseNet169 Model Loss")
plt.xlabel("Epoch")
```

```
plt.ylabel("Loss")
plt.legend(loc="upper left")
plt.show()
plt.savefig('DenseNet169_Model_Loss.png')
```

DenseNet201

```
import numpy as np
import keras
from keras.layers import Dense, GlobalAveragePooling2D
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

print ("[INFO] program started")

train_path='Skin_DiseasesV2/train'
valid_path='Skin_DiseasesV2/validation'
test_path='Skin_DiseasesV2/test'

class_labels=['Acne',
              'Eczema',
              'Pityriasis rosea',
              'Psoriasis',
              'Tinea corporis',
              'Varicella(Chickenpox)',
              'Vitiligo']
```

```
train_batches=ImageDataGenerator(preprocessing_function=keras.applications.densenet.
preprocess_input)\
    .flow_from_directory(train_path, target_size=(224,224), classes=class_labels,
batch_size=5)
valid_batches=ImageDataGenerator(preprocessing_function=keras.applications.densenet.
preprocess_input)\
    .flow_from_directory(valid_path, target_size=(224,224), classes=class_labels,
batch_size=5)
test_batches=ImageDataGenerator(preprocessing_function=keras.applications.densenet.p
reprocess_input)\
    .flow_from_directory(test_path, target_size=(224,224), classes=class_labels,
batch_size=5, shuffle=False)

base_model = keras.applications.densenet.DenseNet201(include_top=False,
input_shape=(224,224,3))

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(7, activation='softmax')(x)

for layer in base_model.layers:
    layer.trainable = False

model = Model(inputs=base_model.input, outputs=predictions)
model.summary()

print ("[INFO] successfully loaded base model and model...")
print ("[INFO] training started...")
```

```
N=1

model.compile(Adam(lr=.0001), loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit_generator(train_batches, steps_per_epoch=412, validation_data=valid_batches,
                    validation_steps=35, epochs=N, verbose=1)
print ("[INFO] evaluating model...")

test_labels=test_batches.classes
predictions=model.predict_generator(test_batches, steps=28, verbose=1)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
```

```
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print ("[INFO] confusion matrix")

cm=confusion_matrix(test_labels, predictions.argmax(axis=1))
plt.figure()
plot_confusion_matrix(cm, class_labels,normalize=False, title='Confusion Matrix of DenseNet201')
plt.savefig('Confusion_Matrix_of_DenseNet201.png')

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

score=accuracy_score(test_labels, predictions.argmax(axis=1))
print("Rank-1 accuracy: %0.3f" % score)
print(classification_report(test_labels, predictions.argmax(axis=1),
target_names=class_labels))
```

```
print ("[INFO] plotting the results...")

plt.figure()
plt.style.use("ggplot")
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.title("DenseNet201 Model Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(loc="upper left")
plt.show()
plt.savefig('DenseNet201_Model_Loss.png')
```

Heatmap

```
from keras.models import load_model
model=load_model('MobileNetv6.h5')

from keras.preprocessing import image
from keras.applications.mobilenet import preprocess_input
import numpy as np
from keras import backend as K
import matplotlib.pyplot as plt

