TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

Kathmandu Engineering College

Department of Computer Engineering

Minor Project Final Report

On

Skin Cancer Detection and Classification Using Transfer Learning

[Code No: CT654]

By

Prerana Shrestha (KAT077BCT051)

Rinal Paudyal (KAT077BCT059)

Shashank Karki (KAT077BCT076)

Shreeya Pant (KAT077BCT077)

Kathmandu, Nepal

Baisakh, 2081

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Kathmandu Engineering College

Department of Computer Engineering

Skin Cancer Detection and Classification Using Transfer Learning

[Code No: CT654]

PROJECT REPORT SUBMITTED TO

THE DEPARTMENT OF COMPUTER ENGINEERRING

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE BACHELOR OF ENGINEERING

By

Prerana Shrestha (KAT077BCT051)

Rinal Paudyal (KAT077BCT059)

Shashank Karki (KAT077BCT076)

Shreeya Pant (KAT077BCT077)

Kathmandu, Nepal

Baisakh, 2081

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Kathmandu Engineering College

Department of Computer Engineering

CERTIFICATE

The undersigned certify that they have read and recommended to the Department of Computer Engineering a minor project work entitled “Skin Cancer Detection and Classification using Transfer Learning” submitted by Prerana Shrestha - 77051, Rinal Paudyal - 77059, Shashank Karki - 77076, Shreeya Pant - 77077 in partial fulfillment of the requirements for the degree of Bachelor of Engineering.

Er. Ganesh Kumal (External Examiner) Department of Electronics and Computer Engineering IOE, Thapathali Campus	Er. Nabin Neupane (Project Supervisor) Department of Computer Engineering Kathmandu Engineering College
Er. Krista Byanju (Project Coordinator) Department of Computer Engineering Kathmandu Engineering College	Er. Sudeep Shakya (Head Of Department) Department of Computer Engineering Kathmandu Engineering College

ACKNOWLEDGEMENT

We must acknowledge our deep sense of gratitude to everyone who have contributed in this project on **SKIN CANCER DETECTION AND CLASSIFICATION USING TRANSFER LEARNING**. The collective efforts, guidance and unwavering support of various personnel has played an important role in this project. First and foremost, we would like to extend our cordial gratitude towards **Institute of Engineering** for providing us with an opportunity to develop a project under the academic requirement as a minor project.

We are very grateful towards **Er. Nabin Neupane, Project Supervisor**, **Er. Sudeep Shakya, Head of the Department** and **Er. Kunjan Amatya, Deputy Head of Department, Computer Engineering** and for their invaluable guidance, effort and time. Their feedback and insights were very crucial in improving the quality of the project and shaping it towards the right direction. Likewise, we must acknowledge our obligation towards **Er. Anju Khanal** ma'am and **Er. Krista Byanju** ma'am for enlightening us with their expertise and helping us throughout the project.

Moreover, we would like to thank **Department of Computer Engineering, Kathmandu Engineering College** for providing us with an opportunity to explore our true potential and build a project of our interest which has highly helped us to implement the knowledge and skills gained over the years as the minor project.

Lastly, we wish to record our appreciation to all of our friends and family who have directly or indirectly helped us throughout this journey.

ABSTRACT

Skin cancer is one of the potentially life-threatening diseases caused by frequent exposure to ultraviolet radiation that damages the deoxyribonucleic acid of the skin cell, resulting in mutations and abnormalities and eventually cancer formation. The main aim of this project is to detect and classify skin cancer with the application of one of the most common transfer learning methods, the pre-trained EfficientNet model, where image of skin lesion and age, sex, site are given as input and the prediction of the cancer type is given as output. This project deals with visual learning and image processing, and as CNN has been proven to be highly effective, thus the pre-trained CNN model along with image pre-processing and data augmentation has been introduced in this project for the classification of skin images into the respective class of Basal Cell Carcinoma (BCC), Squamous Cell Carcinoma (SCC), melanoma, and benign lesions (non-cancerous). This project targets to enhance the accuracy of diagnosis of skin cancer, encourage early diagnoses, improve patient outcomes, and make health care services more efficient and accessible. The model has been trained on 11097 dermoscopic images of the above mentioned four categories from the training set available as ISIC 2019 in Kaggle and then, with the help of the testing set, the accuracy and loss of the model obtained while classifying the skin cancer types is calculated, which is the measure of the success of this project.

Keywords: *Skin cancer, Classification, CNN, EfficientNet, basal cell carcinoma, squamous cell carcinoma, melanoma, benign lesions, transfer learning*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI:	Artificial Intelligence
ASR:	Age-Standardized rate
AUC-ROC:	Area Under the Receiver Operating Characteristic Curve
BCC:	Basal Cell Carcinoma
BSD:	Berkeley Software Distribution
CNN:	Convolutional Neural Network
CSS:	Cascading Style Sheets
CSV:	Comma Separated Values
DNN:	Deep Neural Network
FN:	False Negative
FP:	False Positive
Grad-CAM:	Gradient-weighted Class Activation Mapping
HAM:	Human Against Machine
HTML:	Hyper-Text Markup Language
ISIC:	International Skin Imaging Collaboration
R-CNN:	Region-based Convolutional Neural Network
ReLU:	Rectified Linear Unit
ROC:	Receiver Operator Characteristic
RGB:	Red, Green and Blue
SCC:	Squamous Cell Carcinoma

TN:	True Negative
TP:	True Positive
UML:	Unified Modeling Language
VGG:	Visual Geometry Group
WHO:	World Health Organization

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND THEORY

The abnormal and uncontrolled division of cells which can invade nearby cells, and tissues and can even further spread to other parts of the body through the blood and lymph system is known as cancer. Skin cancer is one of the most common and life-threatening forms of cancer that affects millions of people worldwide. It is mainly categorized into two parts i.e., melanoma and non-melanoma. Melanoma types of cancers are said to be aggressive. Such type of cancer spread to other parts of the body and sadly, lead to the victim's painful death. It can only be cured if diagnosed early otherwise, they, develop further and the cancer cells multiply themselves in the cells called melanocytes and starts to develop healthy melanocytes which gradually starts to grow out of control resulting in creation of cancerous tumors. Similarly non-melanoma tends to grow slowly and doesn't spread to other parts and is comparatively easily treated. Non-melanoma skin cancers are not exactly life threatening. The most common non-melanoma cancers are basal cell carcinoma and squamous cell carcinoma which are generally seen to be formed in middle and upper layers of the epidermis whereas the most common melanoma cancer is Nodular, Superficial, Lentigo malign and Acral lentiginous melanoma.

According to the latest reports, skin cancer has been observed to increase rapidly over the past decades making it one of the most common type of cancers worldwide. The WHO estimates that around 2-3 million non-melanoma and 132000 melanoma skin cancer occurs worldwide each year. According to World Cancer Research Fund International, the country with the highest rates of melanoma and non-melanoma cancer in 2022 is found to be Australia with ASR of 36.6 of melanoma and ASR of 140 of non-melanoma skin cancer. Likewise, New Zealand has the highest mortality rate followed by Norway due to melanoma and Papua New Guinea has the highest rate of non-melanoma skin cancer mortality in 2020 followed by Namibia [1]. This data helps us to realize that there is an immediate need for early detection and diagnosis of skin cancer. Doctors usually perform the biopsy method for detection which involves removal of samples from a suspected skin lesion for medical examination.

The overall process is quite painful and time consuming. During the recent decades, because of the advancement of technology, significant achievements have been made in the medical sciences like medical image processing techniques, pattern recognition etc.

AI algorithms, particularly CNNs, have been proven to be highly capable in image recognition and classification work which has made the diagnosis of skin cancer symptoms more comfortable, less expensive and speedy. Numerous techniques are proposed for examining and deciding whether the skin lesion is malignant or not. The skin cancer classification system is constructed to make tasks easier for the doctors in the determination and classification of skin cancer which will help in improving the accuracy and efficiency of the detection. Through Transfer Learning approach, image is firstly obtained, preprocessed, followed by augmentation, and then the pre-trained model is used as base model which learns and extracts the necessary features from the available data, the top dense and classification layers are added as per the need and then, finally a model is created. This procedure will remove all the possible errors which can be made by the diagnostics. By using the capability of a pre-trained CNN model to detect skin cancer, it can help to reduce doctor-made errors, and can significantly save time and improve efficiency.

So, our project aims to develop a skin cancer classification system with an approach to increase the efficiency and accuracy of our model to detect skin cancer using various dermoscopic images.

1.2 CANCER TYPES CLASSIFIED BY OUR MODEL

The pre-trained CNN models like EfficientNet are capable of automatically learning relevant features from input images and data, incorporating domain-specific features which provides additional information to improve the classification performance of the model.

So, in short, the features that our model learnt to distinguish the skin cancer types are:

- Color Variation
- Symmetricity
- Size

- Texture
- Border and Lesion Boundaries

Additionally, metadata associated with the image like the site of the lesion in body, sex, age is also used to train the model that provides additional insights to the model to increase the accuracy. A short description about the dominant features of the type of skin cancer we have classified in this project namely; Basal Cell Carcinoma, Squamous Cell Carcinoma, Melanoma and Benign Keratosis are:

1.2.1 Squamous Cell Carcinoma

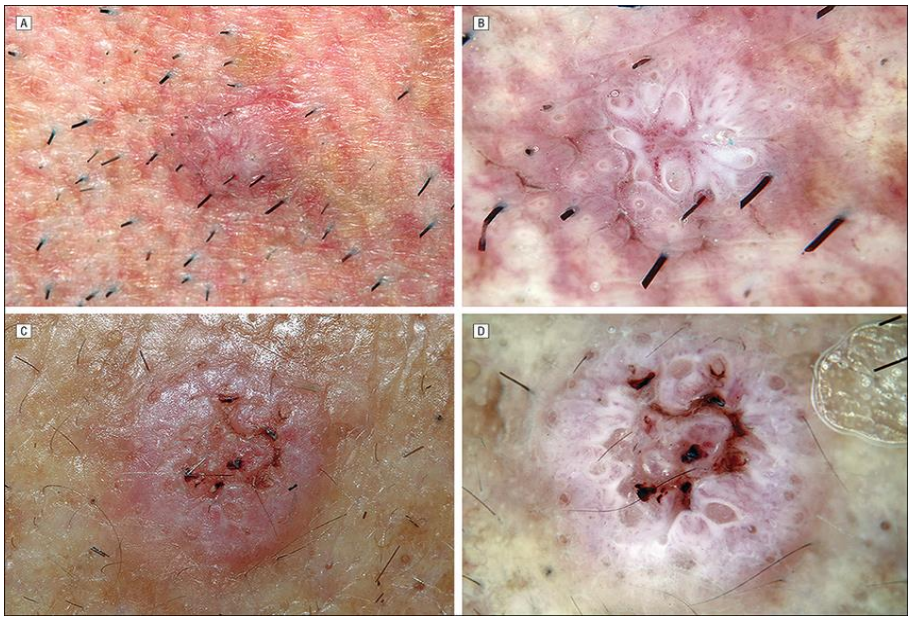


Figure 1.1: Dermoscopic Image of Squamous Cell Carcinoma

a. **Color:**

   Squamous cell carcinomas occur in various colors, such as reddish, pink, or brown shades.

b. **Texture:**

   The texture of squamous cell carcinomas may appear scaly or rough, with a tendency to have ulcerations or crusts.

c. **Elevation:**

   They can have an elevated or raised appearance compared to the surrounding skin.

d. Border:

The border of a squamous cell carcinoma may show irregularity, with poorly defined or jagged edges.

e. Other clinical features:

Squamous cell carcinomas may be associated with symptoms like bleeding, oozing, or non-healing sores.

1.2.2 Basal Cell Carcinoma

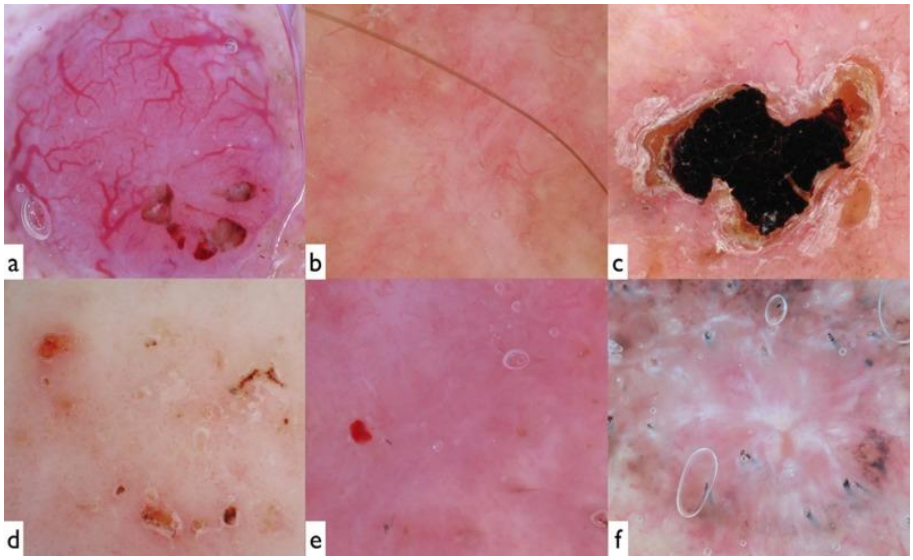


Figure 1.2: Dermoscopic Image of Basal Cell Carcinoma

a. Color:

Basal Cell Carcinomas often have a variety of colors, including pink, flesh-colored, brown, or black.

b. Texture:

The texture of BCCs can be smooth or pearly, sometimes with small blood vessels visible on the surface.

c. Border:

BCCs typically have a well-defined border, appearing rolled or raised.

d. Ulceration:

Some BCCs may exhibit ulceration, with a central area of erosion or crusting.

e. Other clinical features:

BCCs may present as nodules, bumps, or lesions with central depressions.

1.2.3 Melanoma

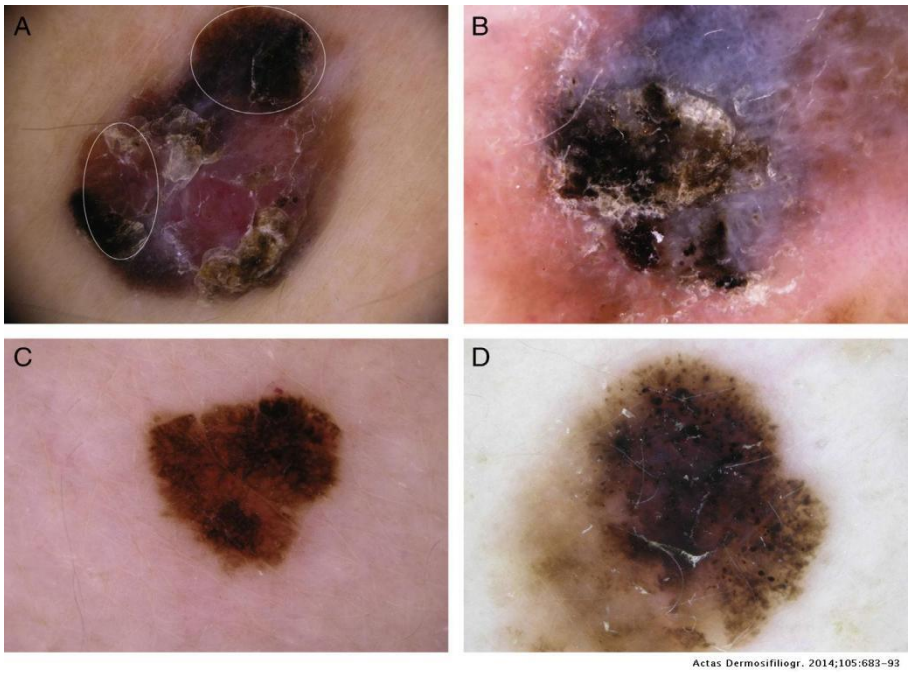


Figure 1.3: Dermoscopic Image of Melanoma

Irregular shape and structure, broad network, blue-white veils, multiple brown dots are some of the striking features of melanoma.

a. **Asymmetry:**
   Melanomas often exhibit asymmetrical shape, with one half differing from the other half.
b. **Color variation:**
   Melanomas can display variations in color, such as multiple shades of brown, black, blue, red, or white within the lesion.
c. **Diameter:**
   Melanomas tend to be larger in diameter, typically greater than 6 mm, although they can also be smaller.
d. **Border irregularity:**
   Melanomas often have irregular, jagged, or poorly defined borders, with notches or scalloped edges.
e. **Other clinical features:**
   Melanomas may have features like rapid growth, changes in size, shape, or color over time.

1.2.4 Benign Keratosis

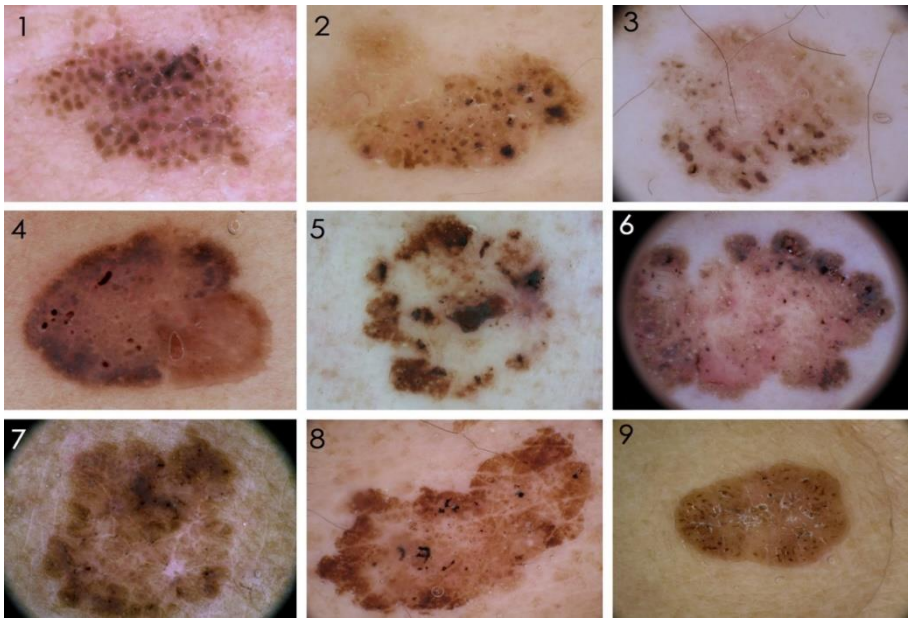


Figure 1.4: Dermoscopic Images of Benign Keratosis

a. **Color:**

   Benign keratosis lesions may have various colors, including light brown, dark brown, or tan.

b. **Texture:**

   They can exhibit rough or scaly texture, often with a thickened or wart-like appearance.

c. **Border:**

   Benign keratosis lesions typically have well-defined borders that are smooth or slightly raised.

d. **Size:**

   They can vary in size, ranging from small to larger lesions.

e. **Other clinical features:**

   Benign keratosis may be associated with symptoms like itching or tenderness.

1.3 PROBLEM STATEMENT

Skin cancer detection is itself a challenge as it relies on visual observation by healthcare professionals in the initial phase followed by biopsy. Different professionals have different levels of experience and expertise which may result in inaccurate results in the early phase of diagnosis leading to further complications and misdiagnosis. One of the main challenges in accurately diagnosing cancer lesions is that both malignant and benign skin conditions have overlapping features which will make it hard to differentiate between them. So, this method of cancer detection i.e., visual observation and evaluation by the health professionals in the initial checking phase suffers from limitations that may lead to misdiagnosis and inaccuracy. Delayed detection and misdiagnosis can result in several severe consequences which might impact both patients, his/her family and health care system.

1.4 OBJECTIVES

- To develop skin cancer detection and classification system using transfer learning.

1.5 SCOPE AND APPLICATIONS

The main scope of this project is to help in accurate detection of skin cancer during the initial phase. Early detection exponentially increases the chances of curing the cancer before it spreads. With the capacity of AI, this model can aid the medical professionals in diagnosing the disease efficiently and accurately. With some modifications and training, this can be extended to detect other types of skin diseases and other forms of cancers.

The model can be further extended, and the applications are listed as follows:

- It helps in preliminary assessment of skin cancer.
- It can assist dermatologists and healthcare professionals to efficiently carry out initial phase skin cancer diagnosis.
- By extending the developed system, it can be used in mass screening programs of high-risk communities.

CHAPTER 2: LITERATURE REVIEW

According to WHO, currently there are 2-3 million non-melanoma and 132 thousand melanoma skin cancer patients worldwide and the numbers increase with each passing year. The number of people affected by skin cancer is expected to rise to almost 13.1 million by 2030.

The recent advancements in the field of AI have shown optimistic outcomes in the automated detection of skin cancer. Over the years, various methods or techniques have been developed to facilitate this. This section aims to provide a synopsis of the relevant studies, research and investigations associated with the subject of our project.

In Multiclass skin cancer classification using EfficientNets – a first step towards preventing skin cancer [2], Karar Ali, Zaffar Ahmed Shaikh, et al developed a preprocessing image pipeline. They performed transfer learning on pre-trained ImageNet weights and fine-tuned the Convolutional Neural Networks and trained the EfficientNets B0-B7 on the HAM10000 dataset. After evaluating the performance of all EfficientNet variants using metrics such as Precision, Recall, Accuracy, F1 Score, and Confusion Matrices, they observed that the EfficientNetB4, achieved an F1 Score of 87 percent and a Top-1 Accuracy of 87.91%.

Google's Inception v4 CNN architecture was tested by H.A. Haenssle, C. Fink, et al and their findings were reported in the article 'Man against Machine' [3]. They measured the sensitivity, specificity and area under the curve (AUC) of receiver operating characteristics (ROC) for diagnostic classification of lesions. The test was performed in 2 levels between CNN and a group of 58 international dermatologists and it was found that the CNN ROC AUC was greater than the mean ROC AUC of the dermatologists (0.86 versus 0.79).

DSCC_Net: Multi-Classification Deep Learning Models for Diagnosing of Skin Cancer Using Dermoscopic Images [4], proposed a CNN model with the intention to build a multi-classification technique for diagnosing skin cancers such as melanoma, BCC, and SCC. It was evaluated on three datasets (ISIC 2020, HAM10000, and DermIS). The classification performance of the model was compared with ResNet-152, VGG-16, VGG-19, Inception-V3, EfficientNet-B0, and MobileNet. The results concluded that

this proposed model performed better than the other models and obtained a 99.43% AUC, 94.17% accuracy, a recall of 93.76%, a precision of 94.28%, and an F1-score of 93.93%.

