A

MINI PROJECT REPORT

ON

Skin Cancer Detection

Submitted in partial fulfillment of the requirements

For the award of Degree of

BACHELOR OF ENGINEERING

IN

CSE (AI-ML)

Submitted By

Anumalla Karthikeyan **245321748006**

Under the guidance

Of

Mr. P Nageswara Rao

ASSISTANT PROFESSOR

Department of CSE(AIML)

NEIL GOGTE INSTITUTE OF TECHNOLOGY

KachivaniSingaram Village, Hyderabad, Telangana 500058.

JANUARY 2023

NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)

Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

<u>CERTIFICATE</u>

This is to certify that the Mini project work entitled "**Skin Cancer Detection"** *is a bonafide work carried out by* **Anumalla Karthikeyan(245321748006),** of III-year V semester **Bachelor of Engineering** *in CSE(***AIML)** *by Osmania University, Hyderabad during the academic year* **2023-2024** *is a record of bonafide work carried out by them. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree .*

Internal Guide	**Head of Department**
Mr.P Nageswara Rao	Dr. T.Prem Chander
Assistant Professor	Associate Professor

NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)

Approved by AICTE, New Delhi & Affiliated to Osmania University Hyderabad

DECLARATION

I hereby declare that the Mini Project Report entitled, "**Skin Cancer Detection**" submitted for the B.E degree is entirely my work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

Date:

Anumalla Karthikeyan **245321748006**

ACKNOWLEDGEMENT

I am happy to express our deep sense of gratitude to the principal of the college **Dr. R. Shyam Sunder, Professor**, Neil Gogte Institute of Technology, for having provided us with adequate facilities to pursue our project.

I would like to thank **Dr. T.Prem Chander**, **Head of the Department,** CSE(AIML), Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

I would also like to thank my internal guide **Mr. Nageswara Rao, Assistant Professor** for his technical guidance & constant encouragement.

I sincerely thank my seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering for their timely suggestions, healthy criticism and motivation during this work.

Finally, I express my immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

ABSTRACT

Pattern recognition (PR) is realized as a human recognition process which can be completed by computer technology. I should first enter useful information about identifying the object into the computer. For this reason, I must abstract the recognition object and establish its mathematical model to describe it and replace the recognition object for what the machine can process. The description of this object is the pattern. Simply speaking, the pattern recognition is to identify the category to which the object belongs, such as the face in face recognition. My application is based on PR which is to identify the types of Cancer.

TABLE OF CONTENTS

CHAPTER – 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Skin cancer is one of the most common types of cancer globally, with millions of new cases reported each year. Early detection plays a crucial role in improving patient outcomes and reducing mortality rates. However, accurate diagnosis of skin lesions requires specialized knowledge and expertise, often leading to delays in diagnosis and treatment. The objective of this project is to develop comprehensive documentation for a skin cancer detection system that utilizes artificial intelligence (AI) and machine learning techniques. The documentation will serve as a guide for healthcare professionals, researchers, and developers interested in understanding and implementing automated skin cancer detection solutions.

1.2 MOTIVATION

Skin cancer is a serious health concern, and early detection significantly improves treatment outcomes. Melanoma, a type of skin cancer, has a high treatment cost and mortality rate. Detecting skin cancer manually is time-consuming and can delay timely intervention. The proposed project aims to create an automated carcinoma detection system using image processing and machine learning techniques. By analyzing skin images, the system extracts features from affected skin cells after segmentation. This approach accelerates the diagnosis process and facilitates timely treatment. There's growing interest in computer-assisted technology for skin cancer detection using dermatoscopic images.

1.3 SCOPE

This application deals with the problem of developing a Skin Cancer identification system with the use of CNN (Convolutional Neural Networks) to classify different skin cancers that can be made on a general usage such as (actinic keratosis, basal cell carcinoma, melanoma , etc..). The proposed system consists of mainly three phases: the first phase (i.e., pre-processing), the next phase (i.e., model training) and the final phase (i.e., classification). The first phase includes the "ImageDataGenerator" module from the "Keras" package which is used to resize the image and segregate the images into training and testing datasets. In the next phase, I train the model on "training" and "validation" datasets using 10 "epoch" cycles and 2208 "steps_per_epoch" as well as 71 "validation_steps". The next phase, which constitutes the main part of this project, is devoted to the classification problem where the model is used to classify different cancer types.