img_path = 'Skin_diseaseV5/Saliency/Acne3.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
```

x = preprocess_input(x)

MODEL ARCHITECTURE OF MOBILENET

Layer (type)	Output Shape	Param #
input 1 (InputLayer)	(None, 224, 224, 3)	0
conv1 pad (ZeroPadding2D)	(None, 225, 225, 3)	0
conv1 (Conv2D)	(None, 112, 112, 32)	864
conv1 bn (BatchNormalization	(None, 112, 112, 32)	128
conv1 relu (ReLU)	(None, 112, 112, 32)	0
conv dw 1 (DepthwiseConv2D)	(None, 112, 112, 32)	288
conv dw 1 bn (BatchNormaliza	(None, 112, 112, 32)	128
conv dw 1 relu (ReLU)	(None, 112, 112, 32)	0
conv pw 1 (Conv2D)	(None, 112, 112, 64)	2048
conv pw 1 bn (BatchNormaliza	(None, 112, 112, 64)	256
conv pw 1 relu (ReLU)	(None, 112, 112, 64)	0
conv pad 2 (ZeroPadding2D)	(None, 113, 113, 64)	0
conv dw 2 (DepthwiseConv2D)	(None, 56, 56, 64)	576
conv dw 2 bn (BatchNormaliza	(None, 56, 56, 64)	256
conv dw 2 relu (ReLU)	(None, 56, 56, 64)	0
conv pw 2 (Conv2D)	(None, 56, 56, 128)	8192
conv pw 2 bn (BatchNormaliza	(None, 56, 56, 128)	512
conv pw 2 relu (ReLU)	(None, 56, 56, 128)	0

conv_dw_3 (DepthwiseConv2D) (None, 56, 56, 128) 1152

conv_dw_3_bn (BatchNormaliza (None, 56, 56, 128) 512

conv_dw_3_relu (ReLU) (None, 56, 56, 128) 0

conv_pw_3 (Conv2D) (None, 56, 56, 128) 16384

conv_pw_3_bn (BatchNormaliza (None, 56, 56, 128) 512

conv_pw_3_relu (ReLU) (None, 56, 56, 128) 0

conv_pad_4 (ZeroPadding2D) (None, 57, 57, 128) 0

conv_dw_4 (DepthwiseConv2D) (None, 28, 28, 128) 1152

conv_dw_4_bn (BatchNormaliza (None, 28, 28, 128) 512

conv_dw_4_relu (ReLU) (None, 28, 28, 128) 0

conv_pw_4 (Conv2D) (None, 28, 28, 256) 32768

conv_pw_4_bn (BatchNormaliza (None, 28, 28, 256) 1024

conv_pw_4_relu (ReLU) (None, 28, 28, 256) 0

conv_dw_5 (DepthwiseConv2D) (None, 28, 28, 256) 2304

conv_dw_5_bn (BatchNormaliza (None, 28, 28, 256) 1024

conv_dw_5_relu (ReLU) (None, 28, 28, 256) 0

conv_pw_5 (Conv2D) (None, 28, 28, 256) 65536

conv_pw_5_bn (BatchNormaliza (None, 28, 28, 256) 1024

conv_pw_5_relu (ReLU) (None, 28, 28, 256) 0

conv_pad_6 (ZeroPadding2D) (None, 29, 29, 256) 0

conv_dw_6 (DepthwiseConv2D) (None, 14, 14, 256) 2304

conv_dw_6_bn (BatchNormaliza (None, 14, 14, 256) 1024

conv_dw_6_relu (ReLU) (None, 14, 14, 256) 0

conv_pw_6 (Conv2D) (None, 14, 14, 512) 131072

conv_pw_6_bn (BatchNormaliza (None, 14, 14, 512) 2048

conv_pw_6_relu (ReLU) (None, 14, 14, 512) 0

conv_dw_7 (DepthwiseConv2D) (None, 14, 14, 512) 4608

conv_dw_7_bn (BatchNormaliza (None, 14, 14, 512) 2048

conv_dw_7_relu (ReLU) (None, 14, 14, 512) 0

conv_pw_7 (Conv2D) (None, 14, 14, 512) 262144

conv_pw_7_bn (BatchNormaliza (None, 14, 14, 512) 2048

conv_pw_7_relu (ReLU) (None, 14, 14, 512) 0

conv_dw_8 (DepthwiseConv2D) (None, 14, 14, 512) 4608

conv_dw_8_bn (BatchNormaliza (None, 14, 14, 512) 2048

conv_dw_8_relu (ReLU) (None, 14, 14, 512) 0

conv_pw_8 (Conv2D) (None, 14, 14, 512) 262144

conv_pw_8_bn (BatchNormaliza (None, 14, 14, 512) 2048

conv_pw_8_relu (ReLU) (None, 14, 14, 512) 0

conv_dw_9 (DepthwiseConv2D) (None, 14, 14, 512) 4608

conv_dw_9_bn (BatchNormaliza (None, 14, 14, 512) 2048

conv_dw_9_relu (ReLU) (None, 14, 14, 512) 0

conv_pw_9 (Conv2D) (None, 14, 14, 512) 262144

conv_pw_9_bn (BatchNormaliza (None, 14, 14, 512) 2048

conv_pw_9_relu (ReLU) (None, 14, 14, 512) 0

conv_dw_10 (DepthwiseConv2D) (None, 14, 14, 512) 4608