In research work conducted by S. Likhitha and R. Baskar [5], R-CNN and Inception V3 algorithms were used for skin cancer detection and the accuracy was determined. The accuracy and specificity of R-CNN was 96.01% and 86.76 respectively and that of Inception V3 was 92% and 89.47. Thus, it was concluded that R-CNN performs more accurately than Inception V3. In other research [6] by the same team, it was found that the accuracy of CNN algorithm (95.03%) was better than Support Vector Machine algorithm (93.04%) for skin cancer detection.

In the paper by Z. Li [7], a web application based on CNN was designed to accurately diagnose skin cancer. HAM10000 dataset, eight convolutional layers, four max pooling layers, and batch normalization and dropout layers were utilized in this project. The model was then compiled on adam optimizer and an accuracy of 0.9703 was obtained. Finally, Flask web framework was implemented for users to submit their skin images and view their predictions.

Navid Razmjooy and Ali Arshaghi [8] presented a method for skin cancer diagnosis based on deep learning and metaheuristics. For this, the method combined a multi-level optimal thresholding segmentation technique with a metaheuristic algorithm, based on Multi-agent Fuzzy Buzzard algorithm and obtained an accuracy of 0.94 and specificity of 0.93.

2.1 EXISTING AI SYSTEMS

Lots of research has been carried out in this field and there are lots of apps available. Few of them are:

2.1.1 AI Dermatologist: Skin Scanner

It is an app available in Play Store that helps in analyzing the skin lesion.

The features of AI Dermatologist: Skin Scanner are:

- It accepts the skin lesion image either in real time through camera or through images saved in gallery.
- The app gives an option to save the result.
- The app provides instructions like; the image must be clicked from 5-20 cm apart and in proper sunlight so that the result is as accurate as possible.

Despite being a great app, few limitations of it are:

- It gives only one free trial and then payment is to be made.
- It relies only on the image, not on other information like age, site and sex.

2.1.2 Scanoma

It is another app available in Play Store that helps in analyzing the moles. The app focuses on only moles and to determine whether they are concerning or not, rather than classifying the skin cancer type.

The features of scanoma app are:

- The app provides instructions like to place the phone 90 degree to the mole so that the result is as accurate as possible.
- It accepts the skin lesion image either in real time through camera.

The limitations of this app are:

- It only deals with moles, not the skin cancer types like basal cell carcinoma, squamous cell carcinoma, and melanoma.

Skin Vision app is not available in Nepal, and Miiskin app requires payment subscription, otherwise we can't move forward.

2.2 EXISTING SYSTEM IN NEPAL

In the context of Nepal, the first step involves patients suspecting the changes on their skin based on visual observation followed by visiting a dermatologist.

For the next step, the medical professional predicts the chances of skin cancer and the probable type of the cancer by visual observation of the affected area or dermatoscopic image and recommends biopsy if needed.

The final step is biopsy which accurately diagnoses skin cancer and its type.

In foreign countries AI has been dominantly used in medical sector and there is a health-tech company called as Skin Analytics that uses AI to diagnose skin cancer but there is no extensive use of AI in medical field in Nepal.

2.3 LIMITATIONS OF EXISITNG SYSTEM

The main problem seems to lie in the first step i.e., patient might ignore the changes in skin and second step i.e., initially doctors do a visual examination and based on that, a biopsy is carried out. Now, there is a high chance of misdiagnosis in the initial phase, which may lead to cases like biopsy being done when not needed, and biopsy not been done, when needed. In short, there is a chance of misdiagnosis and may risk the life of the patient as some cancer types like melanoma spread rapidly. With the assistance of AI based model, it can help reduce the problems currently existing.

One of the common limitations present in the existing apps are solely depending on images without other information like age, sex, site, or they are designed only to differentiate skin moles into cancerous or non-cancerous. If all these aspects are also taken into consideration, then the result would be more accurate.

2.4 SOLUTIONS PROVIDED BY OUR SYSTEM

With regard to the existing system in Nepal, the main purpose of the developed model is to minimize the problems encountered during the first and second step of the current system. This developed method is an efficient method where image processing and data augmentation are performed to the dermatoscopic images provided by the patients or dermatologists, so that the diagnosis made can be consulted with the result of the model. Our model has been developed taking clinical data like age, sex, site into consideration. Although the developed model may not match the standard of the system like Skin Analytics or existing apps like Skin Vision, developed by experts in the field, it can surely be a step towards revolutionizing AI in the field of skin cancer diagnosis in Nepal.

CHAPTER 3: METHODOLOGY

3.1 PROCESS MODEL

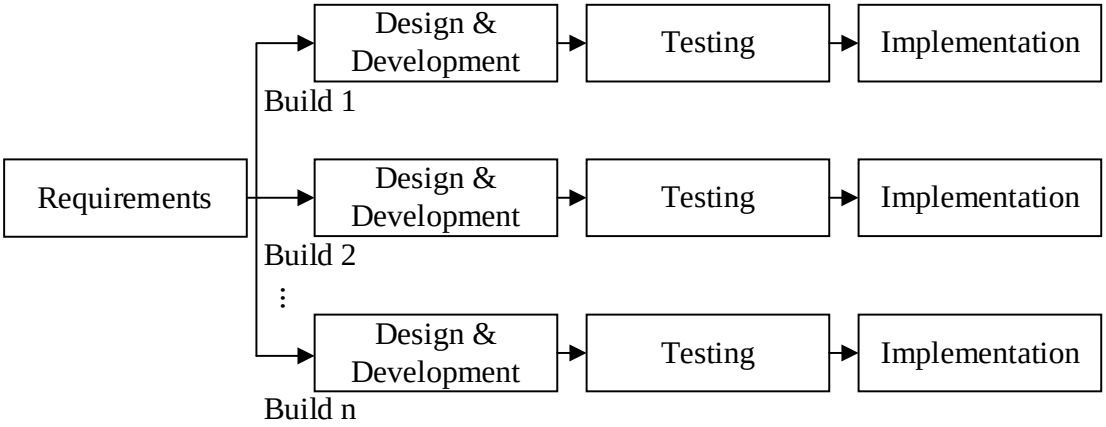


Figure 3.1: Incremental Model

Our system is developed by using the incremental model. In incremental model, firstly, requirements were properly analyzed and based on the initial feasibility study, the entire system is divided into smaller modules called increments which are developed and deployed independently. We worked on module-by-module basis, where first, a single module was developed completely before shifting to another module. While working with each module, the steps- requirement analysis and gathering, design and development, testing and implementation are followed. With the release of each module, functionality gets added to the previous release of the system and finally as the process continues, the system gets developed.

The phases of incremental model and the works carried out in each phase with regard to our project are:

3.1.1 Requirement Phase

The main purpose and aim of the project are determined in this phase. All the system functional requirements and software specifications are identified in this phase. The feasibility for the development is also analyzed. Before developing each module, requirement analysis is carried out for that module.

As for our project, in the requirement phase, firstly, a field visit to 'Alka Hospital', 'Teaching Hospital' and 'National Hospital and Cancer Research Center' was made and

discussion was carried out with the doctors regarding the situation of skin cancer and the relation between AI and skin cancer field. Then, the requirements for the project, images and the related metadata were obtained through ISIC 2019 dataset available in Kaggle.

3.1.2 Design and Development

All the requirements identified in the previous phase are used to make a structure that is suitable for programming. In this phase the software architecture that is to be used for developing the software module is decided. The coding is started in this phase to create the desired software, but here the design and development takes place module by module. In each increment, a particular module is worked upon.

In this phase, based on the analysis carried out in requirement phase, different ML models were explored and finally EfficientNetB4 model was chosen to develop our model. Then, the basic system block diagrams, flowchart were designed, and coding was done module by module by decomposing the whole program into modules like preprocessing, one-hot encoding, augmentation, model development, fine-tuning, performance evaluation modules.

3.1.3 Testing

Once the design and development phase are completed for a module, it is released for testing. All the bugs in the software module are found out in this phase. The quality and functionality of the module is ensured in this phase.

After developing each module, testing was done. After developing the preprocessing module, it was tested, and errors were detected and the code was refined. This process was carried out for each module mentioned above.

3.1.4 Implementation

The implementation phase is the final stage where the module that has been tested is made available to the review team after each increment. It is the step where the increment is made available for evaluation.

After designing and testing each module, the module was declared as the part of the code, and the phases; designing, development, and testing were applied to the next module, until the complete software is developed.

The conditions to use incremental model and the alignment with our project are:

- The requirements of the system are specified and understood clearly upfront.
- The product is demanded to be released early.
- The developing team is in shortage of resources or lacks skill set.
- The use of new technologies and when high risk attributes and goals are involved.
- The entire software can be broken down into modules.
- Project has quite lengthy development schedule.

The above-mentioned points align with the need of our project to develop skin cancer classification model as the concept of CNN was new concepts to us, and while starting the project, we lacked the knowledge and skills in this field. Also, our software system could be broken down into modules and the requirements were clearly understood before beginning the project through field visit and deep research in the net.

3.2 SYSTEM BLOCK DIAGRAM

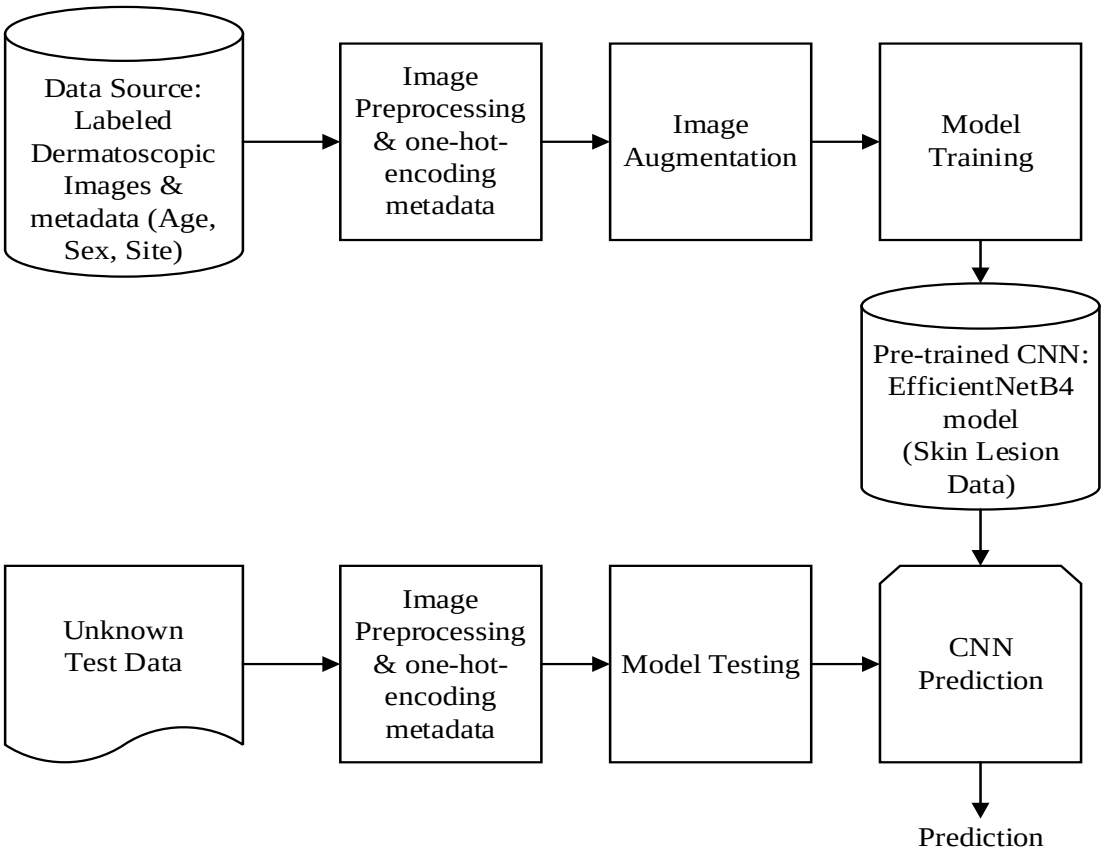


Figure 3.2: System Block Diagram

The above picture illustrates the basic block diagram of our system, which takes image of skin lesion, and the corresponding age, sex, anatomical sites (if available) as the input and gives the prediction of the skin cancer types as output.

3.2.1 Image Processing and One-hot Encoding Metadata

Preprocessing is one of the very important steps before feeding the data to the neural network so that the data is compatible with the model. The images are resized to a particular dimension, 224 X 224 in our case, converted and checked to be in RGB format and color normalized. Also, the metadata (age, sex, site) of the image are one-hot-encoded, as they need to be converted to vector form that is acceptable and can be processed upon by the model. Keeping the fact in mind that all images might not have each value of the associated metadata and missing values might cause error further, code was also written to handle missing values by using placeholders.

3.2.2 Image Augmentation

The preprocessed images were augmented like rotated, scaled, flipped, shifted to increase the variations in input so that model can produce more effective results without any bias. Also, the distribution of data in the four classes showed class imbalance, i.e. 'Melanoma' has the highest number of data, and 'Squamous Cell Carcinoma' has the least data, and it was tackled using balanced class weight function so that the model learnt equally from all classes without any bias. Also, the processed and augmented data were then converted to NumPy array so that model could process them.

3.2.3 EfficientNetB4 model as the Base Model

The architecture of CNN model was developed, using the EfficientNetB4 model trained on ImageNet as the base model where few layers of the base model were fine-tuned by freezing the first few layers and unfreezing the rest so that our model could get benefit of the learning of the EfficientNet model as well as weights can be adjusted during the training process.

3.2.4 Model Training

The top layers of the EffcientNetB4 model were removed and final classification layer containing 4 neurons was added to classify the four types of skin cancer, also max pooling, average pooling, dense and dropout layers were added and the model is trained with the preprocessed and augmented training dataset till the desired accuracy is obtained by adjusting the hyperparameters like batch size and number of epoch. During the process of training the model, we got the best possible result taking batch size 32 and epoch as 30.

Then testing was carried out with the help of unknown test data, which is preprocessed, and fed to the CNN model which produces the prediction.

3.3 DATA COLLECTION

We have used ISIC 2019 dataset available in Kaggle for the development of our system. The dataset contains 8 classes of skin cancer as follows:

Class Label	Abbreviation	Class	Number of Images
1	AK	Actinic keratosis	867
2	BCC	Basal Cell Carcinoma	3323
3	BKL	Benign keratosis	2624
4	DF	Dermatofibroma	239
5	MEL	Melanoma	4522
6	NV	Melanocytic Nevi	12,875
7	SCC	Squamous cell carcinoma	628
8	VASC	Vascular Lesions	253
	Total		25,331

Figure 3.3: Analysis of ISIC-2019 Dataset

First of all, 11097 labeled dermatoscopic skin images were obtained from ISIC 2019 dataset available in Kaggle which belong to either of the four classes of skin cancer as melanoma, SCC, BCC or benign keratosis. The type of cancer of the images were in a CSV file named ‘ground_truth’ and the corresponding metadata about the image were in another CSV file named ‘metadata’ all related by image number. So, we separated the data for training and testing, and we associated the images, and the csv files together and prepared the data.

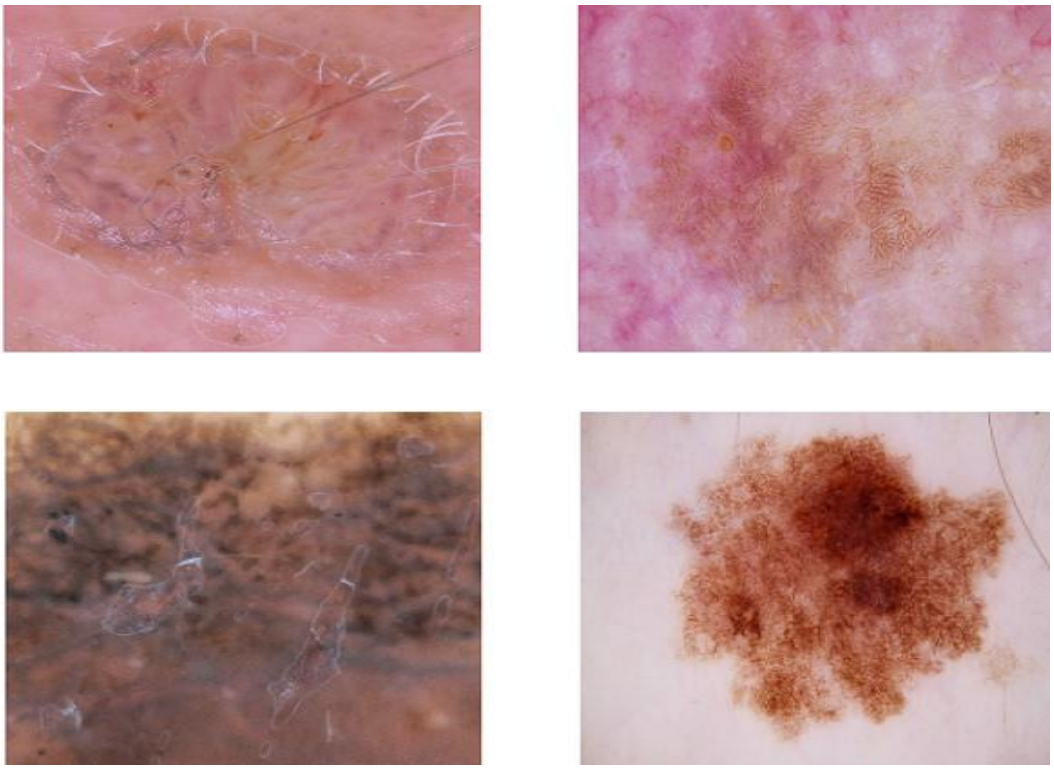


Figure 3.4: Sample of Images Available in ISIC-2019 Dataset

	image	MEL	NV	BCC	AK	BKL	DF	VASC	SCC	UNK
0	ISIC_0000000	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	ISIC_0000001	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	ISIC_0000002	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	ISIC_0000003	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	ISIC_0000004	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 3.5: 'ISIC_2019_Training_GroundTruth' CSV File

	image	age_approx	anatom_site_general	lesion_id	sex
0	ISIC_0000000	55.0	anterior torso	NaN	female
1	ISIC_0000001	30.0	anterior torso	NaN	female
2	ISIC_0000002	60.0	upper extremity	NaN	female
3	ISIC_0000003	30.0	upper extremity	NaN	male
4	ISIC_0000004	80.0	posterior torso	NaN	male
...	...	...	...	...	...
25326	ISIC_0073247	85.0	head/neck	BCN_0003925	female
25327	ISIC_0073248	65.0	anterior torso	BCN_0001819	male
25328	ISIC_0073249	70.0	lower extremity	BCN_0001085	male
25329	ISIC_0073251	55.0	palms/soles	BCN_0002083	female
25330	ISIC_0073254	50.0	upper extremity	BCN_0001079	male

Figure 3.6: 'ISIC_2019_Metadata' CSV File

Firstly, model was made only using images, then research was done and the additional data about the image; age, sex, site were also taken into consideration.

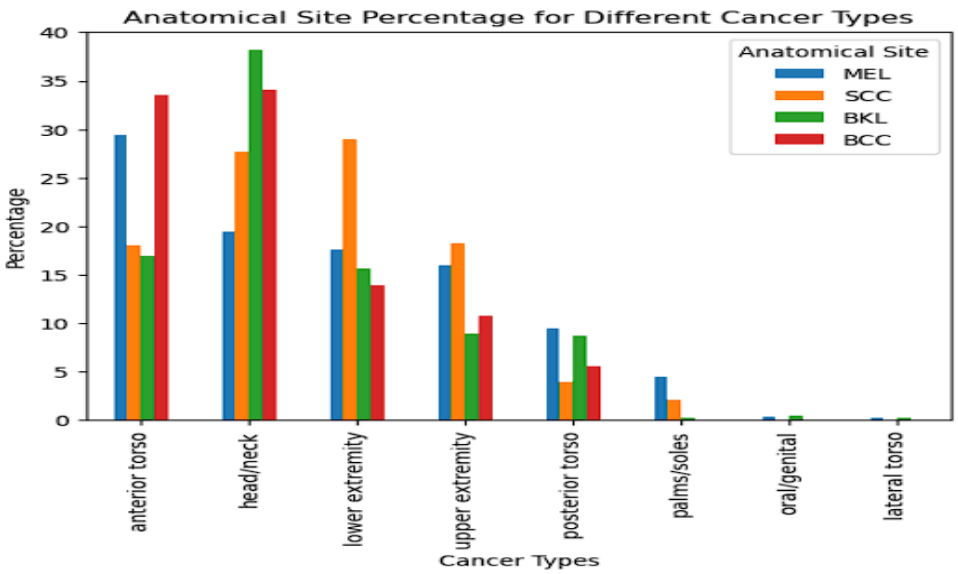


Figure 3.7: Plot of Anatomical Site Distribution across the Classes of Skin Cancer

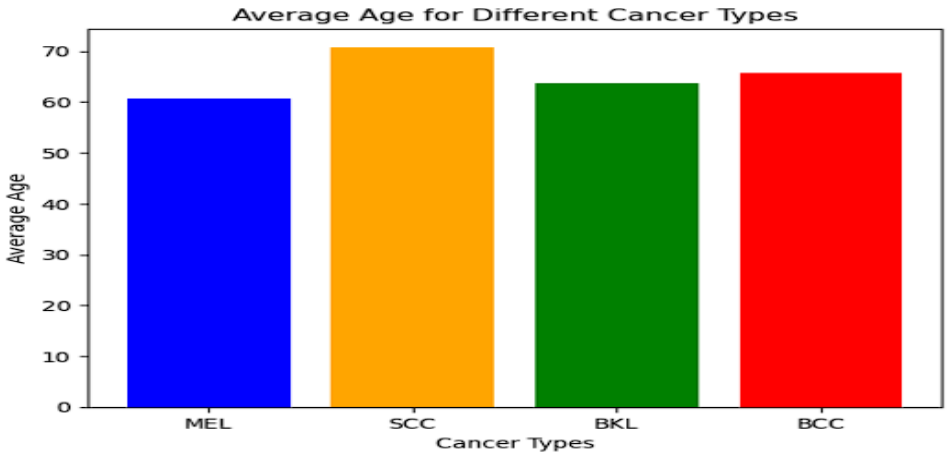


Figure 3.8: Plot of Average Age Distribution across the Classes of Skin Cancer

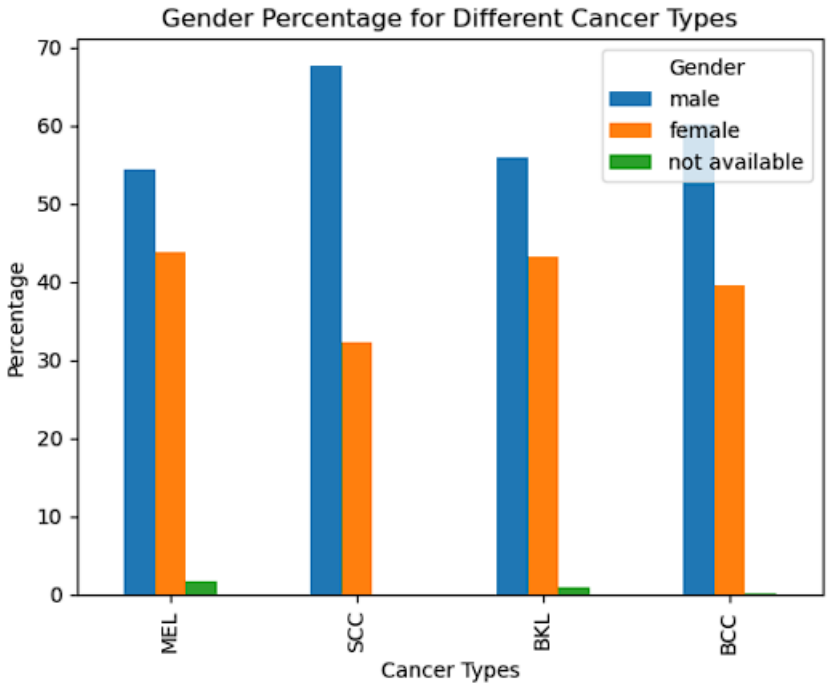


Figure 3.9: Plot of Gender Distribution across the Classes of Skin Cancer

From the above visualizations, it can be concluded that for the images we are going to work with, the metadata can provide additional insights to the model.