1.4 OUTLINE

The Fully Handcrafted CNN model classifies the dataset into three parts, i.e., training dataset, testing dataset and lastly, validation dataset. The model learns from the training dataset and then using the validation dataset, inputs are validated. I used "Streamlit" to create web application that is flexible to use 200MB sized images as its input.

CHAPTER – 2

LITERATURE SURVEY

EXISTING SYSTEM:

Machine learning (ML) represents a set of techniques that allow systems to discover the required representations to feature detection or classification from the raw data. The performance of works in the classification system depends on the quality of the features. As such, this study can be categorized under the field of ML. This is to make a search in this area for the studies that belong to cancer' identification. They used the ratio of the distance from the affected area to normal skin, and the diseased width. This feature was integrated in the decision tree, and then in SVM. This proposal was applied to the database called (SKIN CANCER ISIC dataset) that mentioned in the results achieved that the correct classification rate is 84%.

PROPOSED SYSTEM:

The Convolutional Neural Network (CNN) is a deep learning algorithm which includes an input image and assigns the weights and the distinctions to the various aspects of the images and can then distinguish one image from another. The preprocessing required in CNN compared with other classification algorithms is much lower. In primitive methods, filters are usually hand-engineered; on the other hand, CNN could learn these filters on its own when subjected to enough training. CNN's architecture is quite like that of the pattern of neuron connectivity in the human brain, in which individual neurons respond only to stimuli in the receptive field. These receptive areas collectively overlap the entire visual area. The initial parameters to be known are the elements that are a significant part in the operation of Convolutional Neural Networks.

- Input Image
- CNN
- Output Label (Image Class)

CHAPTER - 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Overall Description:

This SRS is an overview of the whole project scenario. This document is to present a detailed description of the course management system. It will explain the purpose and features of the system, the interfaces of the system, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both stakeholders and developers of the system.

3.2. Operating Environment:

Software Requirements:

Operating System	:	Windows 7 (Min)
Front End	:	Streamlit
Back End	:	Python
Database	:	Skin Cancer ISIC

Hardware Requirements:

Processor	:	Intel Pentium® Dual Core Processor (Min)
Speed	:	2.9 GHz (Min)
RAM	:	2 GB (Min)
Hard Disk	:	2 GB (Min)

3.3 Functional Requirements:

User Functionality:

- The user will be able to upload images regarding the types of cancer he/she wants to find.
- The user will be able to insert images up to 200MB in size.
- The user can see information regarding the cancer such as its genus for example: color of cancer spot and shape of the spot.

Admin Functionality:

- The admin manages the website.
- The admin can increase database size.
- The admin can make changes to the website such as modifying the UI and making it more interactive than earlier.
- The admin can implement a better algorithm if at all a better algorithm is created in future.

3.4 Non-Functional Requirements:

3.4.1 Performance Requirements:

Performance requirements refer to static numerical requirements placed on the interaction between the users and the software.

Response Time:

Average response time shall be less than 5 sec.

Recovery Time:

In case of system failure, the redundant system shall resume operations within 30 secs. Average repair time shall be less than 45 minutes.

Start-Up/Shutdown Time:

The system shall be operational within 1 minute of starting up.

Capacity:

The system accommodates 1000 Concurrent Users.

Utilization of Resources:

The system shall store in the database no more than 9 different types of skin cancer with room for improvement.

3.4.2 Safety Requirements:

-NA-

3.4.3 Security Requirements:

The model will be running on a secure website i.e., an HTTPS website and on a secure browser such as Google Chrome, Brave, etc.

3.4.4 Software Quality Attributes:

Reliability:

The system shall be reliable i.e., in case the webpage crashes, progress will be saved.

Availability:

The website will be available to all its users round the clock i.e., they can access the website at any time.

Security:

The model will be running on a secure website i.e., an HTTPS website and on a secure browser such as Google Chrome, Brave, etc.

Maintainability:

The model shall be designed in such a way that it will be very easy to maintain it in future. Our model is a neural network model and a web-based system and will depend much on the web server and on the neural networks. However, the web application will be designed using Streamlit which is based on neural network approach and proper database modeling along with extensive documentation which will make it easy to develop, troubleshoot and maintain in future.

Usability:

The interfaces of the system will be user friendly enough that every user will be able to use it easily.