conv_dw_10_bn (BatchNormaliz (None, 14, 14, 512) 2048

conv_dw_10_relu (ReLU) (None, 14, 14, 512) 0

conv_pw_10 (Conv2D) (None, 14, 14, 512) 262144

conv_pw_10_bn (BatchNormaliz (None, 14, 14, 512) 2048

conv_pw_10_relu (ReLU) (None, 14, 14, 512) 0

conv_dw_11 (DepthwiseConv2D) (None, 14, 14, 512) 4608

conv_dw_11_bn (BatchNormaliz (None, 14, 14, 512) 2048

conv_dw_11_relu (ReLU) (None, 14, 14, 512) 0

conv_pw_11 (Conv2D) (None, 14, 14, 512) 262144

conv_pw_11_bn (BatchNormaliz (None, 14, 14, 512) 2048

conv_pw_11_relu (ReLU) (None, 14, 14, 512) 0

conv_pad_12 (ZeroPadding2D) (None, 15, 15, 512) 0

conv_dw_12 (DepthwiseConv2D) (None, 7, 7, 512) 4608

conv_dw_12_bn (BatchNormaliz (None, 7, 7, 512) 2048

conv_dw_12_relu (ReLU) (None, 7, 7, 512) 0

conv_pw_12 (Conv2D) (None, 7, 7, 1024) 524288

conv_pw_12_bn (BatchNormaliz (None, 7, 7, 1024) 4096

conv_pw_12_relu (ReLU) (None, 7, 7, 1024) 0

conv_dw_13 (DepthwiseConv2D) (None, 7, 7, 1024) 9216

conv dw 13 bn (BatchNormaliz (None, 7, 7, 1024) 4096

conv dw 13 relu (ReLU) (None, 7, 7, 1024) 0

conv pw 13 (Conv2D) (None, 7, 7, 1024) 1048576

conv pw 13 bn (BatchNormaliz (None, 7, 7, 1024) 4096

conv pw 13 relu (ReLU) (None, 7, 7, 1024) 0

global average pooling2d 1 ((None, 1024) 0

dense 1 (Dense) (None, 7) 7175

==========

Total params: 3,236,039
Trainable params: 3,214,151
Non-trainable params: 21,888

APPENDIX E

Specifications/Datasheet

Samsung Galaxy J2 Prime Specification

General

- **Available Status:** Available. Released 2016, November
- **Network Bands:** 2G & 3G & 4G
- **SIM:** Single SIM (Micro-SIM) or Dual SIM (Micro-SIM, dual stand-by)
- **Dimension:** 144.8 x 72.1 x 8.9 mm
- **Weight:** 160 g (5.64 oz)
- **OS:** Android OS, v6.0 (Marshmallow)
- **CPU:** Quad-core 1.4 GHz Cortex-A53
- **Chipset:** Mediatek MT6737T
- **GPU:** Mali-T720MP2
- **Internal:** 8 GB, 1.5 GB RAM
- **External Memory:** microSD, up to 256 GB (dedicated slot)
- **Battery:** Removable Li-Ion 2600 mAh battery
- **Colors:** Black, Gold

Connectivity

- **GPRS:** Yes
- **EDGE:** Yes
- **3G/4G Speed:** HSPA 42.2/5.76 Mbps, LTE Cat4 150/50 Mbps
- **Wi-Fi:** Wi-Fi 802.11 b/g/n, Wi-Fi Direct, hotspot
- **Bluetooth:** v4.2, A2DP, LE
- **USB:** microUSB v2.0, USB On-The-Go

Entertainment & Features

- **Sensor:** Accelerometer, proximity
- **GPS:** Yes, with A-GPS
- **Messaging:** SMS(threaded view), MMS, Email, Push Mail, IM
- **Radio:** FM radio, RDS, recording
- **Browser:** HTML
- **Java:** No
- - MP4/H.264 player
  - MP3/WAV/eAAC+/Flac player
  - Photo/video editor
  - Document viewer

Camera & Video

- **Primary:** 8 MP, autofocus, LED flash
- **Secondary:** 5 MP, LED flash
- **Video:** 720p@30fps
- **Feature:** Geo-tagging, touch focus, face detection

Screen

- Capacitive touchscreen, 16M colors
- 5.0 inches (~66.0% screen-to-body ratio) 540 x 960 pixels (~220 ppi pixel density)
- **Multi Touch:** Yes

Sound & Music

- **Alert Types:** Vibration; MP3, WAV ringtones
- **Loud Speaker:** Voice 67dB / Noise 64dB / Ring 75dB
- **3.5mm jack:** Yes
- **Radio:** FM radio, RDS, recording

***Features and specifications are subject to change without prior notification.**

The Specs Sheet

	Asus ROG GL552VW
Screen	15.6 inch, 1920 x 1080 px resolution, matte, IPS, non-touch
Processor	Intel Skylake Core i7-6700HQ CPU, quad-core 2.6 GHz (3.5 GHz TBoost)