Anatomical Site: Certain types of skin cancer are more likely to occur in certain specific areas like melanoma is proven to occur most frequently in anterior torso and head and neck. The plot obtained also verifies the fact. BCC is seen to be present mostly in images with anatomical sites such as anterior torso and head/neck.

Age: Just like in the case of anatomical sites, certain types of skin cancer is more likely to occur in certain age groups and this can help the model perform better.

Sex: Sex also plays an important indicator of cancer type. Due to more exposure of men to sun while doing labor works, SCC has been found to occur nearly twice the count in males than in females and the above distribution also verifies the fact. Also, it is proven that occurrence of melanoma is more in males than in females.

Although the nature of skin cancer varies vastly according to different geographical locations and exposure to sun, developing model by using this dataset can act as a good starting point for diving into the realm of skin cancer detection using AI.

3.4 ALGORITHM

The steps intended to be applied in our project are:

Step 1: Start

Step 2: Collect the dataset containing images labeled as their respective skin cancer type along with additional metadata like age, sex, anatomical sites.

Step 3: Preprocess images like resizing, normalization, perform data augmentation and one-hot-encode the attributes sex and anatomical sites.

Step 4: Design the CNN architecture using the pre-trained EfficientNetB4 as a base model and add the dense layers for processing the metadata and top layers including the classification layer as per our task.

Step 5: Train the model by splitting the dataset in the ratio 80:20 for training and validation respectively.

Step 6: Evaluate the accuracy and loss of the model, if not satisfactory go to Step 4 and adjust the hyperparameters, else Step 7.

Step 7: Test the developed model.

Step 8: Deploy the model in a webapp, where users can input image along with the corresponding age, sex, and anatomical site.

Step 9: Preprocess the uploaded data as in step 3.

Step 10: Pass the processed data to the trained model for prediction.

Step 11: Display the prediction to the user.

Step 12: Stop

3.5 CONVOLUTIONAL NEURAL NETWORK

CNN is a DL technique that is specifically designed to analyze visual data such as images. ConvNet is widely used in computer vision tasks such as image classification, object detection, and image segmentation. CNNs draw inspiration from the structure of visual cortex of animals, where specialized neurons respond to specific visual stimuli. The architecture of ConvNet mirrors the connectivity pattern of neurons in the human brain.

CNNs offer distinct advantages for image processing tasks owing to their unique architecture and characteristics. ConvNet employs local receptive fields to capture spatial relationships within images. By focusing on small regions and sharing weights across the image, CNNs efficiently extract relevant features while preserving spatial information. Furthermore, ConvNet requires less pre-processing compared to other classification algorithms, resulting in the decrement of overall preprocessing tasks.

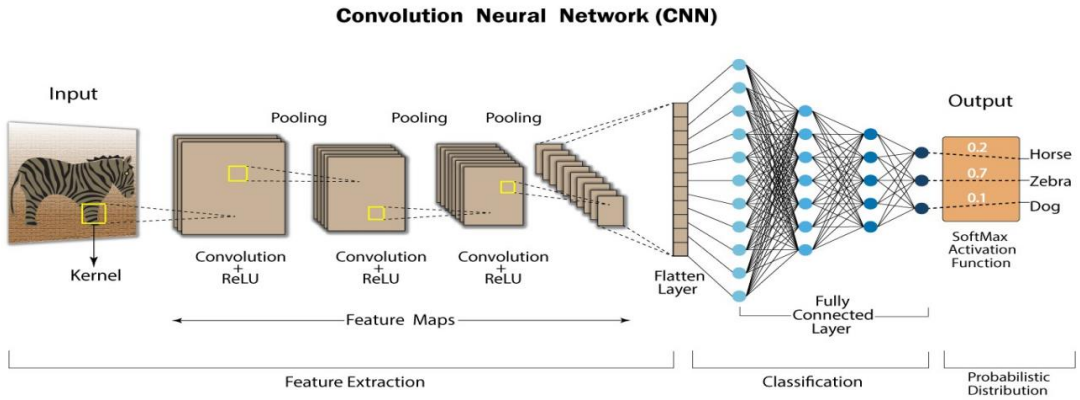


Figure 3.10: Working of CNN

3.6 TRANSFER LEARNING

In traditional machine learning, models are built from scratch using the data we have for a particular task. In contrary, Transfer Learning is an approach in machine learning where while creating a new model for certain task, a previously trained model on similar task is used as the base model, the starting point and then the new classification layers

are added. As for example, if there is a model previously created to identify different animals, then we can use that model as a base model for new classification task to identify different species of birds and this approach is termed as transfer learning. There are many pre-trained CNN models like MobileNet, VGG, ResNet, Inception, DenseNet, EfficientNet, which are already trained on large dataset like ImageNet. So, by simply removing the last few layers, more specifically the classification layers of the original model, and replacing them with the layers needed for our task, we can create a new model.

Transfer learning with CNNs is effective because CNNs are known for their ability to learn useful features from images, and the pre-trained models have already learned the useful features from the dataset they were trained on, so it helps to increase the training time and the efficiency of the new model to be created. Using transfer learning, we can fine-tune the model to achieve the required accuracy for our task by adjusting various hyperparameters such as batch size, learning rate, number of epochs and also setting the fine-tuning layers to freeze and unfreeze.

3.7 EFFICIENTNET ARCHITECTURE

Among many other available pre-trained CNN models used for the purpose of transfer learning, we have used EfficientNet for our model development because it helps in achieving higher accuracy as it balances the weight and cost; it provides high accuracy at low number of parameters. MobileNet is a lightweight model but has less efficiency whereas ResNet, Inception provide higher efficiency but at the cost of heavy weight. There are different variants of EfficientNet like EfficientNet-B0 to EfficientNet-B7; where EfficientNet B0 is the baseline model and through increasing the scaling factor, other higher variants are obtained. As the variant number increases, the efficiency,performance, number of layers,size and computational resources increases. MBConv is the building block of EfficientNet architecture and has a compound scaling method that scales up resolution, width and depth at the same time.

- **Depth:** Depth refers to the number of layers in neural network and it plays a very important role as networks with many layers can capture more complex features but at the same time, it also consumes more resources.

- **Width:** Width refers to the number of channels or feature maps in each layer and it also plays an important role in neural network similar to depth.
- **Resolution**: It is known that lower resolution images provide less detailed information, so model with less accuracy is developed but consumes less resources and vice-versa.

For all these three dimensions, EfficientNet, introduced as "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", in 2019, by Google Researchers, has the provision for compound scaling that combines the above-mentioned scaling; depth, width and resolution scaling to achieve a better model. The logic between compound scaling is to achieve better accuracy by scaling up among the three dimensions, while keeping a balance between them. The simple mathematics behind compound scaling is:

$f = \alpha \ \beta^{\phi} \ \gamma^{\phi}$ -------(i)

where f= network scaling factor

α = depth scaling factor = 1.2

β = width scaling factor = 1.1

γ = resolution scaling factor = 1.15

ϕ is user-defined constant that makes balance between the three scaling dimensions.

EfficientNet model has similar layers to traditional CNN models except that it uses the concept of mobile inverted bottleneck convolution. The basic layers are:

3.7.1 Stem Convolution

It is the first layer that has image as input and it performs repeated cycle of convolution, batch normalization, activation function to extract the features from the input.

3.7.2 EfficientNet Blocks

There is presence of inverted residual blocks that are repeated blocks of MobileNetV2 Block Convolution (MBConv) that perform actions like depthwise separable convolution, pointwise convolution.

3.7.3 Downsampling Layer

In this architecture, there frequently occurs layers like pooling or reduction layer that helps to reduce the spatial dimension of image.

3.7.4 Feature Map

The convolution operation is applied using the concept of compound scaling to obtain the feature maps.

3.7.5 Global Average Pooling Layer

Towards the end, there occurs a pooling layer that helps in reducing the image's dimension as mentioned in downsampling layer.

3.7.6 Dense Layer / Fully Connected Layer

This is the last layer that takes the raw output from previous layers and through activation function like softmax converts into probability values for each class. The fully connected layers establish connection and help to associate every neuron from the previous layer to each neuron in the next layer which implies that every unit in a dense layer receives input from all units in the previous layers.

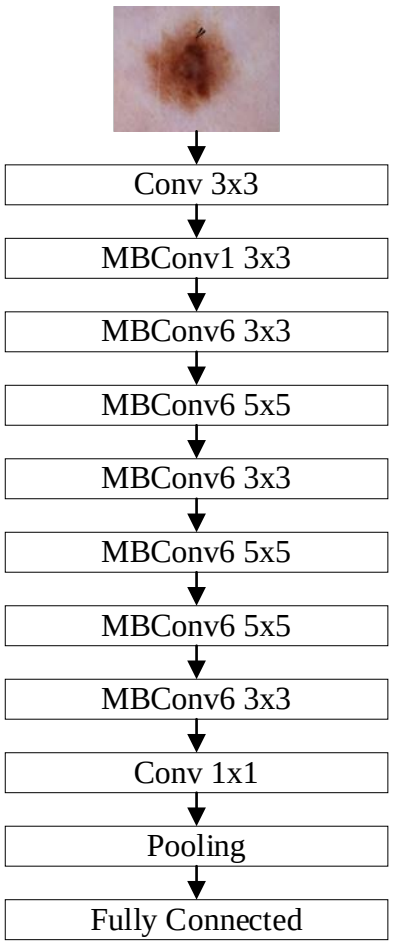


Figure 3.11: Basic Architecture of EfficientNet Model

Some terminologies used in our project are:

- **Convolutional Layer:** Convolution is the backbone of convolutional layers in ConvNet. In the context of CNNs, convolution means to extract relevant features from an input image or feature map by applying a set of adaptable filters using mathematical operations.
- **Activation Function (ReLU):** The output of the convolutional layer, typically, often presented in matrix form, is then acted upon by an activation function such as ReLU which works elementwise to introduce non-linearity and enhance the network's ability to learn complex patterns. ReLU typically converts all negative values in the feature map to zero, leaving the positive values as they are. Mathematically, ReLU function can be defined as f(x)=max (0, x).
- **Pooling:** Pooling Layers down sample the feature maps obtained from convolutional layers, reducing their spatial dimensions. The two common pooling techniques are Maximum Pooling and Average Pooling.

Maximum Pooling also referred to as Max Pooling, the maximum value within each pooling window is selected and propagated on to the output feature map. The main purpose of maximum pooling is to capture the most prominent feature within each region while reducing spatial dimensions.

Whereas, in Average Pooling, the average value of pixels within each pooling window is calculated and used as representative value for that region in the output feature map. The main purpose of average pooling is to reduce spatial dimensions and provide more generalized representation of the input data.

- **Dropout:** Dropout is a regularization technique used in CNNs to mitigate overfitting and to improve the model's ability to generalize. It randomly drops out certain units in neural networks.
- **Flattening:** Flattening is one of the important processes in CNN where multi-dimensional data obtained from convolutional and pooling layers are converted into one-dimensional vectors.
- **Softmax Activation:** Softmax activation is a mathematical function that converts the output of a neural network into a probability distribution, ensuring that the sum of the output values equals one and each value represents the likelihood of a particular class.

To develop the model, first, the model was built to work with images only, by using EffficientnetB4 as base model which helps to extract necessary features from the cancer images, followed by max pooling and average pooling to reduce the dimension of the image. Then, the metadata about the images; age, sex, site were also passed to the dense layer with non-linear activation ReLU and after extracting the necessary features from all the input sources, late fusion concatenation was performed to pass all the input features to the model. Then, two consequent dense and dropout layers were added to introduce regularization followed by the final top layer with 4 neurons and softmax activation for classification. The layers from the Efficientnet model were frozen so that the features Efficientnet learnt previously could be used and the rest added layers are not frozen so that they are trainable and the weights can be adjusted.