Scalability:

The system will be designed in such a way that it will be extendable. If more types or algorithms are going to be added in the system, then it would easily be done. The same system can also be developed to become a mobile application rather than just a website.

CHAPTER-4

SYSTEM DESIGN

Use case Diagram:

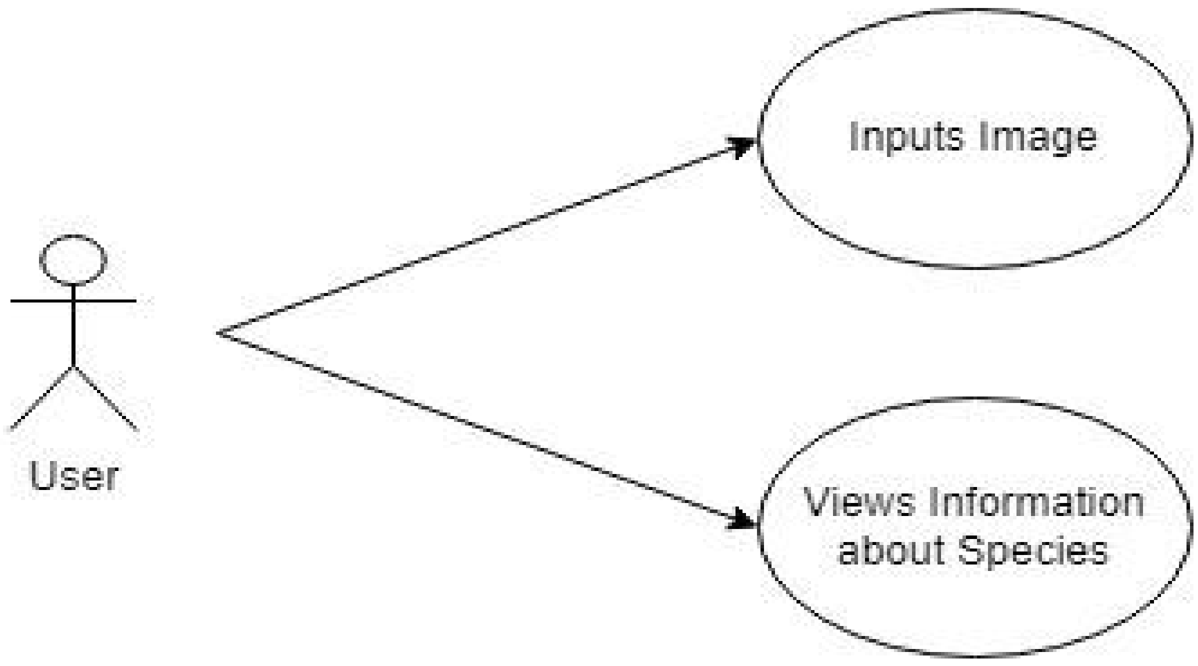


Fig 4.1: Use case diagram for User

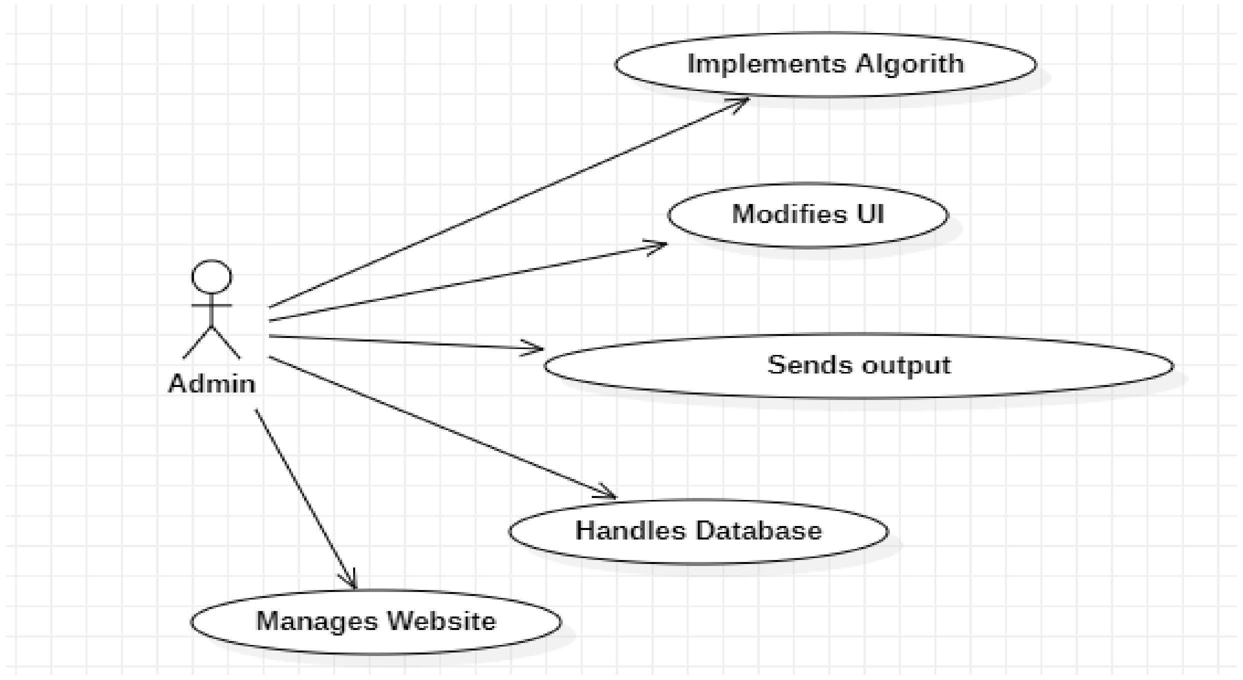


Fig 4.2: Use case diagram for Admin

Class Diagram:

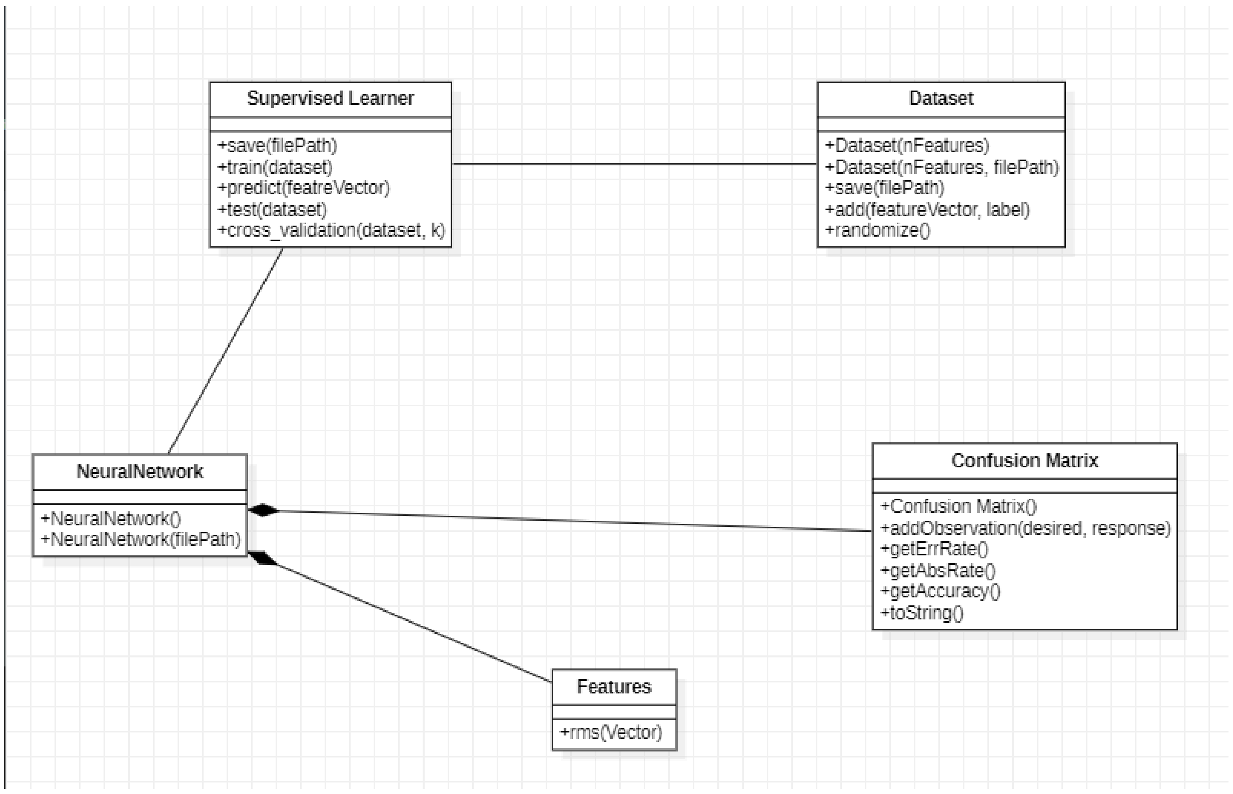


Fig.4.3 : Class diagram for cancer Identification.