Video	Integrated Intel HD 530 + Nvidia GTX 960M 2GB
Memory	16 GB DDR4 2133Mhz (2xDIMMs)
Storage	128 GB M.2 SSD (Samsung MZNLF128HCHP) + 1 TB 2.5″ HDD (Hitachi HTS5410)
Connectivity	Wireless AC Intel 7265 , Gigabit Lan, Bluetooth 4.0
Ports	4x USB 3.0, 1x USB 3.1, HDMI, mic, earphone, SD card reader, LAN
Baterry	48 Wh (external – not encased)
Operating system	Windows 10
Size	384 x 257 x 34.7 mm (15.1” x 10.1” x 1.36”)
Weight	2.57 kg or 5.66 lb
Extras	backlit keyboard with macro keys, webcam, optical drive

APPENDIX F

Project Documentation

Project Documentation

1. Staffs Coordinated

1.1. Ospital ng Maynila

Dr. Benedicto dL Carpio, MD, FPDS

Chairperson

Department of Dermatology

Ospital ng Maynila Medical Center

Malate, Manila

Dr. Mary Grace A. Calvarido, MD

Former Chief Resident

Department of Dermatology

Ospital ng Maynila Medical Center

Malate, Manila

1.2. PNP General Hospital (Camp Crame)

PSSUPT REIMOUND C. SALES

Chief

PNP General Hospital, Health Service.

Camp Crame, Q.C

1.3 Sucat Health Center

Dra. Rowena R. Tan

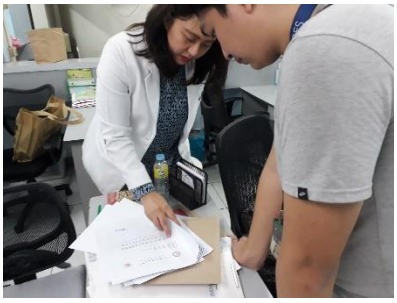
General Practitioner

Sucat Health Center

Muntinlupa City

2. Documentation

Interview with the DOH



Datasets validation

Data gathering

Deployment

Testing of device

Proponets' defenses

APPENDIX G

Proponent's Profile

JEAN WILMAR G. ALBERIO

183-N 25th Avenue, East Rembo, Makati City, Metro Manila
(+63)917 447 6392 dbshop250@gmail.com

OBJECTIVES

- To acquire a challenging career with a solid company utilizing the opportunity to offer proven and developing skills within the company.

- Pursuing opportunity which will allow me to grow professionally, while effectively utilizing my skill set to help promote your corporate mission and exceed team goals.

EDUCATIONAL QUALIFICATIONS

Tertiary 2014 – present	Technological University of the Philippines Ayala Blvd, Ermita, Manila ***Bachelor of Science in Electronics and Communications Engineering***
Secondary 2010 – 2014	La Consolacion College Manila 8 Mendiola St, San Miguel, Manila, 1005 Metro Manila
Primary 2004 – 2010	East Rembo Elementary School J.P. Rizal Ext, Makati, Metro Manila

TECHNICAL SKILLS

- Basic Electrical and Electronics
- Proficient knowledge in Windows Applications (e.g. Microsoft Office)
- Basic Video Editing
  - Final Cut Pro X
  - Adobe Premiere Pro
  - Movavi
- Basic Image Editing
  - Adobe Photoshop
- Knowledge on:
  - NI MultiSim
  - ExpressPCB
  - MatLab
  - Octave
  - Java
  - HTML
  - C++
  - Arduino IDE
- Assembling consumer electronic product, soldering, troubleshooting, and 8D Analysis.

ACHIEVEMENTS AND SEMINARS

- CHED Scholar (Tulong Dunong Scholarship) (2017-present)
- Cryptocurrency and Blockchain Technology 101 Seminar

PERSONAL DETAILS

Age: 20 years old	Citizenship: Filipino	Height: 5’9’’
Birthdate: March 5, 1998	Civil Status: Single	Weight: 192 lbs

CHARACTER REFERENCES:

Engr. Jessica S. Velasco
Professor
(TUP-Manila)

09391615358

Engr. Lean Karlo S. Tolentino
Professor
(TUP-Manila)

09958925845

JONATHAN G. APUANG

Unit 411, Apex Bldg. Arista Place Condominium, J.P. Rizal St., Brgy. Sto. Niño, Parañaque City

(+63)965 103 8204 jonathan.apuang@tup.edu.ph

OBJECTIVE

- To grow capably by giving my skills to association and vice versa.
- To utilize my knowledge and skills in the top likely way for the implementation of organizational goals.