3.8 FLOWCHART

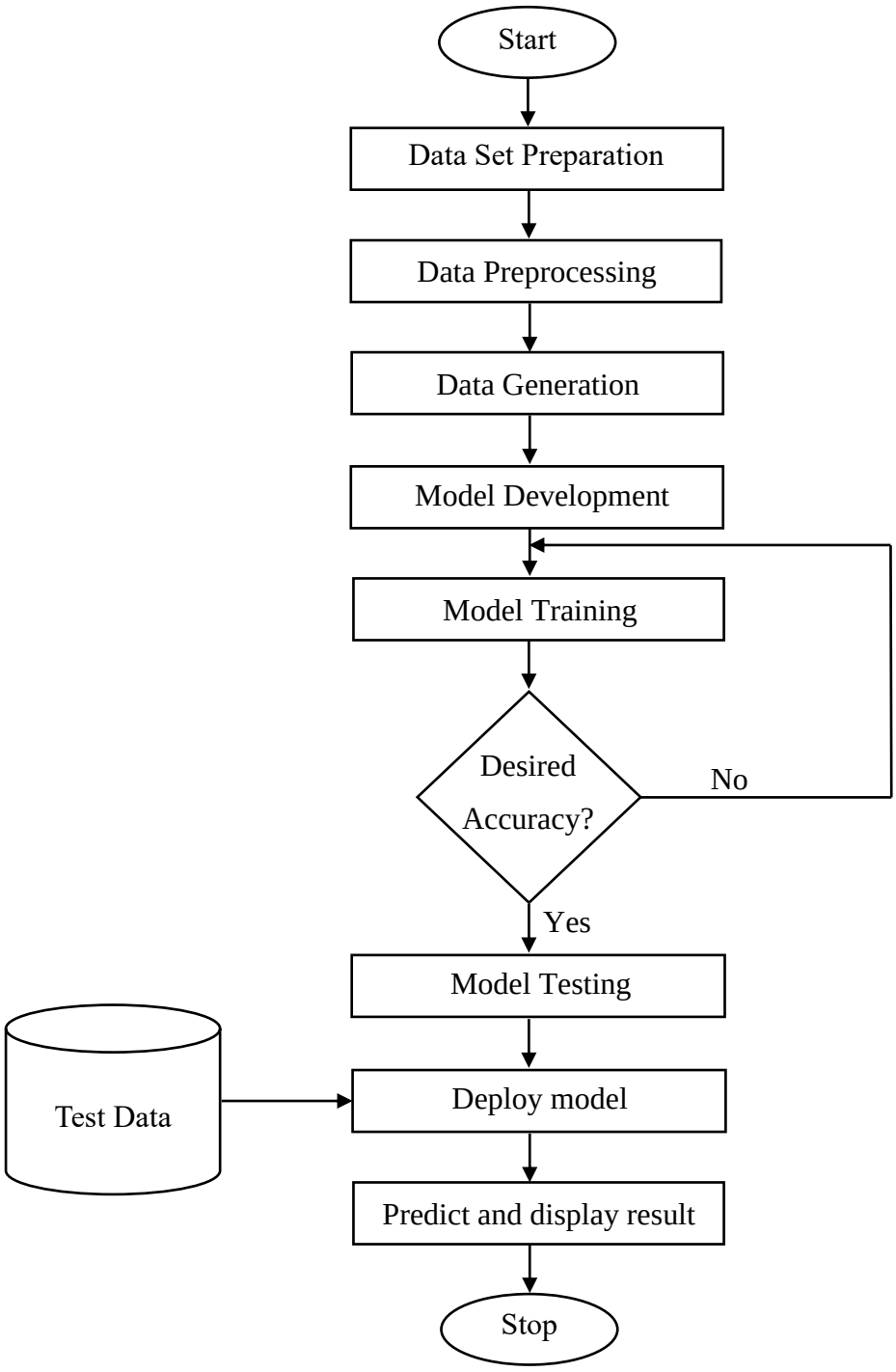


Figure 3.12: Flowchart

3.9 UML DIAGRAMS

3.9.1 Use Case Diagram

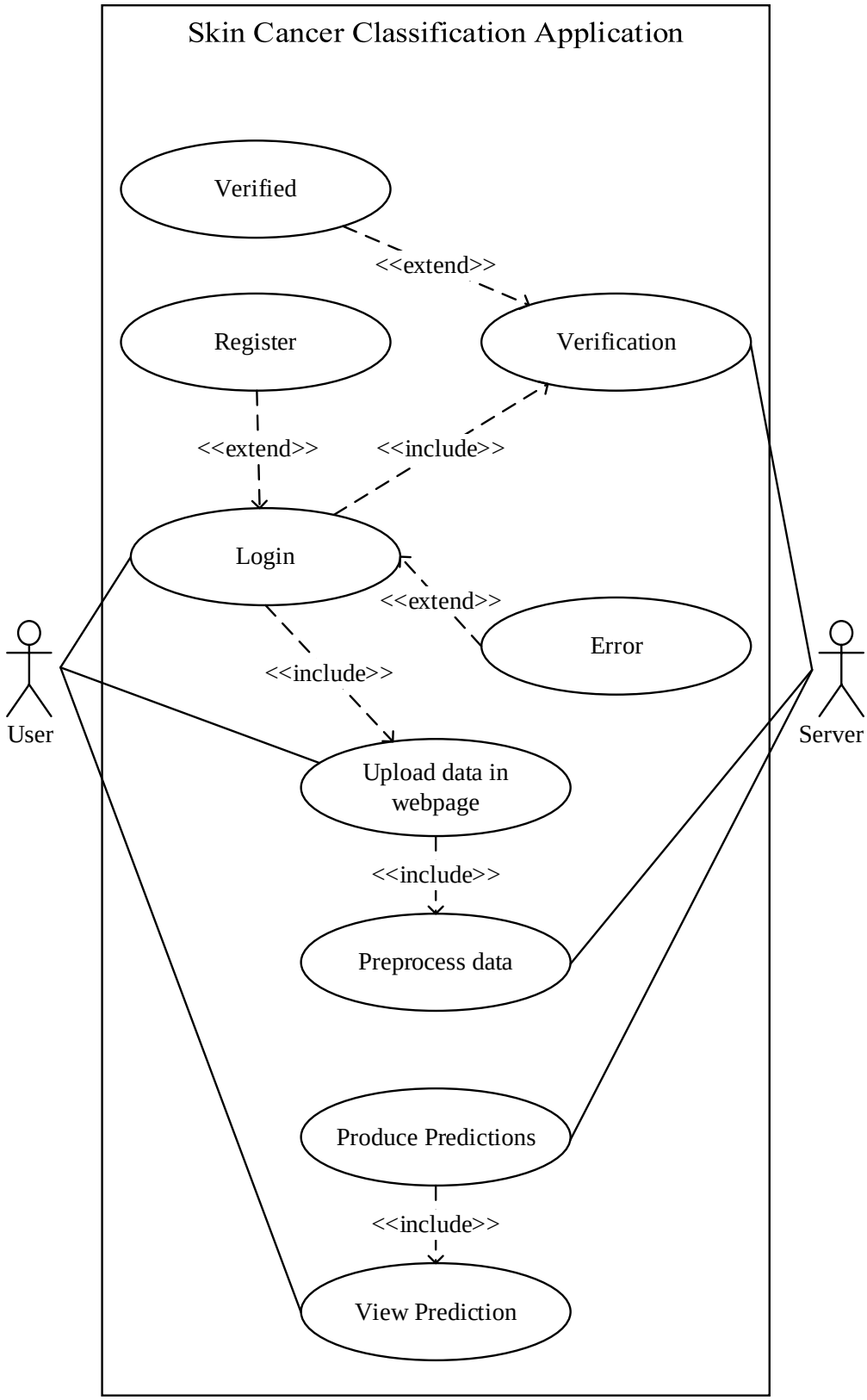


Figure 3.13: Use Case Diagram

3.9.2 Sequence Diagram

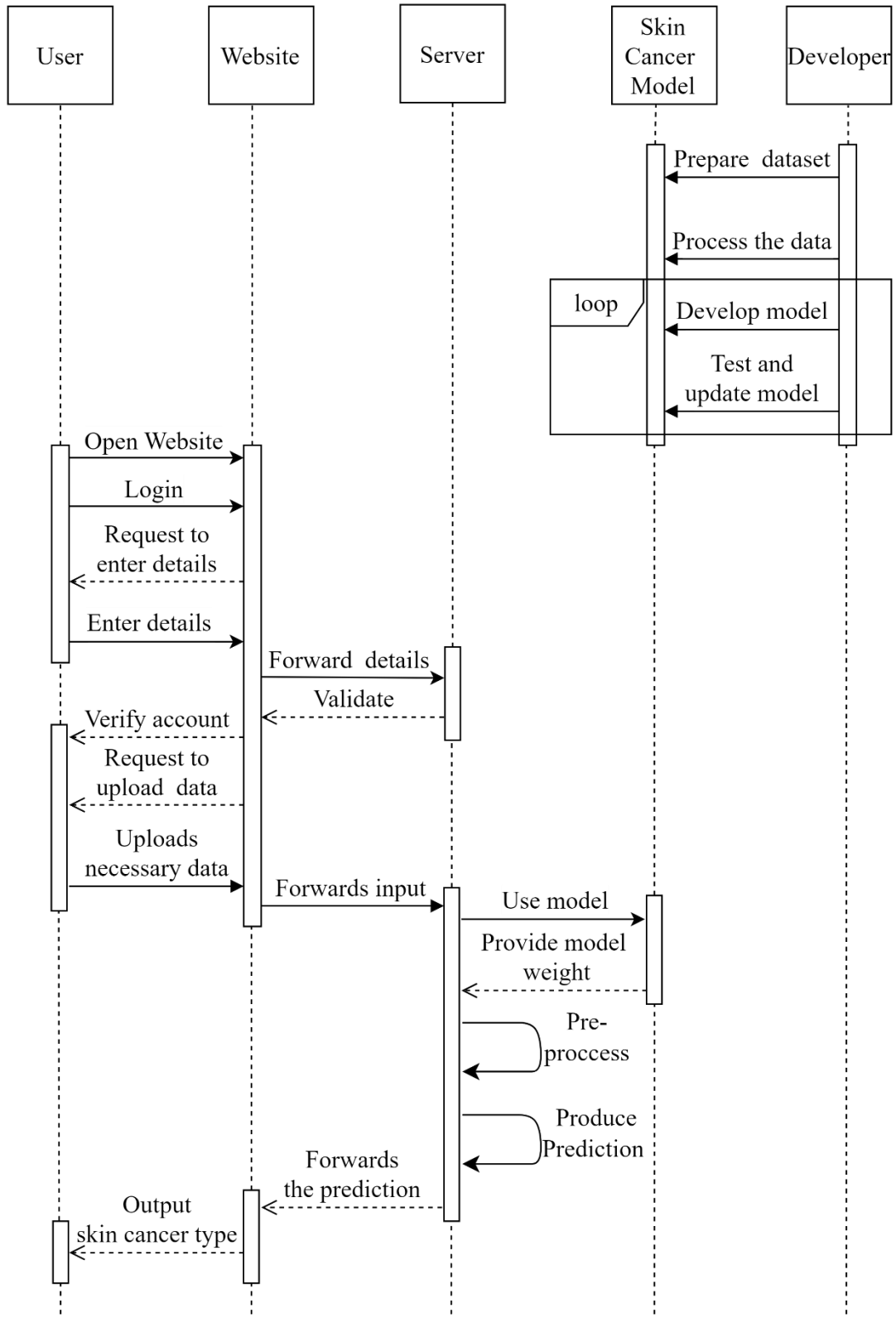


Figure 3.14: Sequence Diagram

3.9.3 Activity Diagram

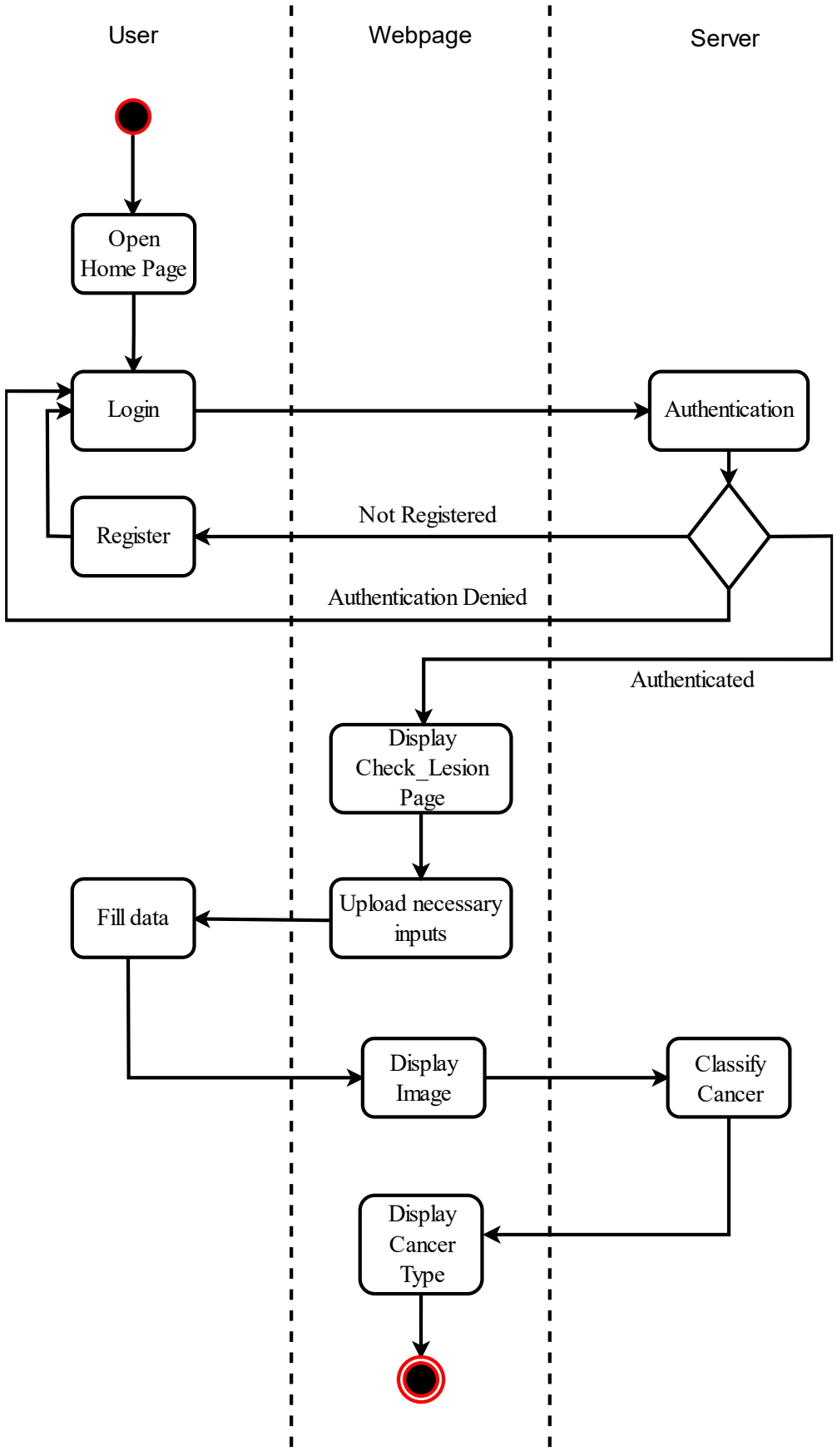


Figure 3.15: Activity Diagram

3.10 TOOLS USED

3.10.1 Python

Python is a programming language that is high-level, interpreted, and used for general-purpose programming tasks like for machine learning, data science, web development and AI. It also supports dynamic typing, is portable with any operating system, and has wide range of libraries. Python is used as the backbone programming language for our project and the entire code for the model has been written using Python.

3.10.2 Pandas

Pandas is an extremely powerful open-source library available in Python and is used for tasks that involve manipulation, analysis, and tabular data. We have used Pandas in our project to create DataFrames of CSV files and function of Pandas during handling missing data.

3.10.3 TensorFlow

TensorFlow is a well-liked and powerful open-source framework for machine learning developed by Google, that provides a thorough set of tools and functionalities for building, training, and deploying different types of machine learning models. In our project, TensorFlow has been used to perform flattening, pooling, and dropout.

3.10.4 NumPy

Released under BSD license, NumPy is an open-source software package which provides multi-dimensional arrays, array object, and matrices and various mathematical functions. In our project, NumPy has been used to convert images into array so that it can be processed by our model.

3.10.5 OpenCV

OpenCV, standing for Open-Source Computer Vision, is an open-source library that helps users in areas that deal with computer vision and processing images like loading, pre-processing and manipulating them and is used in various fields such as augmented reality, medical imaging, robotics, and deep learning. Since our project deals with

detecting skin cancer, the images were processed, loaded, scaled, augmented, resized and many other operations were formed on the images of skin, so OpenCV has played a very important role in our project.

3.10.6 Keras

Keras is a high-level open-source deep learning library written in Python, which provides interface for building, training and deploying deep learning models. Pre-trained models such as VGG16, ResNet, and InceptionV3, are also available in Keras which can be used as a tool for extracting features to be used in a wide range of computer vision tasks. To build our model, we have used EfficientNetB4 as the base model and it was imported through Keras library.

3.10.7 Matplotlib

Matplotlib is a library available in Python, that is used for converting our data into animated visualizations like graphs, charts, histograms, scatterplots, bar diagrams. One of its striking features is that it integrates well with other Python libraries to provide excellent functionalities, such as NumPy and Matplotlib are a good pair for numerical computing and, Pandas and Matplotlib are a good pair for manipulating data. In our project, Matplotlib has been used to plot the training and testing accuracy and validation graphs, confusion matrix and ROC-AUC curve.

3.10.8 GitHub and Git

Git, a distributed version control system, is a very important resource for people working on coding projects, as it allows multiple people to collaborate on a project, along with keeping track of progress and facilitating efficient teamwork. GitHub is an online platform that hosts Git repository and offers tools for developers to collaborate. Together, they have helped us, as a team to work on a project simultaneously, while keeping records of changes and facilitating efficient teamwork by creating GitHub repo and pulling and pushing the changes.

3.10.9 HTML, CSS, JavaScript and Flask

Our model on skin cancer detection and classification is going to be deployed through combination of HTML and CSS in frontend to design UI, JavaScript to add interactivity

and dynamic behavior to web pages, and Flask as a backend framework to handle requests from the frontend. This combination is a very powerful tool to deploy our model as a web application.

3.10.10 Firebase

Firebase is a comprehensive mobile and web application development platform provided by Google. It offers various tools and services to help developers build, improve, and grow their apps. Realtime Database is a NoSQL cloud database that stores data as JSON and synchronizes it across connected clients in real-time. Firebase Realtime Database is used to store user-related data such as username and email. Firebase Authentication allows users to securely sign in using their chosen authentication method.

3.10.11 Kaggle

Kaggle is an online platform that brings people who are interested in machine learning and data science together. In addition, it also has a vast dataset collection that people can incorporate in their ML and data science projects. For our project, we have used ISIC 2019 dataset from Kaggle.

3.10.12 Jupyter Notebook

Jupyter Notebook is an open-source, interactive, cell-based web application that allows us to create and share documents that helps to integrate executable code snippets arranged in the form of cells with explanatory text, visualizations. It is mainly used for machine learning and data science related tasks as it is compatible with popular machine learning frameworks like Pandas, Numpy, Matplotlib, Seaborn, Tensorflow, Keras. The entire code for the model has been developed in Jupyter Notebook environment.

3.11 VERIFICATION AND VALIDATION

While training the model, we can observe the performance by monitoring the accuracy and loss for both training and validation sets during each epoch generated by dividing using 80/20 ratio. The total number of epoch was set to 30 and as the batch size is set to 32 with the total number of training data in each epoch being (0.8 * 11097) =8877.6 ceiled to 8878, the number of steps per epoch becomes $\frac{8878}{32}$ =278 (ceiled). After the accuracy and losses are obtained, the further evaluation of the model using different evaluation metrices can easily be done.

While preparing the model for training, the values of hyperparameters play a very important role because these parameters govern behavior, learning, and optimization. The hyperparameters used in our model are:

- **Batch-size:** It specifies the number of samples processed in each iteration. It affects training time, memory and condition of overfitting as small batch size takes longer time to train but uses less memory in contrast to large batch size that may cause overfitting problem. We have defined batch size of 32.
- **Activation function and dropout:** Activation function like ReLU, sigmoid help to introduce non-linearity as most of the real-world data have non-linear relationship and we have used ReLU. Also, dropout helps in preventing overfitting by randomly dropping off some neurons, we have kept the dropout values around 0.1. In our case, when dropout value of around 0.3 was selected, underfitting was observed and less dropout value of 0.05 gave overfitting. So, the value of dropouts plays a very important role during the training process.
- **Learning Rate:** It helps in controlling the step size by which the weight is updated during optimization.
- **Optimizer:** It helps in updating the model parameters during training. We have used Adam optimizer.

```
Epoch 5: val_loss improved from 1.31796 to 1.28359, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf5
278/278 [==============================] - 1554s 6s/step - loss: 0.6639 - accuracy: 0.7408 - val_loss: 1.2836 - val_accuracy: 0.3257 - lr: 3.1000e-04
Epoch 6/30
278/278 [==============================] - ETA: 0s - loss: 0.5945 - accuracy: 0.7732
Epoch 6: val_loss improved from 1.28359 to 1.24953, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf5
278/278 [==============================] - 1548s 6s/step - loss: 0.5945 - accuracy: 0.7732 - val_loss: 1.2495 - val_accuracy: 0.3829 - lr: 3.1000e-04
Epoch 7/30
278/278 [==============================] - ETA: 0s - loss: 0.5028 - accuracy: 0.8113
...
Epoch 24/30
278/278 [==============================] - ETA: 0s - loss: 0.1307 - accuracy: 0.9481
Epoch 24: val_loss did not improve from 0.43855
278/278 [==============================] - 1536s 6s/step - loss: 0.1307 - accuracy: 0.9481 - val_loss: 0.4419 - val_accuracy: 0.8707 - lr: 8.8750e-07
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
Epoch 25/30
278/278 [==============================] - ETA: 0s - loss: 0.1345 - accuracy: 0.9474
Epoch 25: val_loss improved from 0.43855 to 0.43725, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf5
278/278 [==============================] - 1537s 6s/step - loss: 0.1345 - accuracy: 0.9474 - val_loss: 0.4373 - val_accuracy: 0.8680 - lr: 8.8750e-07
Epoch 26/30
278/278 [==============================] - ETA: 0s - loss: 0.1315 - accuracy: 0.9458
Epoch 26: val_loss improved from 0.43725 to 0.43690, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf5
278/278 [==============================] - 1534s 6s/step - loss: 0.1315 - accuracy: 0.9458 - val_loss: 0.4369 - val_accuracy: 0.8712 - lr: 8.8750e-07
```

Figure 3.16: Output Snippet of Model Training and Epochs

```
278/278 [==============================] - 298s 1s/step - loss: 0.1334 - accuracy: 0.9549
Training Loss: 0.13344310224056244
Training Accuracy: 0.9549397230148315
70/70 [==============================] - 79s 1s/step - loss: 0.4243 - accuracy: 0.8590
Testing Loss: 0.4243398904800415
Testing Accuracy: 0.8590090274810791
```

Figure 3.17: Training and Testing Accuracy and Loss

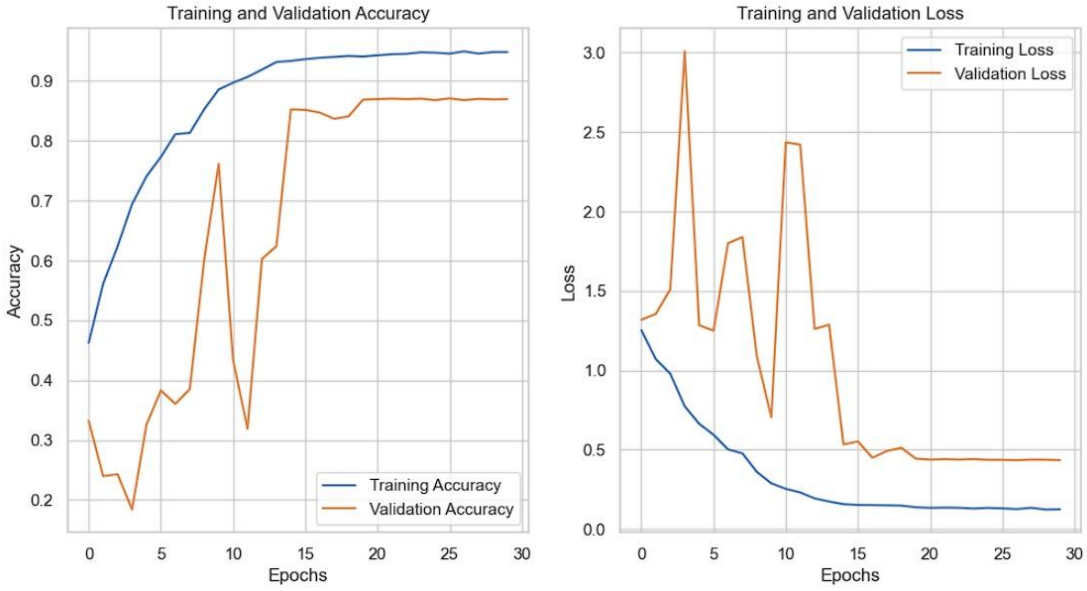


Figure 3.18: Accuracy and Loss Graph of Training and Validation

Accuracy and loss help to analyze any machine learning model. Accuracy refers to the model's ability to correctly classify the instances. Loss refers to the deviation of predicted value from the actual value. The accuracy and loss calculated on the data separated for training are called as training accuracy and training loss and the same

concept applies to validation data. We have trained our model on 8878 train data and 2219 validation data for over 30 epochs.

As seen in the above figure, the training accuracy increased from around 0.48 to 0.94 and the validation accuracy increased from 0.33 to 0.86. The high value of training accuracy indicates that our model is trained well and is distinguishing the skin cancer images with great accuracy. Also, the high value of validation accuracy indicates that our model is not only performing well on seen data; but also on unseen data, the model is able to classify the images. Observing the nature of the curve, we can conclude that our model is neither over-fitted as there is no sign that training loss is extremely high but the validation accuracy is extremely low, nor under-fitted as there is no sign of both the training and validation accuracies being low.

Also, as seen in the above figure, the training loss decreased from 1.3 to 0.12 and validation loss from 1.3 to 0.43. Both the values are quite low and they indicate that there is less difference in actual and predicted values of both the training and validation dataset. Till epoch 10, the values of accuracies and losses are fluctuating rapidly, which is normal during the first few epochs as the model is starting to learn and adjust weights and this initial phase can be described as fitting-in. After ten epochs, the values of accuracy are increasing and the value of loss is decreasing which is the desired trend.

Only accuracy and loss alone can't determine the performance of model and there are other metrices like precision, recall, F1-score, ROC-AUC Curve, Confusion matrix, Classification Report and many more tools.

3.12 Grad-CAM Heatmap Visualization of Model Learning from Skin Image

While the internal working of CNN model is hard to understand, and due to this nature, it is sometimes referred to as black-box model, there are some feature visualization techniques that provide some degree of interpreting the predictions and decisions. Grad-CAM is one of such techniques that can help us visualize the amount of contribution of the parts, features and patterns of input image contribute to the final prediction. First of all, the gradient is calculated with respect to feature map of the last convolutional layer, then they are global average pooled for the entire image to get a set of weights that

show importance of each feature map for the entire image. Then, a weighted sum of feature map is obtained by linear combination followed by ReLU to make sure that only important parts of image contribute to the prediction. At last, up-sampling is carried out to match the original dimension of image. Then, the heatmap can be laid over the original image to get a proper visualization. The cooler colors like blue or green highlight less important features whereas warmer colors like yellow or red highlight more important features.

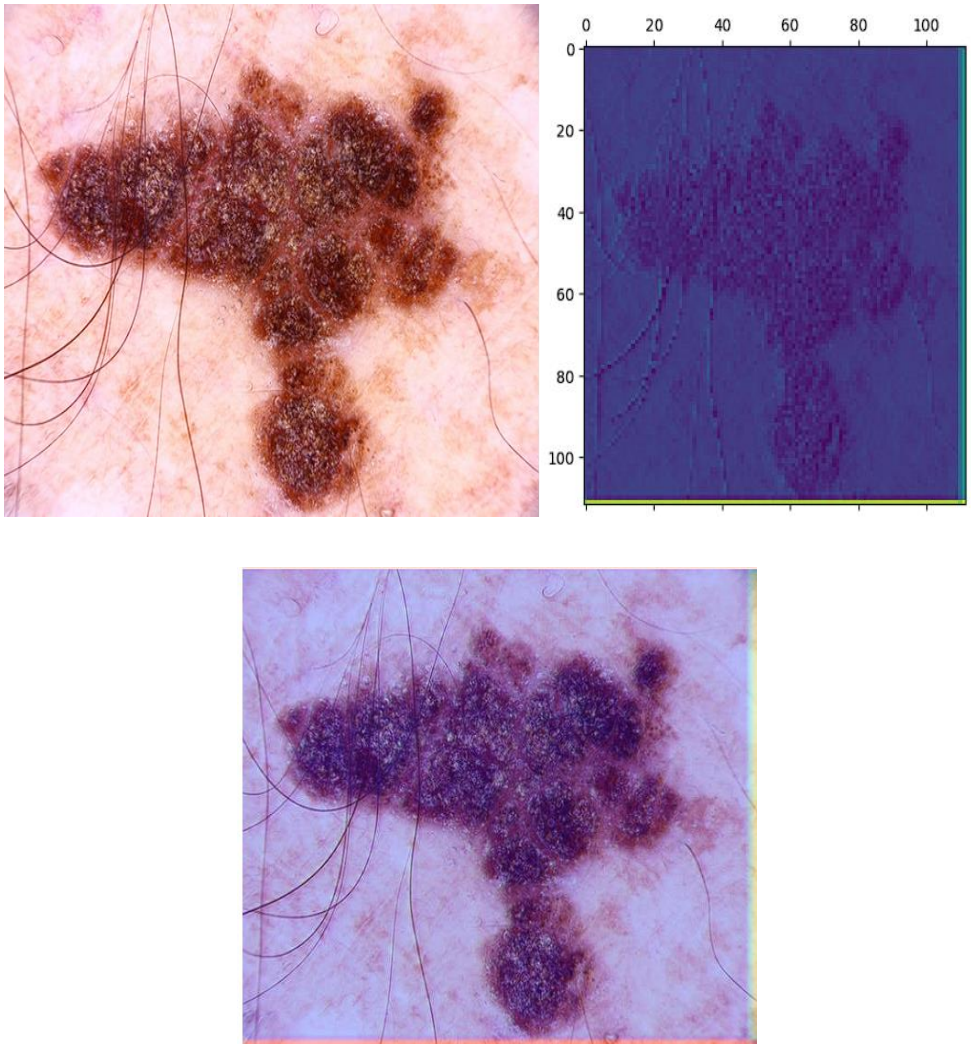


Figure 3.19: Grad-CAM Visualization Produced by Early Layer

The early layers of the model simply focus on finding the low-level features such as basic outline and edges which is quite clear from the above demonstration. It just provides visualization for how model process input at early stage and it doesn't capture the necessary high-level relevant features.