Sequence Diagram:

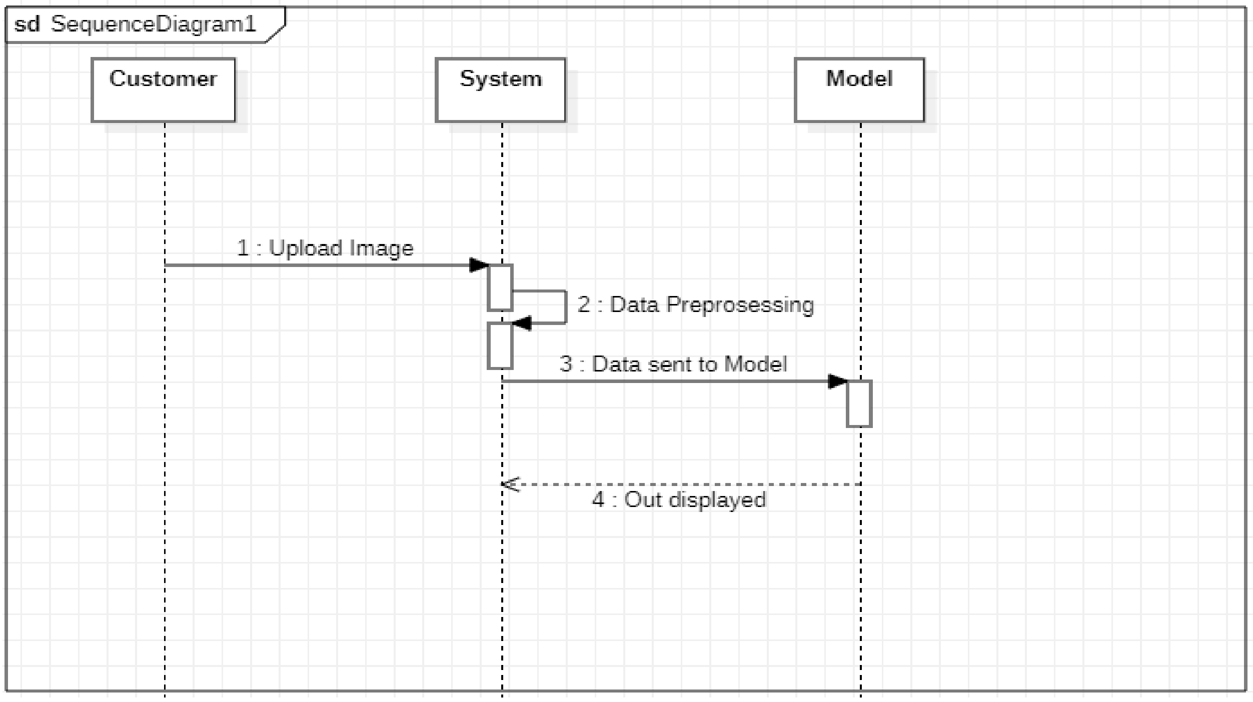