EDUCATIONAL QUALIFICATIONS

Tertiary 2014 – present	Technological University of the Philippines Ayala Blvd, Ermita, Manila ***Bachelor of Science in Electronics and Communications Engineering***
Secondary 2010 – 2014	St. Joseph's Academy Las Piñas City, Philippines
Primary 2007-2010	St. Joseph's Academy Las Piñas City, Philippines
2004 – 2007	Devine Light Academy Las Piñas City, Philippines

TECHNICAL SKILLS

- Software
  - Adobe Photoshop
  - Microsoft Word
  - Microsoft Visual Studio
  - Lego Mindstorms
  - Natepad
  - Windows Movie Maker
  - Lego NXT
- Programming
  - HTML
  - CSS+
  - Dreamweaver
  - Adobe Flash
  - Visual Basic
  - Visual Studio

ACHIEVEMENTS AND SEMINARS

- Certificate of Achievement for completing *TechnoStore* computer curriculum and obtaining the skills to create systems to simplify day-to-day tasks by programming with *Visual Basic* and *Visual Studio (2012-2013)*
- 2012-2013 Certificate of Achievement for completing the *HTML, CSS, and Dreamweaver* computer curriculum and obtaining skills to develop web pages(2012-2013)
- Award of Excellence for being the over-all *rank 12* of the 2nd Year level for the Second Grading Period and *ranked 2nd* in the class of the same period (2011-2012)

PERSONAL DETAILS

Age: 20 years old	Citizenship: Filipino	Height: 5'7''
Birthdate: September 5, 1997	Civil Status: Single	Weight: 132 lbs

CHARACTER REFERENCES:

Engr. Jessica S. Velasco
Professor
(TUP-Manila)

09391615358

Engr. Lean Karlo S. Tolentino
Professor
(TUP-Manila)

09958925845

JOHN STEPHEN B. CRUZ

7685 B J.B. Roxas Street, Barangay Olympia, Makati City, Metro Manila

(+63)926 575 3069 johnstephcruz@gmail.com

OBJECTIVES

- To further my experience and knowledge in the field of electronics
- To acquire a challenging career with a solid company utilizing the opportunity to offer proven and developing skills within the company.

EDUCATIONAL QUALIFICATIONS

Tertiary 2012 – present	Technological University of the Philippines Ayala Blvd, Ermita, Manila ***Bachelor of Science in Electronics and Communications Engineering***
Secondary 2009 – 2012	Gen. Pio Del Pilar National High School Osias, Makati City, Metro Manila
Primary 2002-2008	Nicanor Garcia Elementary School 7712 J.B. Roxas, Makati City, 1207 Metro Manila

SKILLS

- Able to work with limited supervision.
- Ability to balance workloads efficiently. Analytical, and able to work in a constantly changing work environment
- Ability to maintain of confidentially in handling sensitive information
- Team player

PERSONAL DETAILS

Age: 22 years old	Citizenship: Filipino	Height: 5’8’’
Birthdate: December 29, 1995	Civil Status: Single	Weight: 132 lbs

CHARACTER REFERENCES:

Engr. Jessica S. Velasco *Professor* *(TUP-Manila)*	Engr. Lean Karlo S. Tolentino *Professor* *(TUP-Manila)*	Estrella General *Customs Representative* *325 JP Rizal Tejeros Makati*
09391615358	09958925845	09472246092

MARK ANGELO B. GOMEZ

700 Amethyst Street, Palmera 2 Subd., Brgy. San Isidro, Taytay, Rizal

(+63)9552102794 gelo022897@yahoo.com

OBJECTIVE

To improve myself to engage in any kind of job, and to enhance my skills in any aspects to became a better one.

EDUCATIONAL QUALIFICATIONS

Tertiary 2013 – present	Technological University of the Philippines Ayala Blvd, Ermita, Manila ***Bachelor of Science in Electronics and Communications Engineering***