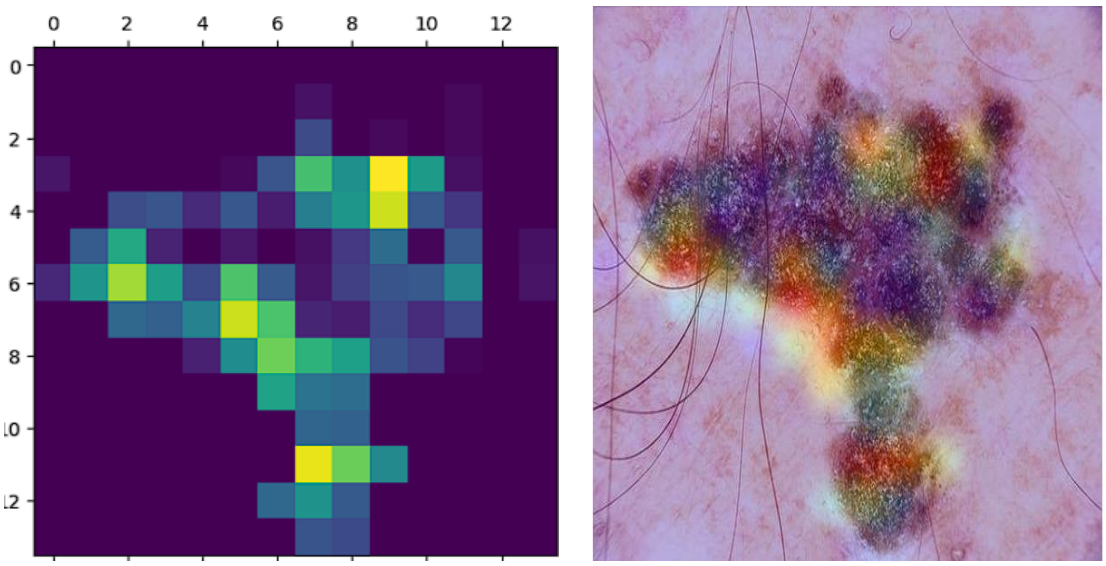


Figure 3.20: Grad-CAM Visualization Produced by Intermediate Layer

The intermediate layers of the model focus on finding slightly more complex features such as textures and specific patterns which is better at capturing features than early layers. It just provides a balanced visualization for both low and high-level features.

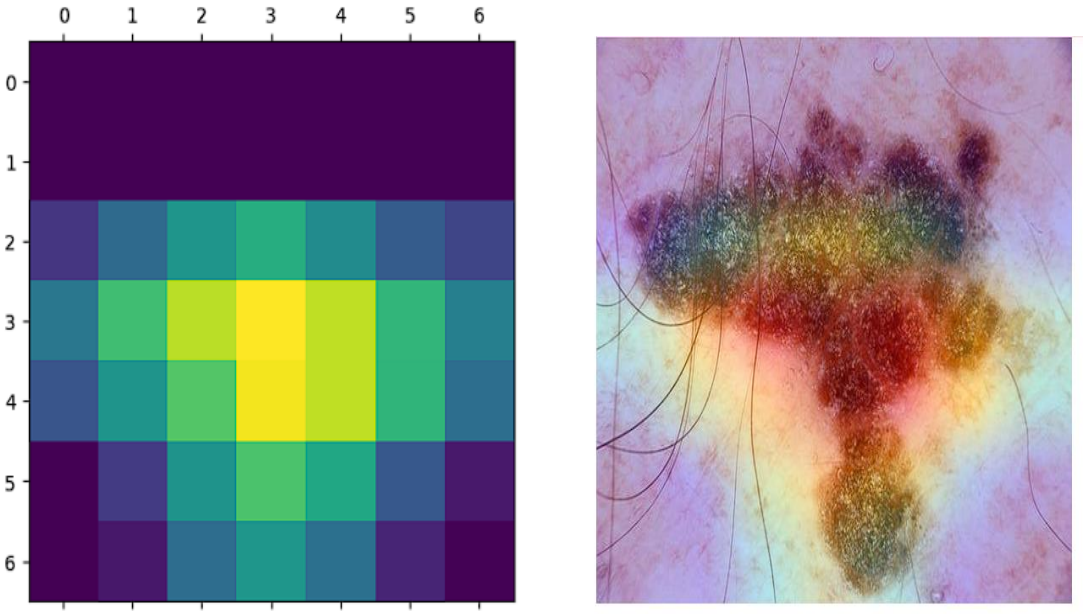


Figure 3.21: Grad-CAM Visualization Produced by Last Convolutional Layer

The later layers of the model focus on finding the high-level features such as the part of image that directly contributes most to the model's decision. Thus, in this way, the model learns to extract necessary features from the input image.

CHAPTER 4: EPILOGUE

4.1 RESULT AND DISCUSSSION

Our system is designed using ISIC-2019 dataset from Kaggle using EfficientB4 as the as discussed earlier and the following result is obtained:

Training Accuracy: 95.49%

Testing Accuracy: 85.90%

Training Loss: 0.1334

Testing Loss: 0.4243

4.1.1 Area Under ROC Curve (AUC-ROC)

ROC curve is a graphical tool used to evaluate the performance of binary or multi-class classification model and it measures model's ability to correctly classify True Positive Rate and to incorrectly classify False Positive Rate.

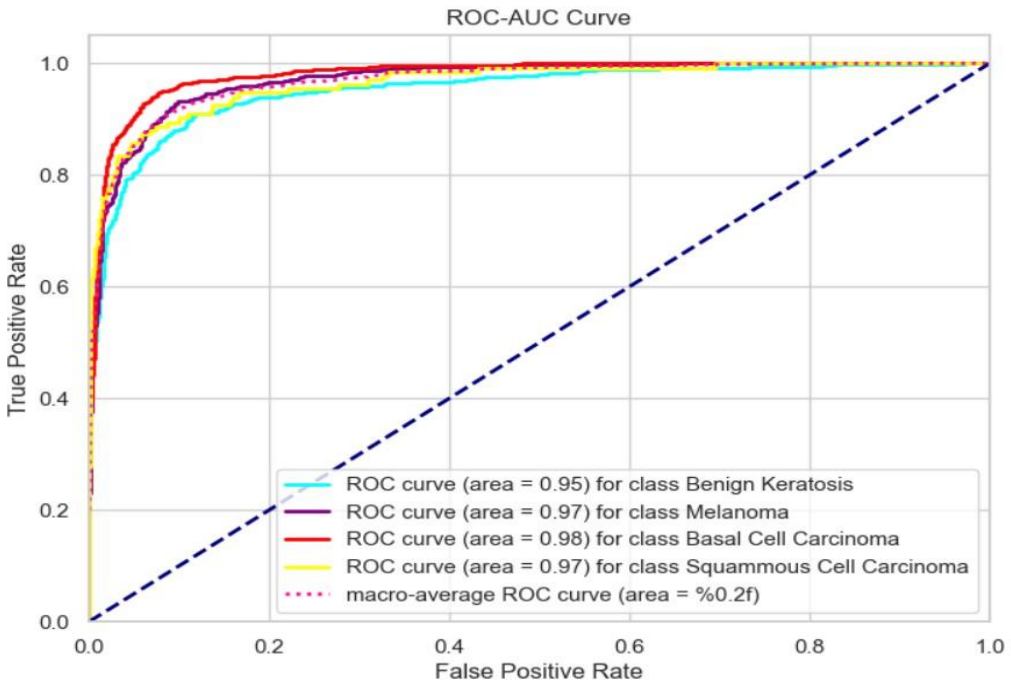


Figure 4.1: AUC-ROC

ROC- AUC is the area under the ROC curve, a number ranging from 0 to 1 indicating the model's ability to discriminate the data among the classes. The closer the value is

to 1, the better it is. In the above obtained curve, the blue dotted line from bottom left to top right corner represents the curve that would have been obtained by random guessing of the classes. So, the curves away from that line, inclining towards the top left corner are considered better. In our model's case, the curve for each classes have value greater than 0.95 and it illustrates that our model can classify instance into the true class with greater probability of being correct.

4.1.2 Confusion Matrix

The table used to describe the performance of a classification model is called as confusion matrix and it is a square matrix that shows the relation between predicted and actual values for each class, and has True Positive, True Negative, False Positive, False Negative as the main components.

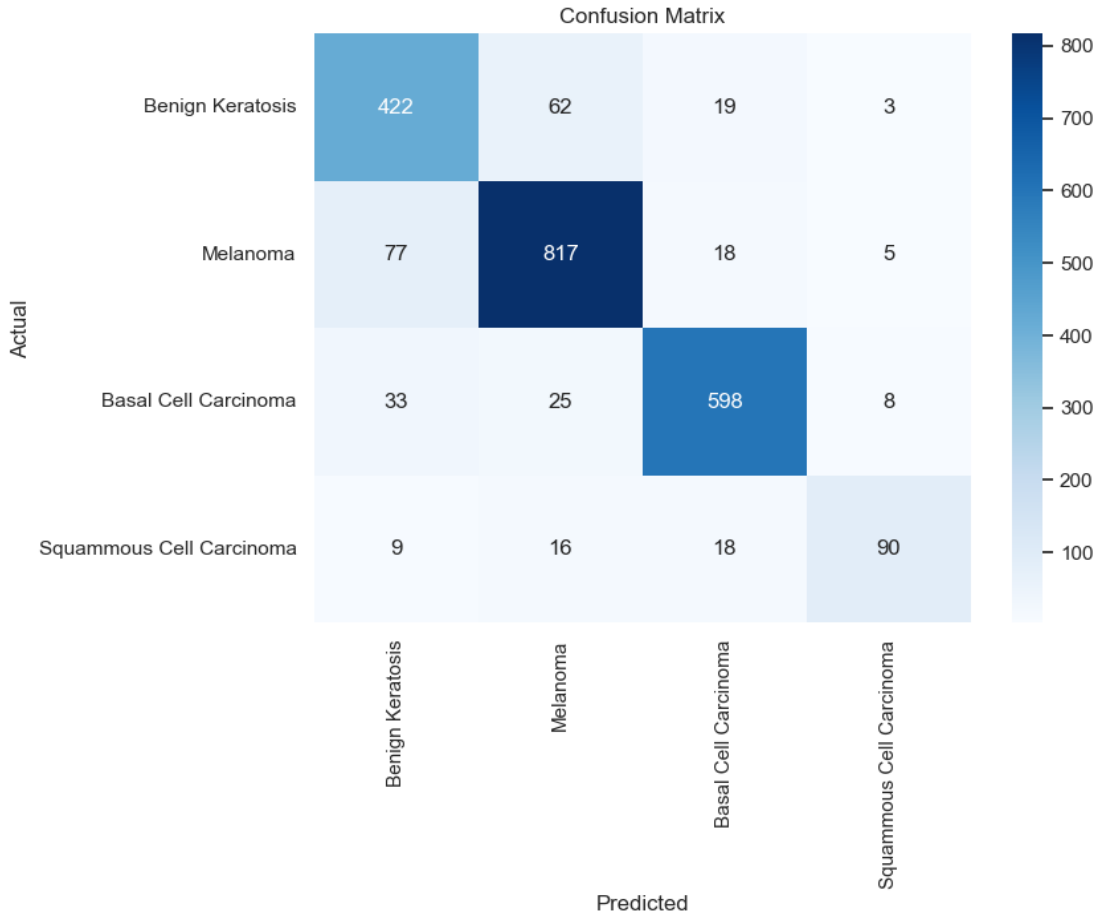


Figure 4.2: Confusion Matrix for Validation Data

The above figure shows confusion matrix plotted for validation dataset. A model can be said to be performing well if the values in the main diagonal that refer to true positive

values are maximum and there are minimum values elsewhere. For example, for melanoma class, from above confusion matrix, it can be said that out of total 917 samples, 817 are predicted as melanoma by our model, but the rest 100 samples are incorrectly predicted as other classes, like 77 as benign keratosis, 18 as BCC, and 5 as SCC. Considering the number of data available in each class, the obtained confusion matrix illustrates that our model is performing fine but not very good in case of Squamous Cell Carcinoma and availability of only few data samples in this category may be the cause of it.

4.1.3 Classification Report

Classification report is the report that gives summary of the primary metrices like precision, recall, f1-score to properly present the evaluation metrices for multiple classes.

	precision	recall	f1-score	support
Benign Keratosis	0.78	0.83	0.81	506
Melanoma	0.89	0.89	0.89	917
Basal Cell Carcinoma	0.92	0.90	0.91	664
Squammous Cell Carcinoma	0.85	0.68	0.75	133
accuracy			0.87	2220
macro avg	0.86	0.83	0.84	2220
weighted avg	0.87	0.87	0.87	2220

Figure 4.3: Classification Report for Validation Data

	precision	recall	f1-score	support
Benign Keratosis	0.91	0.93	0.92	45
Melanoma	0.84	0.96	0.90	45
Basal Cell Carcinoma	1.00	0.91	0.95	45
Squmaous Cell Carcinoma	0.98	0.91	0.94	45
accuracy			0.93	180
macro avg	0.93	0.93	0.93	180
weighted avg	0.93	0.93	0.93	180

Figure 4.4: Classification Report for Test Data

The values of precision, recall and f1-score that are close to 1 are better than the values close to 0 as nearer the value to one, the better the model is at distinguishing the data correctly among different classes. In the classification report for test data, all the values are in the range of 0.90 which indicates that our model is performing well on test data.

- **Accuracy**

Accuracy gives the overall measure of how well the model is performing. It refers to the proportion of correctly classified instances.

Mathematically,

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

- **Precision**

Precision refers to the ratio of correctly predicted positive instances to all the instances predicted as positive by the model. It describes the model's ability to avoid false positives.

Mathematically,

$$\text{Precision} = \frac{TP}{TP+FP}$$

- **Recall / Sensitivity / True Positive Rate**

Recall refers to the ratio of correctly predicted positive instances to all the actual positive instances. It describes the model's ability to find all the positive instances.

Mathematically,

$$\text{Recall} = \frac{TP}{TP+FN}$$

- **f1-Score**

It is the metric that gives balance between precision and recall.

Mathematically,

$$\text{f1-Score} = \frac{2*(\text{Precision}*\text{Recall})}{\text{Precision}+\text{Recall}}$$

- **Support**

Support can be defined as the number of data available in that particular class.

- **Macro-average**

Macro-average is the metric that represents unweighted mean of all classes.

Mathematically,

$$\text{Precision Macro-average} = \frac{\text{Precision 1+Precision2+Precision3+Precision4}}{4}$$

Similar formula for Recall and f1-Score Macro-average.

- **Weighted-average**

Weighted average is the metric that represents weighted mean of all classes.

$$\text{Weighted average} = \frac{\text{Precision 1x Suppot1+Precision2X Support2+Precision3XSupport3+Precision4XSupport4}}{\text{Total Support}};$$

Similar formula for weighted recall and f1-Score.

The values of precision, recall, f1-score, weighted-average and macro-average can all be obtained from the classification matrix.

Taking example of melanoma class in validation data having total support of 917, the following conclusion can be made from the classification matrix:

True Positive=815

False Positive= (62+25+16) = 103

False Negative= (77+ 18+5) = 100

So, calculations can be made for the evaluation metrics by using formula as:

$$\text{Precision} = \frac{815}{815+103} = 0.89$$

$$\text{Recall} = \frac{815}{815+100} = 0.89$$

$$\text{F1-Score} = \frac{2*(0.89*0.89)}{0.89+0.89} = 0.89$$

Similarly, the metrices for remaining classes can also be calculated in the same way.

$$\text{And overall accuracy for validation set} = \frac{422+817+598+90}{506+917+664+133} = 0.87$$

4.1.4 Output Accuracy and Loss Obtained at Various Hyperparameters

Dropout: 0.3

```
Epoch 1/30
278/278 [==============================] - ETA: 0s - loss: 1.6535 - accuracy: 0.1275
Epoch 1: val_loss improved from inf to 1.38781, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\hhghh.hdf5
C:\Users\Asus\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\training.py:3079: UserWarning: You are saving your model as an HDF5 file vi
a `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
278/278 [==============================] - 1589s 6s/step - loss: 1.6535 - accuracy: 0.1275 - val_loss: 1.3878 - val_accuracy: 0.2279 - lr: 0.0010
Epoch 2/30
278/278 [==============================] - ETA: 0s - loss: 1.3865 - accuracy: 0.2330
Epoch 2: val_loss improved from 1.38781 to 1.38737, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\hhghh.hdf5
278/278 [==============================] - 1549s 6s/step - loss: 1.3865 - accuracy: 0.2330 - val_loss: 1.3874 - val_accuracy: 0.0599 - lr: 0.0010
Epoch 3/30
278/278 [==============================] - ETA: 0s - loss: 1.3867 - accuracy: 0.1460
Epoch 3: val_loss did not improve from 1.38737
278/278 [==============================] - 1529s 6s/step - loss: 1.3867 - accuracy: 0.1460 - val_loss: 1.3875 - val_accuracy: 0.2279 - lr: 0.0010
Epoch 4/30
278/278 [==============================] - ETA: 0s - loss: 1.3865 - accuracy: 0.3341
Epoch 4: val_loss improved from 1.38737 to 1.38708, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\hhghh.hdf5
278/278 [==============================] - 1531s 6s/step - loss: 1.3865 - accuracy: 0.3341 - val_loss: 1.3871 - val_accuracy: 0.2279 - lr: 0.0010
Epoch 5/30
278/278 [==============================] - ETA: 0s - loss: 1.3864 - accuracy: 0.1067
Epoch 5: val_loss improved from 1.38708 to 1.38658, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\hhghh.hdf5
278/278 [==============================] - 1535s 6s/step - loss: 1.3864 - accuracy: 0.1067 - val_loss: 1.3866 - val_accuracy: 0.2991 - lr: 0.0010
Epoch 6/30
278/278 [==============================] - ETA: 0s - loss: 1.3865 - accuracy: 0.2250
Epoch 6: val_loss improved from 1.38658 to 1.38612, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\hhghh.hdf5
278/278 [==============================] - 1534s 6s/step - loss: 1.3865 - accuracy: 0.2250 - val_loss: 1.3861 - val_accuracy: 0.2279 - lr: 0.0010
Epoch 7/30
278/278 [==============================] - ETA: 0s - loss: 1.3865 - accuracy: 0.1263
Epoch 7: val_loss did not improve from 1.38612
278/278 [==============================] - 1533s 6s/step - loss: 1.3865 - accuracy: 0.1263 - val_loss: 1.3865 - val_accuracy: 0.2279 - lr: 0.0010
Epoch 8/30
278/278 [==============================] - ETA: 0s - loss: 1.3866 - accuracy: 0.2582
Epoch 8: val_loss improved from 1.38612 to 1.38522, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\hhghh.hdf5
278/278 [==============================] - 1538s 6s/step - loss: 1.3866 - accuracy: 0.2582 - val_loss: 1.3852 - val_accuracy: 0.2991 - lr: 0.0010
Epoch 9/30
278/278 [==============================] - ETA: 0s - loss: 1.3864 - accuracy: 0.2797
Epoch 9: val_loss did not improve from 1.38522
278/278 [==============================] - 1530s 6s/step - loss: 1.3864 - accuracy: 0.2797 - val_loss: 1.3867 - val_accuracy: 0.2279 - lr: 0.0010
Epoch 10/30
278/278 [==============================] - ETA: 0s - loss: 1.3865 - accuracy: 0.1297
278/278 [==============================] - 1531s 6s/step - loss: 1.3865 - accuracy: 0.1297 - val_loss: 1.3867 - val_accuracy: 0.2279 - lr: 0.0010
Epoch 11/30
278/278 [==============================] - ETA: 0s - loss: 1.3864 - accuracy: 0.1543
Epoch 11: val_loss did not improve from 1.38522
278/278 [==============================] - 1536s 6s/step - loss: 1.3864 - accuracy: 0.1543 - val_loss: 1.3864 - val_accuracy: 0.2279 - lr: 3.1000e-04
Epoch 12/30
278/278 [==============================] - ETA: 0s - loss: 1.3865 - accuracy: 0.2042
Epoch 12: val_loss did not improve from 1.38522

Epoch 12: ReduceLROnPlateau reducing learning rate to 9.61000009556301e-05.
278/278 [==============================] - 1534s 6s/step - loss: 1.3865 - accuracy: 0.2042 - val_loss: 1.3867 - val_accuracy: 0.2279 - lr: 3.1000e-04
Epoch 13/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.0589
Epoch 13: val_loss did not improve from 1.38522
278/278 [==============================] - 1537s 6s/step - loss: 1.3863 - accuracy: 0.0589 - val_loss: 1.3867 - val_accuracy: 0.0599 - lr: 9.6100e-05
Epoch 14/30
278/278 [==============================] - ETA: 0s - loss: 1.3864 - accuracy: 0.2344
Epoch 14: val_loss did not improve from 1.38522

Epoch 14: ReduceLROnPlateau reducing learning rate to 2.9791001288685948e-05.
278/278 [==============================] - 1533s 6s/step - loss: 1.3864 - accuracy: 0.2344 - val_loss: 1.3866 - val_accuracy: 0.2279 - lr: 9.6100e-05
Epoch 15/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.2389
Epoch 15: val_loss did not improve from 1.38522
278/278 [==============================] - 1532s 6s/step - loss: 1.3863 - accuracy: 0.2389 - val_loss: 1.3866 - val_accuracy: 0.2279 - lr: 2.9791e-05
Epoch 16/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.1408
Epoch 16: val_loss did not improve from 1.38522

Epoch 16: ReduceLROnPlateau reducing learning rate to 9.235210309270769e-06.
278/278 [==============================] - 1552s 6s/step - loss: 1.3863 - accuracy: 0.1408 - val_loss: 1.3866 - val_accuracy: 0.2279 - lr: 2.9791e-05
Epoch 17/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.1007
Epoch 17: val_loss did not improve from 1.38522
278/278 [==============================] - 1572s 6s/step - loss: 1.3863 - accuracy: 0.1007 - val_loss: 1.3866 - val_accuracy: 0.2279 - lr: 9.2352e-06
Epoch 18/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.2385
Epoch 18: val_loss did not improve from 1.38522

Epoch 18: ReduceLROnPlateau reducing learning rate to 2.8629151620407354e-06.
278/278 [==============================] - 1546s 6s/step - loss: 1.3863 - accuracy: 0.2385 - val_loss: 1.3865 - val_accuracy: 0.2279 - lr: 9.2352e-06
```

```
Epoch 21/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.2387
Epoch 21: val_loss did not improve from 1.38522
278/278 [==============================] - 1529s 5s/step - loss: 1.3863 - accuracy: 0.2387 - val_loss: 1.3866 - val_accuracy: 0.2279 - lr: 8.8750e-07
Epoch 22/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.2387
Epoch 22: val_loss did not improve from 1.38522

Epoch 22: ReduceLROnPlateau reducing learning rate to 2.751261433786567e-07.
278/278 [==============================] - 1528s 5s/step - loss: 1.3863 - accuracy: 0.2387 - val_loss: 1.3866 - val_accuracy: 0.2279 - lr: 8.8750e-07
Epoch 23/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.2386
Epoch 23: val_loss did not improve from 1.38522
278/278 [==============================] - 1511s 5s/step - loss: 1.3863 - accuracy: 0.2386 - val_loss: 1.3866 - val_accuracy: 0.2279 - lr: 2.7513e-07
Epoch 24/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.2386
Epoch 24: val_loss did not improve from 1.38522

Epoch 24: ReduceLROnPlateau reducing learning rate to 1e-07.
278/278 [==============================] - 1531s 6s/step - loss: 1.3863 - accuracy: 0.2386 - val_loss: 1.3866 - val_accuracy: 0.2279 - lr: 2.7513e-07
Epoch 25/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.2386
Epoch 25: val_loss did not improve from 1.38522
278/278 [==============================] - 1528s 5s/step - loss: 1.3863 - accuracy: 0.2386 - val_loss: 1.3866 - val_accuracy: 0.2279 - lr: 1.0000e-07
Epoch 26/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.2386
Epoch 26: val_loss did not improve from 1.38522
278/278 [==============================] - 1529s 5s/step - loss: 1.3863 - accuracy: 0.2386 - val_loss: 1.3866 - val_accuracy: 0.2279 - lr: 1.0000e-07
Epoch 27/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.2386
Epoch 27: val_loss did not improve from 1.38522
278/278 [==============================] - 1534s 5s/step - loss: 1.3863 - accuracy: 0.2386 - val_loss: 1.3866 - val_accuracy: 0.2279 - lr: 1.0000e-07
Epoch 28/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.2386
Epoch 28: val_loss did not improve from 1.38522
278/278 [==============================] - 1535s 6s/step - loss: 1.3863 - accuracy: 0.2386 - val_loss: 1.3866 - val_accuracy: 0.2279 - lr: 1.0000e-07
Epoch 29/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.2386
Epoch 29: val_loss did not improve from 1.38522
278/278 [==============================] - 1532s 6s/step - loss: 1.3863 - accuracy: 0.2386 - val_loss: 1.3866 - val_accuracy: 0.2279 - lr: 1.0000e-07
Epoch 30/30
278/278 [==============================] - ETA: 0s - loss: 1.3863 - accuracy: 0.2386
```