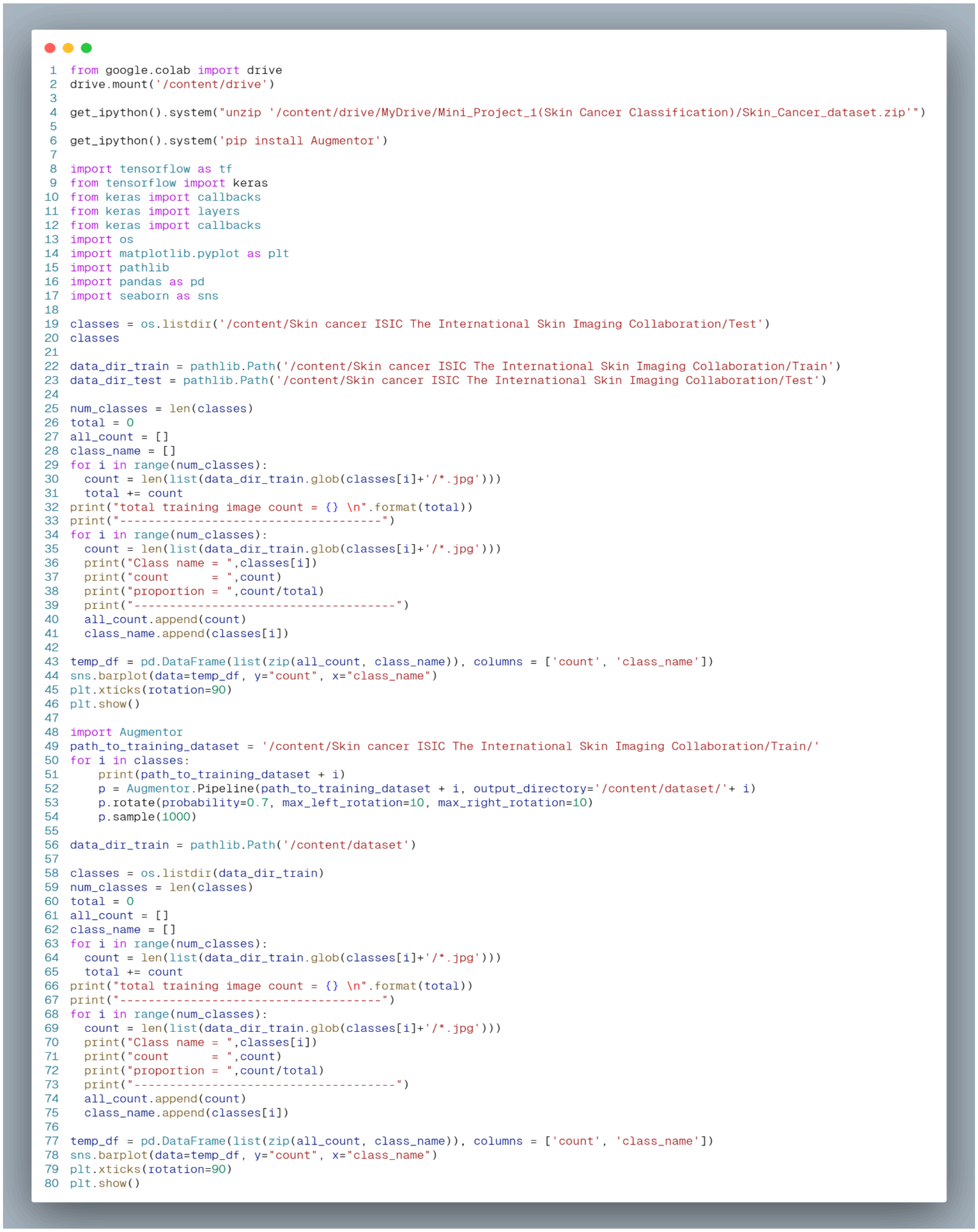