Secondary 2009 – 2013	Sumulong Memorial High School General Luna St, Antipolo, 1870 Rizal
Primary 2003 – 2009	Glendale Immanuel Academy Glendale Circle St., Glendale Subd., Taytay, Dolores, 1920, Rizal

TECHNICAL SKILLS

- Basic Electrical and Electronics
- Proficient knowledge in Windows Applications (e.g. Microsoft Office)
- Basic Video Editing (Sony Vegas Pro, Power Director)
- Basic knowledge on (NI Multisim, MATLAB, and JAVA)

EXPERIENCE

July 1, 2017 – August 15, 2017	Service Crew (Part Time) Jollibee Food Corporation - Harrison Plaza JOYMAN (Kitchen)
October 17, 2017 – January 17,2018	Service Crew (Part Time) Jollibee Food Corporation – EAC-Gen. Luna FRYMAN(Kitchen)

ACHIEVEMENTS AND SEMINARS

- Elementary Class Valedictorian (2009)
- Metrobank MTAP Participant (2011)
- "Expanding Electronics Engineers' Horizon" TUP Manila – July 23,2013
- "Young Entrepreneurs' Sponsorship Program" TUP Manila – September 30,2013
- "Bakit ECE? Eh meron namang iba" TUP Manila – March 5, 2014
- "HOOKE: Heading Over Opportunities & Keeping Engaged" San Andres Sports Complex – August 14, 2016
- "Seminars and Fieldtrip 2018 – TUP" Angeles, Pampanga – January 22, 2018
- "Developing Leaders, Building Nations" UP Diliman – January 27, 2018
- "COalascE: Opposing Factions Uniting for a Common End" TUP Manila – February 27, 2018
- "Cryptocurrency and Blockchain Technology 101" TUP Manila – February 28, 2018

PERSONAL DETAILS

Age: 21 years old	Citizenship: Filipino	Height: 5'4''
Birthdate: February 28, 1997	Civil Status: Single	Weight: 154 lbs

CHARACTER REFERENCES:

Engr. Jessica S. Velasco
Professor
(TUP-Manila)

09391615358

Engr. Lean Karlo S. Tolentino
Professor
(TUP-Manila)

09958925845

BENJAMIN V. MOLINA Jr.

Block 6, Lot 22, Villa Susana Subd., Brgy. Malitlit, Sta. Rosa ,Laguna

(+63)977 036 0530 benjiemolina15@gmail.com

OBJECTIVES

To utilize my skills and abilities in the field of electronics engineering industry that offers professional growth while being resourceful, innovative and flexible as well as to practice my leadership skills and decision-making skills.

EDUCATIONAL QUALIFICATIONS

Tertiary 2011 – present	Technological University of the Philippines Ayala Blvd, Ermita, Manila ***Bachelor of Science in Electronics and Communications Engineering***
Secondary 2006 – 2011	Camiling Catholic School Arellano Street, Camiling, Tarlac
Primary 2000 – 2006	Malacampa Elementary School Brgy. Malacampa, Camiling, Tarlac

TECHNICAL SKILLS

- Programing
  - C,
  - Basic Java/Java Script
  - HTML
- Software
  - Microsoft Office (Word, Excel, and Powerpoint)
  - Multisim
  - MATLAB
  - Eclipse

ACHIEVEMENTS AND SEMINARS

- Engineering Innovations Congress 2018: Social Entrepreneurial Engineering Symposium, UP Diliman - January 27, 2018
- APPRECIATE: Annual Presentation of Project Research in Research in Electromechanical, Civil, Information Telecommunications Engineering TUP Manila - March 3, 2017
- Trends in Analog and Digital IC with Technopreneurship, TUP Manila - February 9, 2016
- Road to Semiconductor Industry, TUP Manila -January 29, 2016
- Latest Trends in MicroChip 8-bit Microcontrollers, TUP Manila – January 22, 2016
- The Open Source Technology, TUP Manila - February 5, 2016

PERSONAL DETAILS