Figure 4.5 Output Snippet of Model Training and Epochs at Dropout 0.3

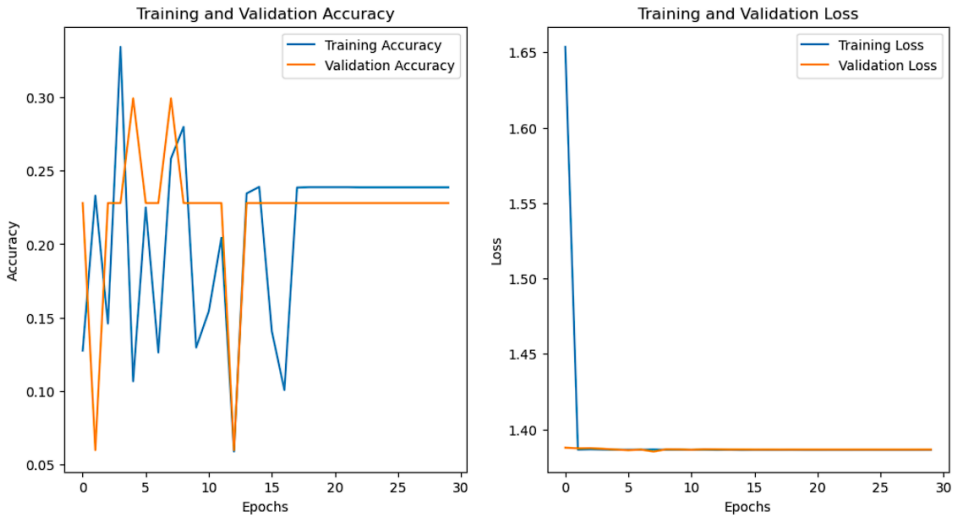


Figure 4.6 Accuracy and Loss Graph of Training and Validation at Dropout 0.3

Training Accuracy: 23.86%

Validation Accuracy: 22.79%

Training Loss: 1.3863

Validation Loss: 1.3866

It is observed that at dropout 0.3, the training accuracy increases from 12.75% in the first epoch to around 23.86% towards the final epoch and the validation accuracy

remains as 22.79% in most of the epochs with some fluctuations. Similarly, the training loss decreases from 1.6535 in the first epoch to around 1.3863 towards the final epochs and the validation loss remains constant at around 1.3866. Both the training and validation accuracies are low, and the losses are quite high. It can be said to be the case of underfitting as in this dropout value, the model is neither trained well nor can distinguish between the validation dataset well, it has poor performance on both training and validation. Underfitting occurs when a model is too simplistic to capture the underlying structure of the data, leading to poor performance on both the training and validation sets. Other dropout rates above 0.3 like 0.5,0.7 also led to underfitting. Dropout regularization is intended to prevent overfitting, but too high of a dropout rate can also lead to underfitting. So, for the next epoch, a dropout of 0.05 was selected.

Dropout: 0.05

```
C:\Users\Asus\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\training.py:3079: UserWarning: You are saving your model as an HDF5 file vi
a `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(

278/278 [==============================] - 1580s 6s/step - loss: 1.3180 - accuracy: 0.4529 - val_loss: 1.7992 - val_accuracy: 0.2022 - lr: 0.0010
Epoch 2/30
278/278 [==============================] - ETA: 0s - loss: 1.2290 - accuracy: 0.5689
Epoch 2: val_loss improved from 1.7992 to 1.6503, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1548s 6s/step - loss: 1.2290 - accuracy: 0.5689 - val_loss: 1.6503 - val_accuracy: 0.2067 - lr: 0.0010
Epoch 3/30
278/278 [==============================] - ETA: 0s - loss: 1.3293 - accuracy: 0.4745
Epoch 3: val_loss improved from 1.6503 to 1.6200, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1546s 6s/step - loss: 1.3293 - accuracy: 0.4745 - val_loss: 1.6200 - val_accuracy: 0.2256 - lr: 0.0010
Epoch 4/30
278/278 [==============================] - ETA: 0s - loss: 0.8212 - accuracy: 0.5870
Epoch 4: val_loss improved from 1.6200 to 1.5309, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1547s 6s/step - loss: 0.8011 - accuracy: 0.6209 - val_loss: 1.5309 - val_accuracy: 0.2345 - lr: 0.0010
Epoch 5/30
278/278 [==============================] - ETA: 0s - loss: 0.8011 - accuracy: 0.6209
Epoch 5: val_loss improved from 1.5309 to 1.2039, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1547s 6s/step - loss: 0.8011 - accuracy: 0.6209 - val_loss: 1.2039 - val_accuracy: 0.2378 - lr: 0.0010
Epoch 6/30
278/278 [==============================] - ETA: 0s - loss: 0.8199 - accuracy: 0.6166
Epoch 6:  val_loss improved from 1.2039 to 1.1135, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1549s 6s/step - loss: 0.8199 - accuracy: 0.6166- val_loss: 1.1135- val_accuracy: 0.2756 - lr: 0.0010
Epoch 7/30
278/278 [==============================] - ETA: 0s - loss: 0.7023 - accuracy: 0.6612
Epoch 7: val_loss did not improve from 1.1135
278/278 [==============================] - 1549s 6s/step - loss: 0.7023 - accuracy: 0.6612 - val_loss: 2.0135 - val_accuracy: 0.2209 - lr: 0.0010
Epoch 8/30
278/278 [==============================] - ETA: 0s - loss: 0.6929 - accuracy: 0.6956
Epoch 8: val_loss did not improve from 1.1135

Epoch 8: ReduceLROnPlateau reducing learning rate to 0.0003100000147242099.
278/278 [==============================] - 1548s 6s/step - loss: 0.6929 - accuracy: 0.6956 - val_loss: 1.9037 - val_accuracy: 0.2256 - lr: 0.0010
Epoch 9/30
278/278 [==============================] - ETA: 0s - loss: 0.7130 - accuracy: 0.6790
Epoch 9: val_loss improved from 1.1135 to 1.0092, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1550s 6s/step - loss: 0.7130 - accuracy: 0.6790 - val_loss: 1.0092 - val_accuracy: 0.2978 - lr: 3.1000e-04
Epoch 10/30
278/278 [==============================] - ETA: 0s - loss: 0.6632 - accuracy: 0.7367
Epoch 10: val_loss improved from 1.0092 to 0.9973, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1545s 6s/step - loss: 0.6632 - accuracy: 0.7367 - val_loss: 0.9973 - val_accuracy: 0.3398 - lr: 3.1000e-04
```

```
Epoch 10: val_loss improved from 1.0092 to 0.9973, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1545s 6s/step - loss: 0.6632 - accuracy: 0.7367 - val_loss: 0.9973 - val_accuracy: 0.3398 - lr: 3.1000e-04
Epoch 11/30
278/278 [==============================] - ETA: 0s - loss: 0.6903 - accuracy: 0.6903
Epoch 11: val_loss improved from 0.9973 to 0.9792, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1541s 6s/step - loss: 0.6903 - accuracy: 0.6903 - val_loss: 0.9792 - val_accuracy: 0.3456 - lr: 3.1000e-04
Epoch 12/30
278/278 [==============================] - ETA: 0s - loss: 0.6102 - accuracy: 0.7755
Epoch 12: val_loss improved from 0.9792 to 0.9510, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1538s 6s/step - loss: 0.6102 - accuracy: 0.7755 - val_loss: 0.9510 - val_accuracy: 0.3784 - lr: 3.1000e-04
Epoch 13/30
278/278 [==============================] - ETA: 0s - loss: 0.5933 - accuracy: 0.7986
Epoch 13: val_loss did not improve from 0.9510
278/278 [==============================] - 1541s 6s/step - loss: 0.5933 - accuracy: 0.7986 - val_loss: 0.9669 - val_accuracy: 0.3609 - lr: 3.1000e-04
Epoch 14/30
278/278 [==============================] - ETA: 0s - loss: 0.4221 - accuracy: 0.8103
Epoch 14: val_loss improved from 0.9669 to 0.9493, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1542s 6s/step - loss: 0.4221 - accuracy: 0.8103 - val_loss: 0.9493 - val_accuracy: 0.3789 - lr: 3.1000e-04
Epoch 15/30
278/278 [==============================] - ETA: 0s - loss: 0.3973 - accuracy: 0.8007
Epoch 15: val_loss improved from 0.9493 to 0.9401, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1540s 6s/step - loss: 0.3973 - accuracy: 0.8007- val_loss: 0.9401 - val_accuracy: 0.3976- lr: 3.1000e-04
Epoch 16/30
278/278 [==============================] - ETA: 0s - loss: 0.3521 - accuracy: 0.8188
Epoch 16: val_loss improved from 0.9401 to 0.9102, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1539s 6s/step - loss: 0.3521 - accuracy: 0.8188 - val_loss: 0.9102 - val_accuracy: 0.4231 - lr: 3.1000e-04
Epoch 17/30
278/278 [==============================] - ETA: 0s - loss: 0.3109 - accuracy: 0.8204
Epoch 17: val_loss did not improve from 0.9102
278/278 [==============================] - 1535s 6s/step - loss: 0.3109 - accuracy: 0.8204 - val_loss: 0.9372 - val_accuracy: 0.3908 - lr: 3.1000e-04
Epoch 18/30
278/278 [==============================] - ETA: 0s - loss: 0.3089- accuracy: 0.8277
Epoch 18: val_loss did not improve from 0.9102

Epoch 18: ReduceLROnPlateau reducing learning rate to 9.61000009556301e-05.
278/278 [==============================] - 1541s 6s/step - loss: 0.3089 - accuracy: 0.8277 - val_loss: 0.9372 - val_accuracy: 0.3907 - lr: 9.6100e-05
Epoch 19/30
278/278 [==============================] - ETA: 0s - loss: 0.3989 - accuracy: 0.8044

Epoch 19/30
278/278 [==============================] - ETA: 0s - loss: 0.3989 - accuracy: 0.8044
Epoch 19: val_loss improved from 0.9102 to 0.8823, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1537s 6s/step - loss: 0.3989 - accuracy: 0.9563 - val_loss: 0.4614 - val_accuracy: 0.8662 - lr: 9.6100e-05
Epoch 20/30
278/278 [==============================] - ETA: 0s - loss: 0.3989 - accuracy: 0.8044
Epoch 20: val_loss improved from 0.8823 to 0.8709, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1541s 6s/step - loss: 0.2811 - accuracy: 0.8367 - val_loss: 0.8709 - val_accuracy: 0.4657 - lr: 9.6100e-05
Epoch 21/30
278/278 [==============================] - ETA: 0s - loss: 0.2823 - accuracy: 0.8366
Epoch 21: val_loss did not improve from 0.8709
278/278 [==============================] - 1539s 6s/step - loss: 0.2823 - accuracy: 0.8366 - val_loss: 0.8729 - val_accuracy: 0.4655 - lr: 9.6100e-05
Epoch 22/30
278/278 [==============================] - ETA: 0s - loss: 0.2902 - accuracy: 0.8332
Epoch 22: val_loss did not improve from 0.8709

Epoch 22: ReduceLROnPlateau reducing learning rate to 2.9791001288685948e-05.
278/278 [==============================] - 1548s 6s/step - loss: 0.2902  - accuracy: 0.8332 - val_loss: 0.9137 - val_accuracy: 0.4265 - lr: 9.6100e-05
Epoch 23/30
278/278 [==============================] - ETA: 0s - loss: 0.2623 - accuracy: 0.8496
Epoch 23: val_loss improved from 0.8709 to 0.8323, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1552s 6s/step - loss: 0.2623 - accuracy: 0.8496 - val_loss: 0.8323 - val_accuracy: 0.4976 - lr: 2.9791e-05
Epoch 24/30
278/278 [==============================] - ETA: 0s - loss: 0.2612 - accuracy: 0.8498
Epoch 24: val_loss improved from 0.8323 to 0.8293, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1611s 6s/step - loss: 0.2612 - accuracy: 0.8498 - val_loss: 0.8293 - val_accuracy: 0.5312 - lr: 2.9791e-05
Epoch 25/30
278/278 [==============================] - ETA: 0s - loss: 0.2583 - accuracy: 0.8507
Epoch 25: val_loss improved from 0.8293 to 0.8287, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1914s 7s/step - loss: 0.2583  - accuracy: 0.8507 - val_loss: 0.8287 - val_accuracy: 0.5345 - lr: 2.9791e-05
Epoch 26/30
278/278 [==============================] - ETA: 0s - loss: 0.2509 - accuracy: 0.8567
Epoch 26: val_loss improved from 0.8287 to 0.8104, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1901s 7s/step - loss: 0.2509 - accuracy: 0.8567 - val_loss: 0.8104 - val_accuracy: 0.5677 - lr: 2.9791e-05
Epoch 27/30
278/278 [==============================] - ETA: 0s - loss: 0.2093 - accuracy: 0.8711
Epoch 27: val_loss improved from 0.8104 to 0.8103, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\overfitt0.05.hdf5
278/278 [==============================] - 1539s 6s/step - loss: 0.2093 - accuracy: 0.8711 - val_loss: 0.8103 - val_accuracy: 0.5678 - lr: 2.9791e-05

Epoch 28/30
278/278 [==============================] - ETA: 0s - loss: 0.2134 - accuracy: 0.8602
Epoch 28: val_loss did not improve from 0.8103
278/278 [==============================] - 1548s 6s/step - loss: 0.2134  - accuracy: 0.8602 - val_loss: 0.8182 - val_accuracy: 0.5643 - lr: 2.9791e-05
Epoch 29/30
278/278 [==============================] - ETA: 0s - loss: 0.2134 - accuracy: 0.8603
Epoch 29: val_loss did not improve from 0.8103

Epoch 29: ReduceLROnPlateau reducing learning rate to 9.235210309270769e-06.
278/278 [==============================] - 1548s 6s/step - loss: 0.2134  - accuracy: 0.8603 - val_loss: 0.8177 - val_accuracy: 0.5655 - lr: 2.9791e-05
Epoch 30/30
278/278 [==============================] - ETA: 0s - loss: 0.2134 - accuracy: 0.8602
Epoch 30: val_loss did not improve from 0.8103
278/278 [==============================] - 1548s 6s/step - loss: 0.2134  - accuracy: 0.8602 - val_loss: 0.8175 - val_accuracy: 0.5654 - lr: 9.2352e-06
```

Figure 4.7 Output Snippet of Model Training and Epochs at Dropout 0.05

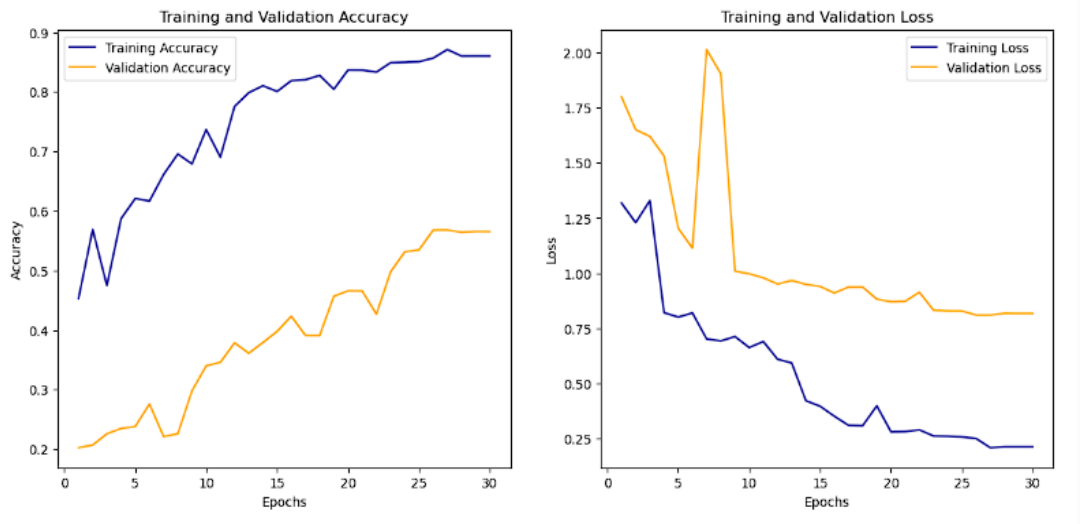


Figure 4.8 Accuracy and Loss Graph of Training and Validation at Dropout 0.05

Training Accuracy: 86.02%

Validation Accuracy: 56.54%

Training Loss: 0.2134

Validation Loss: 0.8175

It is observed that at dropout 0.05, the training accuracy increases from 45.29% in the first epoch to around 86.02% towards the final epoch and the validation accuracy increases from 20.22% to 56.54%. Similarly, the training loss decreases from 1.3180 in the first epoch to around 0.2134 towards the final epochs and the validation loss decreases from 1.7992 to 0.8175. Here, the training accuracy is quite high so it can be inferred that the model is learning very well from the training dataset. But the gap between the training and validation accuracy is very high, around 30%, and the validation accuracy is quite low. It can be said to be the case of overfitting as in this dropout value, the model is trained well but fails to generalize to unseen data, leading to poor performance on the validation or test set, suggesting that it might have memorized the training examples instead of learning the underlying patterns. Overfitting occurs when a machine learning model learns the training data too well, capturing noise or random fluctuations in the data as if they were genuine patterns. And other dropout rates less than 0.05 like 0.01 or no dropout also led to overfitting. Thus, from before, it was observed that dropouts above 0.3 led to underfitting and dropouts below 0.05 led to overfitting. So, for the next epoch, a dropout of 0.1 was selected.

Dropout: 0.1

```
Epoch 1/30
278/278 [==============================] - ETA: 0s - loss: 1.2528 - accuracy: 0.4627
Epoch 1: val_loss improved from inf to 1.31796, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf5
C:\Users\Asus\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\training.py:3079: UserWarning: You are saving your model as an HDF5 file vi
a `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
278/278 [==============================] - 1613s 6s/step - loss: 1.2528 - accuracy: 0.4627 - val_loss: 1.3180 - val_accuracy: 0.3320 - lr: 0.0010
Epoch 2/30
278/278 [==============================] - ETA: 0s - loss: 1.0704 - accuracy: 0.5612
Epoch 2: val_loss did not improve from 1.31796
278/278 [==============================] - 1567s 6s/step - loss: 1.0704 - accuracy: 0.5612 - val_loss: 1.3548 - val_accuracy: 0.2396 - lr: 0.0010
Epoch 3/30
278/278 [==============================] - ETA: 0s - loss: 0.9788 - accuracy: 0.6237
Epoch 3: val_loss did not improve from 1.31796

Epoch 3: ReduceLROnPlateau reducing learning rate to 0.0003100000147242099.
278/278 [==============================] - 3591s 13s/step - loss: 0.9788 - accuracy: 0.6237 - val_loss: 1.5093 - val_accuracy: 0.2428 - lr: 0.0010
Epoch 4/30
278/278 [==============================] - ETA: 0s - loss: 0.7740 - accuracy: 0.6942
Epoch 4: val_loss did not improve from 1.31796
278/278 [==============================] - 1549s 6s/step - loss: 0.7740 - accuracy: 0.6942 - val_loss: 3.0068 - val_accuracy: 0.1838 - lr: 3.1000e-04
Epoch 5/30
278/278 [==============================] - ETA: 0s - loss: 0.6639 - accuracy: 0.7408
Epoch 5: val_loss improved from 1.31796 to 1.28359, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf5
278/278 [==============================] - 1554s 6s/step - loss: 0.6639 - accuracy: 0.7408 - val_loss: 1.2836 - val_accuracy: 0.3257 - lr: 3.1000e-04
Epoch 6/30
278/278 [==============================] - ETA: 0s - loss: 0.5945 - accuracy: 0.7732
Epoch 6: val_loss improved from 1.28359 to 1.24953, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf5
278/278 [==============================] - 1548s 6s/step - loss: 0.5945 - accuracy: 0.7732 - val_loss: 1.2495 - val_accuracy: 0.3829 - lr: 3.1000e-04
Epoch 7/30
278/278 [==============================] - ETA: 0s - loss: 0.5028 - accuracy: 0.8113
Epoch 7: val_loss did not improve from 1.24953
278/278 [==============================] - 1542s 6s/step - loss: 0.5028 - accuracy: 0.8113 - val_loss: 1.7992 - val_accuracy: 0.3604 - lr: 3.1000e-04
Epoch 8/30
278/278 [==============================] - ETA: 0s - loss: 0.4776 - accuracy: 0.8133
Epoch 8: val_loss did not improve from 1.24953

Epoch 8: ReduceLROnPlateau reducing learning rate to 9.61000009556301e-05.
278/278 [==============================] - 1542s 6s/step - loss: 0.4776 - accuracy: 0.8133 - val_loss: 1.8388 - val_accuracy: 0.3851 - lr: 3.1000e-04

Epoch 8: ReduceLROnPlateau reducing learning rate to 9.61000009556301e-05.
278/278 [==============================] - 1542s 6s/step - loss: 0.4776 - accuracy: 0.8133 - val_loss: 1.8388 - val_accuracy: 0.3851 - lr: 3.1000e-04
Epoch 9/30
278/278 [==============================] - ETA: 0s - loss: 0.3610 - accuracy: 0.8529
Epoch 9: val_loss improved from 1.24953 to 1.08672, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf5
278/278 [==============================] - 1544s 6s/step - loss: 0.3610 - accuracy: 0.8529 - val_loss: 1.0867 - val_accuracy: 0.6014 - lr: 9.6100e-05
Epoch 10/30
278/278 [==============================] - ETA: 0s - loss: 0.2888 - accuracy: 0.8859
Epoch 10: val_loss improved from 1.08672 to 0.70484, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf
5
278/278 [==============================] - 1539s 6s/step - loss: 0.2888 - accuracy: 0.8859 - val_loss: 0.7048 - val_accuracy: 0.7617 - lr: 9.6100e-05
Epoch 11/30
278/278 [==============================] - ETA: 0s - loss: 0.2547 - accuracy: 0.8973
Epoch 11: val_loss did not improve from 0.70484
278/278 [==============================] - 1544s 6s/step - loss: 0.2547 - accuracy: 0.8973 - val_loss: 2.4341 - val_accuracy: 0.4338 - lr: 9.6100e-05
Epoch 12/30
278/278 [==============================] - ETA: 0s - loss: 0.2316 - accuracy: 0.9068
Epoch 12: val_loss did not improve from 0.70484

Epoch 12: ReduceLROnPlateau reducing learning rate to 2.9791001288685948e-05.
278/278 [==============================] - 1536s 6s/step - loss: 0.2316 - accuracy: 0.9068 - val_loss: 2.4201 - val_accuracy: 0.3189 - lr: 9.6100e-05
Epoch 13/30
278/278 [==============================] - ETA: 0s - loss: 0.1943 - accuracy: 0.9191
Epoch 13: val_loss did not improve from 0.70484
278/278 [==============================] - 1538s 6s/step - loss: 0.1943 - accuracy: 0.9191 - val_loss: 1.2603 - val_accuracy: 0.6027 - lr: 2.9791e-05
Epoch 14/30
278/278 [==============================] - ETA: 0s - loss: 0.1748 - accuracy: 0.9318
Epoch 14: val_loss did not improve from 0.70484

Epoch 14: ReduceLROnPlateau reducing learning rate to 9.235210309270769e-06.
278/278 [==============================] - 1542s 6s/step - loss: 0.1748 - accuracy: 0.9318 - val_loss: 1.2872 - val_accuracy: 0.6239 - lr: 2.9791e-05
Epoch 15/30
278/278 [==============================] - ETA: 0s - loss: 0.1580 - accuracy: 0.9336
Epoch 15: val_loss improved from 0.70484 to 0.53354, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf
5
278/278 [==============================] - 1571s 6s/step - loss: 0.1580 - accuracy: 0.9336 - val_loss: 0.5335 - val_accuracy: 0.8527 - lr: 9.2352e-06
Epoch 16/30
278/278 [==============================] - ETA: 0s - loss: 0.1535 - accuracy: 0.9366
Epoch 16: val_loss did not improve from 0.53354
278/278 [==============================] - 1565s 6s/step - loss: 0.1535 - accuracy: 0.9366 - val_loss: 0.5525 - val_accuracy: 0.8518 - lr: 9.2352e-06
Epoch 17/30
```

```
Epoch 18/30
278/278 [==============================] - ETA: 0s - loss: 0.1508 - accuracy: 0.9403
Epoch 18: val_loss did not improve from 0.45056
278/278 [==============================] - 1561s 6s/step - loss: 0.1508 - accuracy: 0.9403 - val_loss: 0.4920 - val_accuracy: 0.8369 - lr: 9.2352e-06
Epoch 19/30
278/278 [==============================] - ETA: 0s - loss: 0.1491 - accuracy: 0.9420
Epoch 19: val_loss did not improve from 0.45056

Epoch 19: ReduceLROnPlateau reducing learning rate to 2.8629151620407354e-06.
278/278 [==============================] - 1561s 6s/step - loss: 0.1491 - accuracy: 0.9420 - val_loss: 0.5133 - val_accuracy: 0.8410 - lr: 9.2352e-06
Epoch 20/30
278/278 [==============================] - ETA: 0s - loss: 0.1385 - accuracy: 0.9410
Epoch 20: val_loss improved from 0.45056 to 0.44456, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf
5
278/278 [==============================] - 1546s 6s/step - loss: 0.1385 - accuracy: 0.9410 - val_loss: 0.4446 - val_accuracy: 0.8689 - lr: 2.8629e-06
Epoch 21/30
278/278 [==============================] - ETA: 0s - loss: 0.1347 - accuracy: 0.9430
Epoch 21: val_loss improved from 0.44456 to 0.43863, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf
5
278/278 [==============================] - 1535s 6s/step - loss: 0.1347 - accuracy: 0.9430 - val_loss: 0.4386 - val_accuracy: 0.8698 - lr: 2.8629e-06
Epoch 22/30
278/278 [==============================] - ETA: 0s - loss: 0.1364 - accuracy: 0.9448
Epoch 22: val_loss did not improve from 0.43863
278/278 [==============================] - 1540s 6s/step - loss: 0.1364 - accuracy: 0.9448 - val_loss: 0.4415 - val_accuracy: 0.8707 - lr: 2.8629e-06
Epoch 23/30
278/278 [==============================] - ETA: 0s - loss: 0.1351 - accuracy: 0.9455
Epoch 23: val_loss improved from 0.43863 to 0.43855, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf
5

Epoch 23: ReduceLROnPlateau reducing learning rate to 8.875036974131945e-07.
278/278 [==============================] - 1542s 6s/step - loss: 0.1351 - accuracy: 0.9455 - val_loss: 0.4385 - val_accuracy: 0.8698 - lr: 2.8629e-06
Epoch 24/30
278/278 [==============================] - ETA: 0s - loss: 0.1307 - accuracy: 0.9481
Epoch 24: val_loss did not improve from 0.43855
278/278 [==============================] - 1536s 6s/step - loss: 0.1307 - accuracy: 0.9481 - val_loss: 0.4419 - val_accuracy: 0.8707 - lr: 8.8750e-07
Epoch 25/30
278/278 [==============================] - ETA: 0s - loss: 0.1345 - accuracy: 0.9474
Epoch 25: val_loss improved from 0.43855 to 0.43725, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf
5
278/278 [==============================] - 1537s 6s/step - loss: 0.1345 - accuracy: 0.9474 - val_loss: 0.4373 - val_accuracy: 0.8680 - lr: 8.8750e-07
Epoch 26/30
Epoch 26: val_loss improved from 0.43725 to 0.43690, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf
5
278/278 [==============================] - 1534s 6s/step - loss: 0.1315 - accuracy: 0.9458 - val_loss: 0.4369 - val_accuracy: 0.8712 - lr: 8.8750e-07
Epoch 27/30
278/278 [==============================] - ETA: 0s - loss: 0.1270 - accuracy: 0.9496
Epoch 27: val_loss improved from 0.43690 to 0.43477, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodelepoch30.hdf
5
278/278 [==============================] - 1538s 6s/step - loss: 0.1270 - accuracy: 0.9496 - val_loss: 0.4348 - val_accuracy: 0.8680 - lr: 8.8750e-07
Epoch 28/30
278/278 [==============================] - ETA: 0s - loss: 0.1353 - accuracy: 0.9459
Epoch 28: val_loss did not improve from 0.43477
278/278 [==============================] - 1539s 6s/step - loss: 0.1353 - accuracy: 0.9459 - val_loss: 0.4385 - val_accuracy: 0.8703 - lr: 8.8750e-07
Epoch 29/30
278/278 [==============================] - ETA: 0s - loss: 0.1241 - accuracy: 0.9484
Epoch 29: val_loss did not improve from 0.43477

Epoch 29: ReduceLROnPlateau reducing learning rate to 2.751261433786567e-07.
278/278 [==============================] - 1539s 6s/step - loss: 0.1241 - accuracy: 0.9484 - val_loss: 0.4378 - val_accuracy: 0.8694 - lr: 8.8750e-07
Epoch 30/30
278/278 [==============================] - ETA: 0s - loss: 0.1260 - accuracy: 0.9485
Epoch 30: val_loss did not improve from 0.43477
278/278 [==============================] - 1539s 6s/step - loss: 0.1260 - accuracy: 0.9485 - val_loss: 0.4350 - val_accuracy: 0.8698 - lr: 2.7513e-07
```

Figure 4.9 Output Snippet of Model Training and Epochs at Dropout 0.1

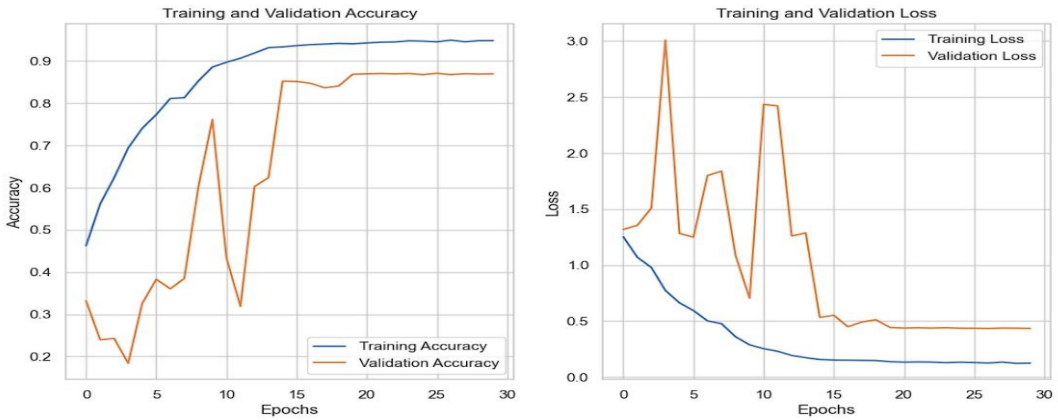


Figure 4.10 Accuracy and Loss Graph of Training and Validation at Dropout 0.1

Training Accuracy: 95.49%

Testing Accuracy: 85.90%

Training Loss: 0.1334

Testing Loss: 0.4243

As seen in the above figure, the training accuracy increased from around 0.48 to 0.94 and the validation accuracy increased from 0.33 to 0.86. The high value of training accuracy indicates that our model is trained well and is distinguishing the skin cancer images with great accuracy. Also, the high value of validation accuracy indicates that our model is not only performing well on seen data; but also on unseen data, the model is able to classify the images. Observing the nature of the curve, we can conclude that our model is neither over-fitted as there is no sign that training loss is extremely high, but the validation accuracy is extremely low, nor under-fitted as there is no sign of both the training and validation accuracies being low.

Also, as seen in the above figure, the training loss decreased from 1.3 to 0.12 and validation loss from 1.3 to 0.43. Both the values are quite low, and they indicate that there is less difference in actual and predicted values of both the training and validation dataset. Thus, the optimal value of dropout was selected through experimentation.

Batch Size: 64

```
138/138 [==============================] - ETA: 0s - loss: 1.3163 - accuracy: 0.4618
Epoch 1: val_loss improved from inf to 1.29975, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodel64.hdf5
C:\Users\Asus\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\training.py:3079: UserWarning: You are saving your model as an HDF5 file v
  saving_api.save_model(
138/138 [==============================] - 1681s 6s/step - loss: 1.3163 - accuracy: 0.4618 - val_loss: 1.2997 - val_accuracy: 0.4158 - lr: 0.0010
Epoch 2/30
138/138 [==============================] - ETA: 0s - loss: 1.0893 - accuracy: 0.5397
Epoch 2: val_loss improved from 1.29975 to 1.26392, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodel64.hdf5
138/138 [==============================] - 1567s 6s/step - loss: 1.0893 - accuracy: 0.5397 - val_loss: 1.2639 - val_accuracy: 0.4824 - lr: 0.0010
Epoch 3/30
138/138 [==============================] - ETA: 0s - loss: 1.0016 - accuracy: 0.5968
Epoch 3: val_loss did not improve from 1.26392
138/138 [==============================] - 1555s 6s/step - loss: 1.0016 - accuracy: 0.5968 - val_loss: 1.7482 - val_accuracy: 0.2419 - lr: 0.0010
Epoch 4/30
138/138 [==============================] - ETA: 0s - loss: 0.9439 - accuracy: 0.6262
Epoch 4: val_loss improved from 1.26392 to 1.20423, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodel64.hdf5
138/138 [==============================] - 1559s 6s/step - loss: 0.9439 - accuracy: 0.6262 - val_loss: 1.2042 - val_accuracy: 0.4671 - lr: 0.0010
Epoch 5/30
138/138 [==============================] - ETA: 0s - loss: 0.8407 - accuracy: 0.6822
Epoch 5: val_loss did not improve from 1.20423
138/138 [==============================] - 1553s 6s/step - loss: 0.8407 - accuracy: 0.6822 - val_loss: 1.2132 - val_accuracy: 0.4779 - lr: 0.0010
Epoch 6/30
138/138 [==============================] - ETA: 0s - loss: 0.7990 - accuracy: 0.6894
Epoch 6: val_loss did not improve from 1.20423

Epoch 6: ReduceLROnPlateau reducing learning rate to 0.0003100000147242099.
138/138 [==============================] - 1553s 6s/step - loss: 0.7990 - accuracy: 0.6894 - val_loss: 1.3511 - val_accuracy: 0.3063 - lr: 0.0010
Epoch 7/30
138/138 [==============================] - ETA: 0s - loss: 0.5911 - accuracy: 0.7706
Epoch 7: val_loss improved from 1.20423 to 1.18525, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodel64.hdf5
138/138 [==============================] - 1555s 6s/step - loss: 0.5911 - accuracy: 0.7706 - val_loss: 1.1853 - val_accuracy: 0.4698 - lr: 3.1000e-04
Epoch 8/30
138/138 [==============================] - ETA: 0s - loss: 0.4840 - accuracy: 0.8174
Epoch 8: val_loss did not improve from 1.18525
138/138 [==============================] - 1556s 6s/step - loss: 0.4840 - accuracy: 0.8174 - val_loss: 1.5285 - val_accuracy: 0.2991 - lr: 3.1000e-04
```

```
Epoch 9: val_loss did not improve from 1.18525

Epoch 9: ReduceLROnPlateau reducing learning rate to 9.61000009556301e-05.
138/138 [==============================] - 1554s 6s/step - loss: 0.4358 - accuracy: 0.8320 - val_loss: 1.6519 - val_accuracy: 0.3023 - lr: 3.1000e-04
Epoch 10/30
138/138 [==============================] - ETA: 0s - loss: 0.3624 - accuracy: 0.8574
Epoch 10: val_loss did not improve from 1.18525
138/138 [==============================] - 1551s 6s/step - loss: 0.3624 - accuracy: 0.8574 - val_loss: 1.2843 - val_accuracy: 0.5977 - lr: 9.6100e-05
Epoch 11/30
138/138 [==============================] - ETA: 0s - loss: 0.2960 - accuracy: 0.8753
Epoch 11: val_loss improved from 1.18525 to 0.72282, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodel64.hdf5
138/138 [==============================] - 1555s 6s/step - loss: 0.2960 - accuracy: 0.8753 - val_loss: 0.7228 - val_accuracy: 0.7500 - lr: 9.6100e-05
Epoch 12/30
138/138 [==============================] - ETA: 0s - loss: 0.2826 - accuracy: 0.8884
Epoch 12: val_loss did not improve from 0.72282
138/138 [==============================] - 1551s 6s/step - loss: 0.2826 - accuracy: 0.8884 - val_loss: 3.3242 - val_accuracy: 0.4505 - lr: 9.6100e-05
Epoch 13/30
138/138 [==============================] - ETA: 0s - loss: 0.2510 - accuracy: 0.9010
Epoch 13: val_loss did not improve from 0.72282

Epoch 13: ReduceLROnPlateau reducing learning rate to 2.9791001288685948e-05.
138/138 [==============================] - 1552s 6s/step - loss: 0.2510 - accuracy: 0.9010 - val_loss: 1.8661 - val_accuracy: 0.5306 - lr: 9.6100e-05
Epoch 14/30
138/138 [==============================] - ETA: 0s - loss: 0.2076 - accuracy: 0.9131
Epoch 14: val_loss improved from 0.72282 to 0.59214, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodel64.hdf5
138/138 [==============================] - 1552s 6s/step - loss: 0.2076 - accuracy: 0.9131 - val_loss: 0.5921 - val_accuracy: 0.7910 - lr: 2.9791e-05
Epoch 15/30
138/138 [==============================] - ETA: 0s - loss: 0.1921 - accuracy: 0.9229
Epoch 15: val_loss did not improve from 0.59214
138/138 [==============================] - 1549s 6s/step - loss: 0.1921 - accuracy: 0.9229 - val_loss: 0.6378 - val_accuracy: 0.7964 - lr: 2.9791e-05
Epoch 16/30
138/138 [==============================] - ETA: 0s - loss: 0.1840 - accuracy: 0.9244
Epoch 16: val_loss did not improve from 0.59214

Epoch 16: ReduceLROnPlateau reducing learning rate to 9.235210309270769e-06.
138/138 [==============================] - 1552s 6s/step - loss: 0.1840 - accuracy: 0.9244 - val_loss: 0.5943 - val_accuracy: 0.8000 - lr: 2.9791e-05
Epoch 17/30
138/138 [==============================] - ETA: 0s - loss: 0.1792 - accuracy: 0.9253
Epoch 17: val_loss did not improve from 0.59214
138/138 [==============================] - 1552s 6s/step - loss: 0.1792 - accuracy: 0.9253 - val_loss: 0.6853 - val_accuracy: 0.8072 - lr: 9.2352e-06
```

```
Epoch 18: val_loss improved from 0.59214 to 0.49620, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodel64.hdf5
138/138 [==============================] - 1550s 6s/step - loss: 0.1685 - accuracy: 0.9299 - val_loss: 0.4962 - val_accuracy: 0.8302 - lr: 9.2352e-06
Epoch 19/30
138/138 [==============================] - ETA: 0s - loss: 0.1640 - accuracy: 0.9322
Epoch 19: val_loss improved from 0.49620 to 0.48012, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodel64.hdf5
138/138 [==============================] - 1551s 6s/step - loss: 0.1640 - accuracy: 0.9322 - val_loss: 0.4801 - val_accuracy: 0.8383 - lr: 9.2352e-06
Epoch 20/30
138/138 [==============================] - ETA: 0s - loss: 0.1545 - accuracy: 0.9348
Epoch 20: val_loss did not improve from 0.48012
138/138 [==============================] - 1554s 6s/step - loss: 0.1545 - accuracy: 0.9348 - val_loss: 0.5253 - val_accuracy: 0.8356 - lr: 9.2352e-06
Epoch 21/30
138/138 [==============================] - ETA: 0s - loss: 0.1664 - accuracy: 0.9294
Epoch 21: val_loss improved from 0.48012 to 0.46580, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodel64.hdf5
138/138 [==============================] - 1551s 6s/step - loss: 0.1664 - accuracy: 0.9294 - val_loss: 0.4658 - val_accuracy: 0.8541 - lr: 9.2352e-06
Epoch 22/30
138/138 [==============================] - ETA: 0s - loss: 0.1578 - accuracy: 0.9316
Epoch 22: val_loss did not improve from 0.46580
138/138 [==============================] - 1545s 6s/step - loss: 0.1578 - accuracy: 0.9316 - val_loss: 0.5163 - val_accuracy: 0.8450 - lr: 9.2352e-06
Epoch 23/30
138/138 [==============================] - ETA: 0s - loss: 0.1530 - accuracy: 0.9356
Epoch 23: val_loss did not improve from 0.46580

Epoch 23: ReduceLROnPlateau reducing learning rate to 2.8629151620407354e-06.
138/138 [==============================] - 1548s 6s/step - loss: 0.1530 - accuracy: 0.9356 - val_loss: 0.5234 - val_accuracy: 0.8338 - lr: 9.2352e-06
Epoch 24/30
138/138 [==============================] - ETA: 0s - loss: 0.1489 - accuracy: 0.9376
Epoch 24: val_loss did not improve from 0.46580
138/138 [==============================] - 1552s 6s/step - loss: 0.1489 - accuracy: 0.9376 - val_loss: 0.4896 - val_accuracy: 0.8505 - lr: 2.8629e-06
Epoch 25/30
138/138 [==============================] - ETA: 0s - loss: 0.1542 - accuracy: 0.9364
Epoch 25: val_loss did not improve from 0.46580

Epoch 25: ReduceLROnPlateau reducing learning rate to 8.875036974131945e-07.
138/138 [==============================] - 1551s 6s/step - loss: 0.1542 - accuracy: 0.9364 - val_loss: 0.4974 - val_accuracy: 0.8509 - lr: 2.8629e-06
Epoch 26/30
138/138 [==============================] - ETA: 0s - loss: 0.1534 - accuracy: 0.9389
Epoch 26: val_loss improved from 0.46580 to 0.44352, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodel64.hdf5
138/138 [==============================] - 1551s 6s/step - loss: 0.1534 - accuracy: 0.9389 - val_loss: 0.44352 - val_accuracy: 0.8564 - lr: 8.8750e-07
Epoch 27/30
```

```
138/138 [==============================] - ETA: 0s - loss: 0.1487 - accuracy: 0.9403
Epoch 27: val_loss improved from 0.44352 to 0.4123, saving model to C:\Users\Asus\Desktop\archive\ISIC_2019_Training_Input\Training\skinmodel64.hdf5
138/138 [==============================] - 1551s 6s/step - loss: 0.1487 - accuracy: 0.9403- val_loss: 0.4123 - val_accuracy: 0.8578 - lr: 8.8750e-07
Epoch 28/30
138/138 [==============================] - ETA: 0s - loss: 0.1494 - accuracy: 0.9392
Epoch 28: val_loss did not improve from 0.4123
138/138 [==============================] - 1552s 6s/step - loss: 0.1494 - accuracy: 0.9392 - val_loss: 0.4234 - val_accuracy: 0.8505 - lr: 8.8750e-07
Epoch 29/30
138/138 [==============================] - ETA: 0s - loss: 0.1467 - accuracy: 0.9426
Epoch 29: val_loss did not improve from 0.4123

Epoch 29: ReduceLROnPlateau reducing learning rate to 2.751261433786567e-07.
138/138 [==============================] - 1551s 6s/step - loss: 0.1467 - accuracy: 0.9426 - val_loss: 0.4268 - val_accuracy: 0.8556 - lr: 8.8750e-07
Epoch 30/30
138/138 [==============================] - ETA: 0s - loss: 0.1475 - accuracy: 0.9459
Epoch 30: val_loss did not improve from 0.4123
138/138 [==============================] - 1552s 6s/step - loss: 0.1475 - accuracy: 0.9459 - val_loss: 0.4198 - val_accuracy: 0.8558 - lr: 2.7512e-07
```

Figure 4.11 Output Snippet of Model Training & Epochs at BS 64 & Dropout 0.1

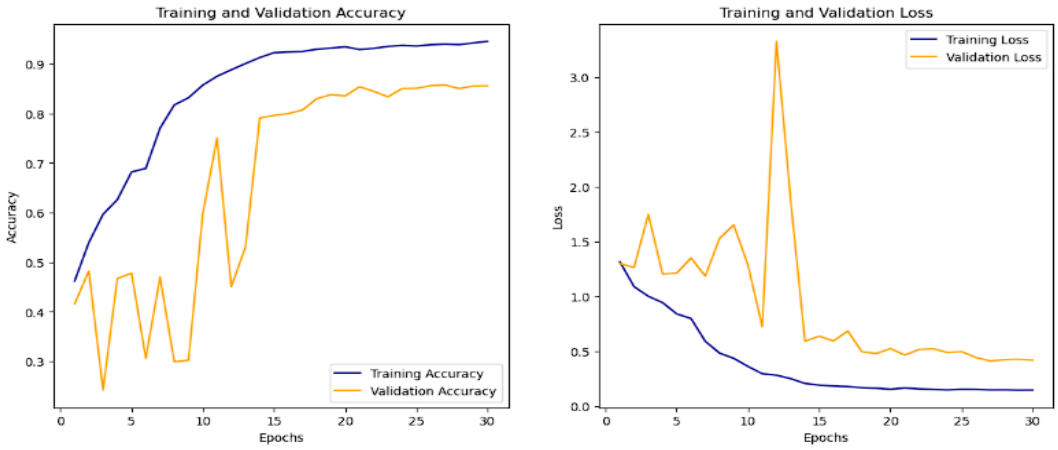


Figure 4.12 Graph of Training &Validation at Batch Size 64 & Dropout 0.1

Training Accuracy: 94.59%

Testing Accuracy: 85.58%

Training Loss: 0.1475

Testing Loss: 0.4198

Keeping the dropout rate as 0.1, which was found to be optimal, we changed the batch size from 32 to 64. Previously, the batch size was set to 32 with the total number of training data in each epoch being (0.8 * 11097) =8877.6 ceiled to 8878, the number of steps per epoch was $\frac{8878}{32}$ =278 (ceiled). When the batch size is set to 64, the number of steps per epoch becomes $\frac{8878}{64}$ =138 (ceiled). When keeping the dropout rate the same, and increasing batch size from 32 to 64, it was observed that the values of training and validation accuracy and loss were almost the same, but it led to shorter training times per epoch. Larger batch sizes typically result in fewer batches per epoch, reducing the overhead associated with loading and preprocessing data. Thus, for the optimal accuracy and loss values, several experimentations were concluded with different values of batch size, number of epochs, dropout rate and then, finally through trial-and-error process, the best values of hyperparameter were thus chosen.