Age: 23 years old	Citizenship: Filipino	Height: 5’7’’
Birthdate: October 15, 1994	Civil Status: Single	Weight: 150 lbs

CHARACTER REFERENCES:

Engr. Jessica S. Velasco
Professor
(TUP-Manila)

09391615358

Engr. Lean Karlo S. Tolentino
Professor
(TUP-Manila)

09958925845

LYNDON T. TUALA

131 B11 L6 PH1 Urbanville, Talon 3, Las Piñas City

(+63)920 202 0036 lyndontuala28@gmail.com

OBJECTIVES

- To be able to work in a highly organized environment and contribute my expertise and knowledge in electronic processes and applications such as designing, testing, troubleshooting, and modifying electronic equipments.

EDUCATIONAL QUALIFICATIONS

Tertiary 2014 – present	Technological University of the Philippines Ayala Blvd, Ermita, Manila ***Bachelor of Science in Electronics and Communications Engineering***
Secondary 2010 – 2014	Las Piñas Science High School Carnival Park St., BFRV, Brgy. Talon 2, Las Piñas City
Primary 2004 – 2010	Almanza Elementary School Real Street, Almanza Uno, Las Piñas City

TECHNICAL SKILLS

- Basic Electrical and Electronics
- Knowledge on:
  - NI MultiSim
  - ExpressPCB
  - MatLab
  - Octave
  - Java
  - C++
  - Arduino IDE
- Assembling consumer electronic product, soldering, troubleshooting, and 8D Analysis.

ACHIEVEMENTS AND SEMINARS

- Satellite Communication Application, APPPRECIATE 2018 – TUP Manila (February 2018)
- Cryptocurrency and Blockchain Technology 101 Seminar – TUP Manila (February 2018)

PERSONAL DETAILS

Age: 20 years old	Citizenship: Filipino	Height: 5’6’’
Birthdate: November 28, 1997	Civil Status: Single	Weight: 154 lbs

CHARACTER REFERENCES:

Engr. Jessica S. Velasco *Professor* *(TUP-Manila)*	Engr. Lean Karlo S. Tolentino *Professor* *(TUP-Manila)*
09391615358	09958925845

REFERENCES

Abu, et al. *A Model for Classification of Skin Lesions using Image Processing Techniques and Neural Network*. 2017.

Abuzaghleh, et al. *SKINcure: An Innovative smart phone-based application to assist in melanoma early detection and prevention*. Retrived 2018.

Ajith, et al. *Digital dermatology: Skin disease detection model using image processing*. 2017.

Ali, et al. *360° view camera based visual assistive technology for contextual scene information*. 2017.

Alquran, et al. *The melanoma skin cancer detection and classification using support vector machine*. 2017.

Aruta, et al. *Mobile-based Medical Assistance for Diagnosing Different Types of Skin Diseases Using Case-based Reasoning with Image Processing*. 2015.

Chaturvedi, et al. *Severity Grading of Psoriatic Plaques using Deep CNN based Multi-Task Learning*. 2016.

Cole, Gary W. "Eczema." (2018): https://www.medicinenet.com/eczema_facts/article.htm#what_is_the_treatment_for_eczema.

Davis, Charles Patrick. "Leprosy." (2017): https://www.medicinenet.com/leprosy/article.htm#what_is_the_history_of_leprosy_hansen#39s_disease.

De Guzman, et al. *Design and Evaluation of a Multi-model, Multi-level Artificial Neural Network for Eczema Skin Lesion Detection*. 2015.

Department of Health. "Annual Data Report." (2015): www.doh.gov.ph.

—. "Annual Data Report." (2007): www.doh.gov.ph.

—. "Leprosy Control and the Burden of Leprosy in the Philippines." (2010).

Dineshkumar,et al. *Big data analytics of IoT based Health care monitoring system*. 2016.