4.2 CONCLUSION

Our system showed satisfactory accuracy for both training and testing purposes, and we developed it using EfficientnetB4 architecture because it is lightweight as well as has higher accuracy than other architectures for the same number of parameters. Then, the model was integrated with the web application using Flask and the inputs were given and the prediction was displayed in the web application. In conclusion, our system can predict most of the input data correctly, while sometimes showing failure to produce the correct result due to presence of some similarity in the images of different classes.

4.3 FUTURE ENHANCEMENT

The developed system can be made even better in the future by implementing the following:

- The system can be extended to classify more classes of skin cancer.
- The web application can be extended to mobile application.

REFERENCES

[1]. World Cancer Research Fund, “Skin Cancer Statistics,” [Online]. Available: https://www.wcrf.org/cancer-trends/skin-cancer-statistics/. [Accessed Jul 2023].

[2]. K. Ali, Z. A. Shaikh, A. A. Khan, A. A. Laghari, “Multiclass skin cancer classification using EfficientNets – a first step towards preventing skin cancer,” Neuroscience Informatics, vol. 2, no. 4, pp. 100034, 2022, ISSN 2772-5286. [Online]. Available: https://doi.org/10.1016/j.neuri.2021.100034. [Accessed Jul 2023].

[3]. H. A. Haenssle, C. Fink, R. Schneiderbauer, F. Toberer, T. Buhl, A. Blum, A. Kalloo, A. Ben Hadj Hassen, L. Thomas, A. Enk, and L. Uhlmann, "Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists," Annals of Oncology, vol. 29, no. 8, pp. 1836-1842, Aug. 2018. [Online]. Available: https://doi.org/10.1093/annonc/mdy166. [Accessed Dec 2023].

[4]. M. Tahir, A. Naeem, H. Malik H, J. Tanveer, R. A. Naqvi, S. -W. Lee, “DSCC_Net: Multi-Classification Deep Learning Models for Diagnosing of Skin Cancer Using Dermoscopic Images,” Cancers, vol. 15, no. 7, p.2179, 2023. [Online]. Available: https://doi.org/10.3390/cancers15072179. [Accessed Dec 2023].

[5]. S. Likhitha and R. Baskar, "Skin Cancer Segmentation Using R-CNN Comparing with Inception V3 for Better Accuracy," in 2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART), Moradabad, India,2022, pp. 1-6.
[Online]. Available: https://ieeexplore.ieee.org/document/10047686. [Accessed Jul 2023].

[6]. S. Likhitha and R. Baskar, "Skin Cancer Classification using CNN in Comparison with Support Vector Machine for Better Accuracy," in 2022 11th International

Conference on System Modeling & Advancement in Research Trends (SMART), Moradabad, India, 2022, pp. 1-6.
[Online]. Available: https://ieeexplore.ieee.org/document/10047280. [Accessed Jul 2023].

[7]. Z. Li, "A Skin Cancer Detection System Based on CNN with Hair Removal," in 2023 IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA), Shenyang, China, 2023, pp. 1-5. [Online]. Available: https://ieeexplore.ieee.org/document/10076164. [Accessed 2023].

[8]. N. Razmjooy and A. Arshaghi, "Application of Multilevel Thresholding and CNN for the Diagnosis of Skin Cancer Utilizing a Multi-Agent Fuzzy Buzzard Algorithm," Biomedical Signal Processing and Control, vol. 84, 2023. [Online]. Available:https://www.sciencedirect.com/science/article/abs/pii/S1746809423004172. [Accessed 2023].

[9]. "Block diagram of the evaluated architecture for classification of skin lesions using convolutional neural networks," ResearchGate, [Online]. Available: https://www.researchgate.net/figure/Block-diagram-of-the-evaluated-architecture-for-classification-of-skin-lesions-using_fig1_342887043/. [Accessed Jul 2023].

[10]. "Convolution Neural Network (Deep Learning)," DevelopersBreach, [Online]. Available:https://developersbreach.com/convolution-neural-network-deep-learning/. [Accessed Jul 2023].

[11]. "Deep Learning and Its Applications in Biomedicine," ResearchGate, [Online]. Available: https://www.researchgate.net/publication/323612444_Deep_Learning_and_It_Applications_in_Biomedicine/. [Accessed Jul 2023].

BIBLIOGRAPHY

- M. Dildar, S. Akram, M. Irfan, H. U. Khan, M. Ramzan, A. R. Mahmood, S. A. Alsaiari, A. H. M. Saeed, M. O. Alraddadi, and M. H. Mahnashi, "Skin Cancer Detection: A Review Using Deep Learning Techniques," International Journal of Environmental Research and Public Health, vol. 18, no. 10, p. 5479, May 2021. [Online]. Available: 10.3390/ijerph18105479. [Accessed Aug 2023].

- "Skin Cancer Detection: A Review Using Deep Learning Techniques," International Journal of Environmental Research and Public Health, vol. 18, no. 10,p.5479,May2021.[Online].Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8160886/. [Accessed Sep 2023].

CHAPTER 5: APPENDIX

Few screenshots of code are:

```
source = 'C:/Users/Asus/Desktop/archive/ISIC_2019_Training_Input/ISIC_2019_Training_Input/'
destination = 'C:/Users/Asus/Desktop/archive/ISIC_2019_Training_Input/BasalCellCarcinoma/'

allfiles = os.listdir(source)

for f in allfiles:
    m=f.split('.')[0]
    if m in basalcell_list:
        file="basalcell_" + str(f)
        print(file)
        shutil.copy(source + f, destination + file)
```

Figure: Code Snippet for Data Preparation

```
# Path to the CSV files
csv_file_path_ground = "C:\\Users\\Asus\\Desktop\\archive\\ISIC_2019_Training_Input\\ISIC_2019_Training_GroundTruth.csv"
csv_file_path_metadata = "C:\\Users\\Asus\\Desktop\\archive\\ISIC_2019_Training_Input\\ISIC_2019_Training_Metadata.csv"

# Read the CSV files
ground_truth = pd.read_csv(csv_file_path_ground)
metadata = pd.read_csv(csv_file_path_metadata)

# Merge the DataFrames based on the "image" column
combined_data = pd.merge(ground_truth, metadata, on="image")
```

Figure: Code Snippet for Data Frame Preparation

```
# Load the pre-trained EfficientNetB4 model without the top (classification) layers
base_model = EfficientNetB4(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Set the number of layers to fine-tune
fine_tune_layers = 500

# Freeze the first few layers and unfreeze the rest
for layer in base_model.layers[:-fine_tune_layers]:
    layer.trainable = False
for layer in base_model.layers[-fine_tune_layers:]:
    layer.trainable = True
```

Figure: Code Snippet for Building Model

```
# Add fully connected layers and dropout
x = Dense(150, activation='relu')(x)
x = Dropout(0.12)(x)
x = Dense(50, activation='relu')(x)
x = Dropout(0.1)(x)
# Output layer for classification (adjust the number of units for your task)
predictions = Dense(4, activation='softmax', name='output')(x)

# Create the final model
model = Model(inputs=[image_input, age_input, sex_input, anatomical_sites_input], outputs=predictions)
```

Figure: Code snippet for adding layers to the model

```python
from tensorflow.keras.optimizers import Adam
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(
    train_data_generator,
    epochs=30,
    validation_data=val_data_generator,
    class_weight=class_wt_dict,
    callbacks=callback_list
)
```

Figure: Code Snippet for Compiling Model

Few screenshots of output are:

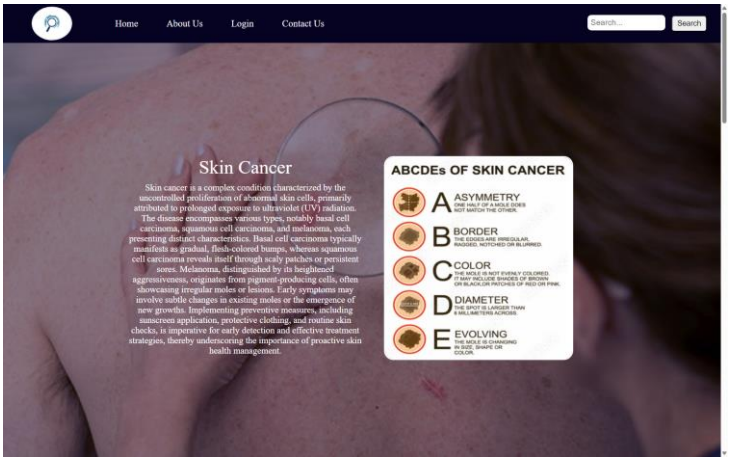
```
Features shape: (11097, 224, 224, 3)
[[[[122 111 109]
   [125 109 106]
   [ 92  69  67]
   ...
   [134 133 131]
   [131 130 128]
   [128 127 125]]

  [[118 110 107]
   [126 115 111]
   [109  91  87]
   ...
   [135 135 132]
   [132 131 129]
   [129 128 126]]

  [[119 109 107]
   [114 103  99]
   [123 112 106]
   ...
   [135 134 132]
   [132 131 128]
   [131 130 128]]
...
['female' 'male' 'female' ... 'male' 'male' 'female']
Anatomical Sites shape: (11097,)
['head/neck' 'unknown' 'unknown' ... 'anterior torso' 'lower extremity'
 'upper extremity']
```

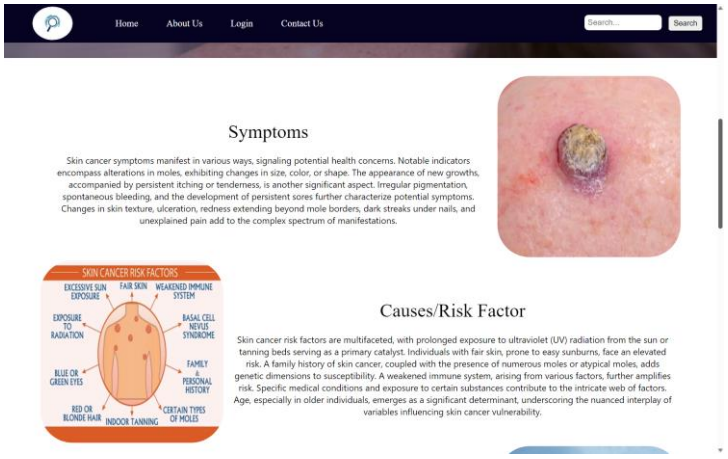
Figure: Conversion of Skin Images to NumPy Array

```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 image_input (InputLayer)    [(None, 224, 224, 3)]     0         []

 efficientnetb4 (Functional  (None, 7, 7, 1792)        17673823  ['image_input[0][0]']
 )                                                     3

 global_max_pooling2d_1 (Gl  (None, 1792)              0         ['efficientnetb4[1][0]']
 obalMaxPooling2D)

 global_average_pooling2d_1  (None, 1792)              0         ['efficientnetb4[1][0]']
  (GlobalAveragePooling2D)

 age_input (InputLayer)      [(None, 1)]               0         []

 sex_input (InputLayer)      [(None, 3)]               0         []

 anatomical_sites_input (In  [(None, 9)]               0         []
 putLayer)

 concatenate_2 (Concatenate  (None, 3584)              0         ['global_max_pooling2d_1[0][0]
 )                                                                ',
                                                                   'global_average_pooling2d_1[0
...
Total params: 18221277 (69.51 MB)
Trainable params: 18096070 (69.03 MB)
Non-trainable params: 125207 (489.09 KB)
__________________________________________________________________________________________________
```

Figure: Summary of the Developed Model

Few screenshots of frontend design are:





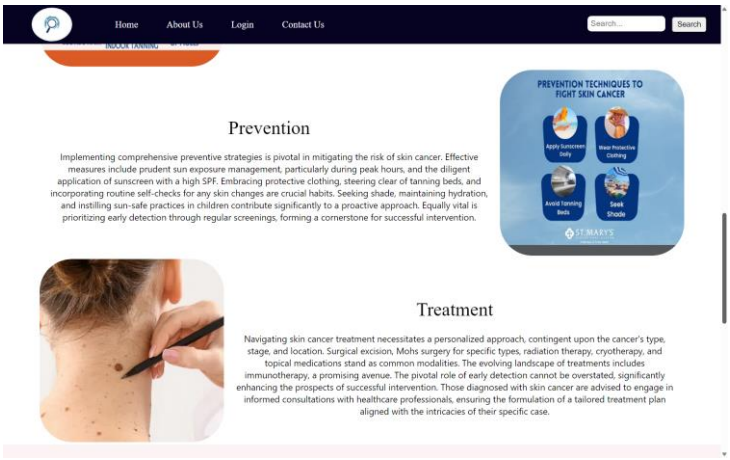
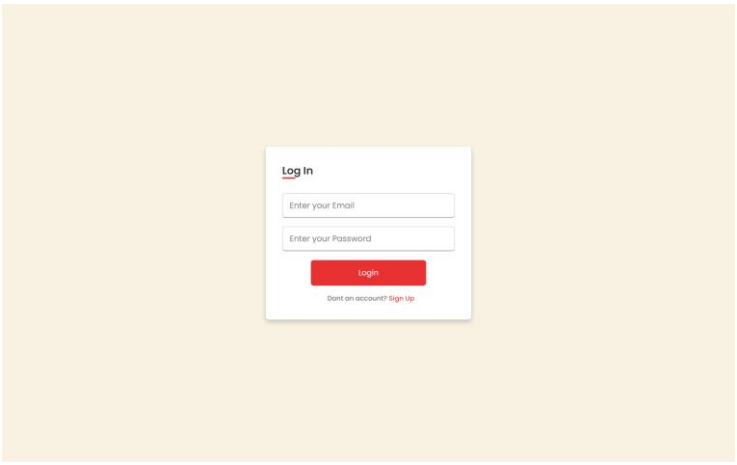
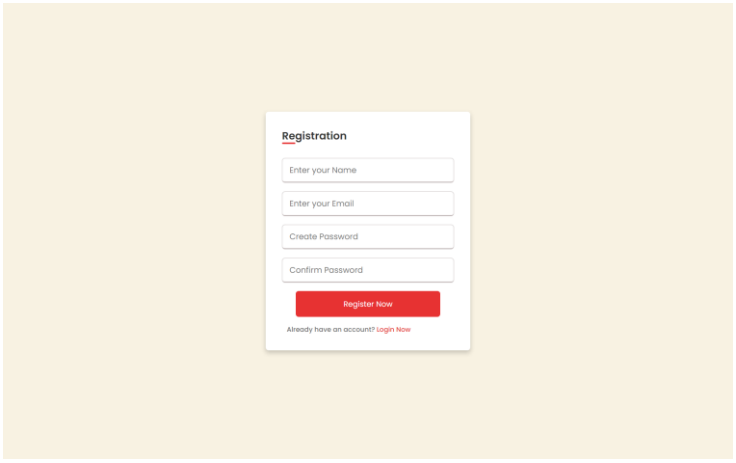


Figure: Home Page





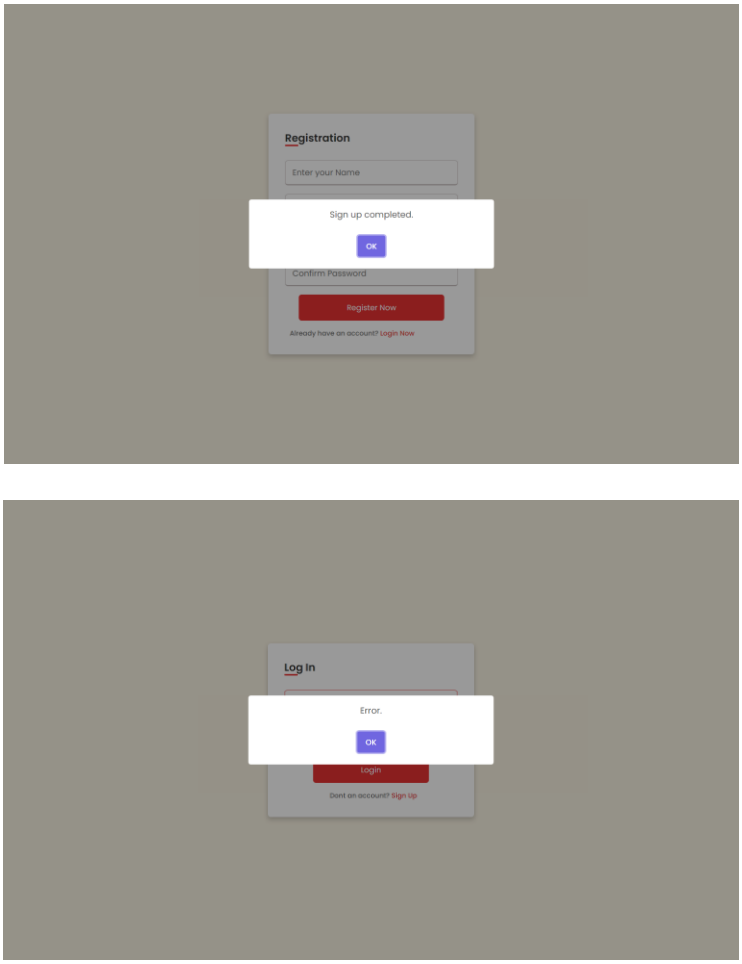
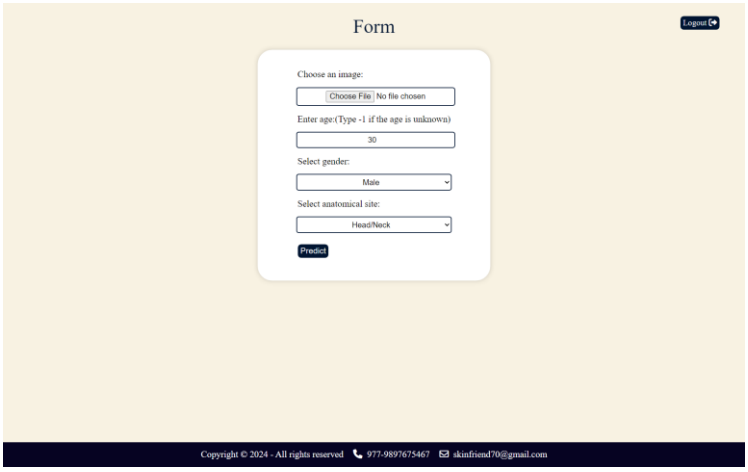


Figure: Login and Registration



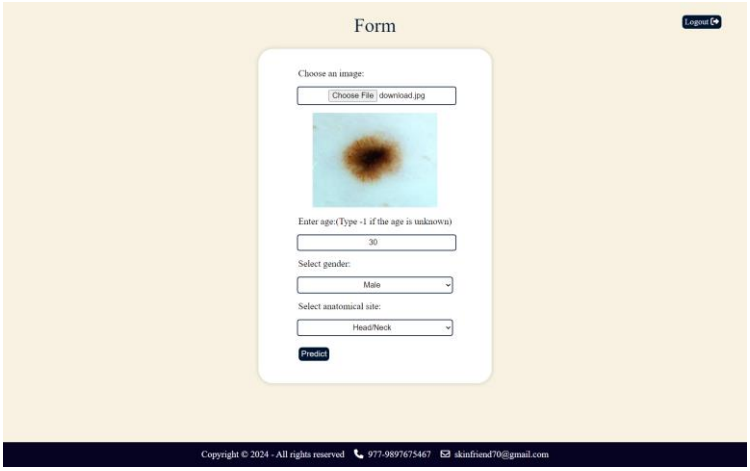


Figure: User Interface to Provide Input Data

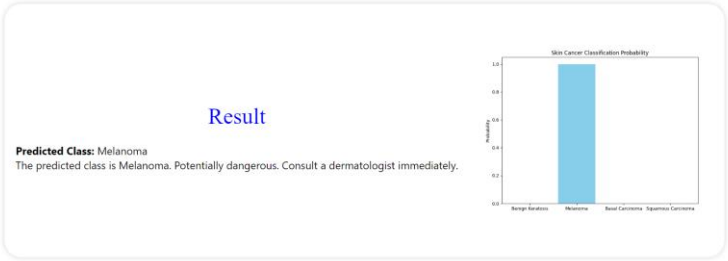


Figure: Result Page- Correctly Predicted as Melanoma