Elmahdy, et. al. *Low Quality Dermal Image Classification Using Transfer Learning*. Retrived 2018.

Forbes. "Big Data." (2015): https://www.forbes.com/sites/bernardmarr/2015/04/21/how-big-data-is-changing-healthcare/#22f61c652873.

Gonzales, R. *Digital Image Processing*. 2008.

Hay, R. "Skin Diseases. In Disease Control Priorities in Developing Countries. 2nd ed." (2006): Retrieved from https://www.ncbi.nlm.nih.gov/books/NBK11733/#.

K. Poot. "Doctor–patient relations in dermatology: obligations and rights for a mutual satisfaction." *European Academy of Dermatology and Venereology* (2009).

Kim,et al. *Driver distraction detection using single convolutional neural network*. 2017.

Kumar,et al. *A convolutional neural network for visual object recognition in marine sector*. 2017.

Macatangay, et al. *A Primary Morphological Classifier for Skin Lesion Images*. Retrieved 2018.

Mathworks. "Machine Learning." (2018): https://www.mathworks.com/discovery/machine-learning.html#howitworks.

MedDeviceOnline. "Point of Care Technology." (2014): https://www.meddeviceonline.com/doc/things-you-need-to-know-about-the-point-of-care-technology-market-0001.

National Center for Disease Prevention and Control. *Leprosy*. Manila: Department of Health, 2017.

NCBI. "Big Data." (2014): https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4341817/.

PAG-ASA. "Philippine Weather Data." (2018): www.pagasa.dost.gov.ph.

Sharma. *Artificial Neural Network*. 2017.

Shavlik and Torrey. *Machine Learning*. Retrieved 2018.

Stanford. "Convolutional Neural Network." (Retrived 2018): http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/.

Sybingco, et al. *Malignancy Detection of Candidate for Basal Cell Carcinoma Using Image Processing and Artificial Neural Network*. 2007.

TechTarget. "Android Studio." (2015): http://searchsoftwarequality.techtarget.com/feature/Learn-more-about-the-Android-Studio-IDE-from-Google.

—. "Big Data." (2016): http://searchcloudcomputing.techtarget.com/definition/big-data-Big-Data.

"Transfer Learning." (n.d.): https://machinelearningmastery.com/transfer-learning-for-deep-learning/.

Velasco, et al. *Urine Sediment Classification using Deep Learning*. n.d.

Vinhal, et al. *Fully convolutional networks for segmenting images from an embedded camera*. 2017.

Wikipedea. "Convolutional Neural Network." (2018): https://en.wikipedia.org.

Wikipedia. "Acne." (2018): http://en.wikipedia.org.

—. "Deep Learning." (n.d.): wikipedia.org.

—. "Deep Learning." (2018): https://en.wikipedia.org.

—. "Image Processing." (2018): https://en.wikipedia.org.

—. "Pityriasis Rosea." (2018): https://en.wikipedia.org.

—. "Psoriasis." (2018): http://en.wikipedia.org.

—. "Python Program." (2018): https://en.wikipedia.org.

—. "Tinea Corporis." (n.d.): https://en.wikipedia.org.

World Life Expectancy. "Skin Disease Deaths in Philippines." (2014): http://www.worldlifeexpectancy.com.

Xia,et al. *Inception-v3 for flower classification*. Retrived 2018.

Xiao,et al. *Scene classification with improved AlexNet model*. 2017.

Zhang, et al. *Computer-aided Diagnosis of Four Common Cutaneous Diseases Using Deep Learning Algorithm*. 2017.

Zhang,et al. *PolyNet: A Pursuit of Structural Diversity in Very Deep Networks*. 2017.

Zhou, et. al. *Multi-classification of skin diseases for dermoscopy images using deep learning*. 2017.