Fig4.3 : Sequence Diagram for Cancer Identification.

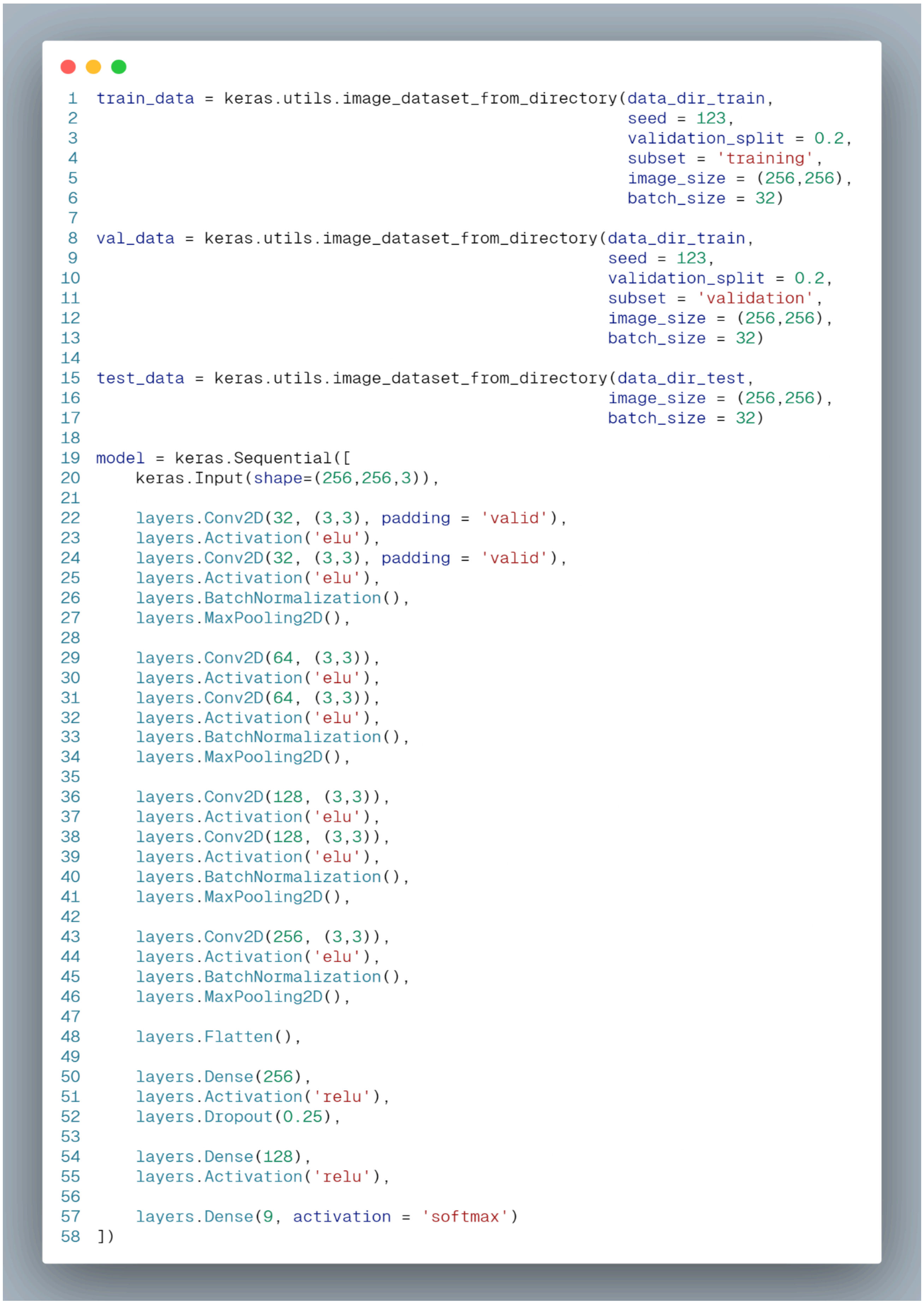
CHAPTER – 5

IMPLEMENTATION

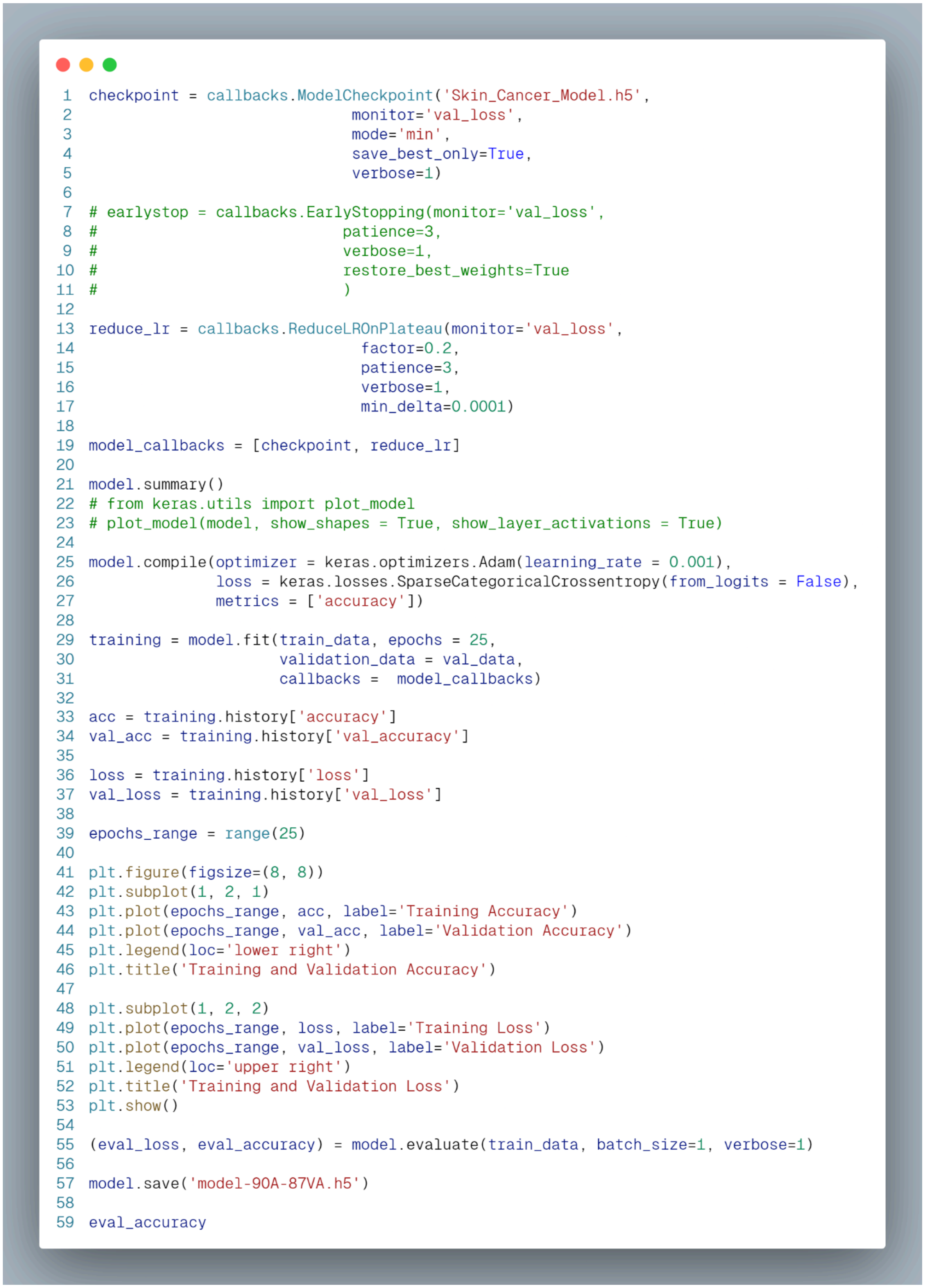
5.1 SAMPLE CODE



```
from google.colab import drive
drive.mount('/content/drive')

get_ipython().system("unzip '/content/drive/MyDrive/Mini_Project_1(Skin Cancer Classification)/Skin_Cancer_dataset.zip'")

get_ipython().system('pip install Augmentor')

import tensorflow as tf
from tensorflow import keras
from keras import callbacks
from keras import layers
from keras import callbacks
import os
import matplotlib.pyplot as plt
import pathlib
import pandas as pd
import seaborn as sns

classes = os.listdir('/content/Skin cancer ISIC The International Skin Imaging Collaboration/Test')
classes

data_dir_train = pathlib.Path('/content/Skin cancer ISIC The International Skin Imaging Collaboration/Train')
data_dir_test = pathlib.Path('/content/Skin cancer ISIC The International Skin Imaging Collaboration/Test')

num_classes = len(classes)
total = 0
all_count = []
class_name = []
for i in range(num_classes):
  count = len(list(data_dir_train.glob(classes[i]+'/*.jpg')))
  total += count
print("total training image count = {} \n".format(total))
print("-------------------------------------")
for i in range(num_classes):
  count = len(list(data_dir_train.glob(classes[i]+'/*.jpg')))
  print("Class name = ",classes[i])
  print("count      = ",count)
  print("proportion = ",count/total)
  print("-------------------------------------")
  all_count.append(count)
  class_name.append(classes[i])

temp_df = pd.DataFrame(list(zip(all_count, class_name)), columns = ['count', 'class_name'])
sns.barplot(data=temp_df, y="count", x="class_name")
plt.xticks(rotation=90)
plt.show()

import Augmentor
path_to_training_dataset = '/content/Skin cancer ISIC The International Skin Imaging Collaboration/Train/'
for i in classes:
    print(path_to_training_dataset + i)
    p = Augmentor.Pipeline(path_to_training_dataset + i, output_directory='/content/dataset/'+ i)
    p.rotate(probability=0.7, max_left_rotation=10, max_right_rotation=10)
    p.sample(1000)

data_dir_train = pathlib.Path('/content/dataset')

classes = os.listdir(data_dir_train)
num_classes = len(classes)
total = 0
all_count = []
class_name = []
for i in range(num_classes):
  count = len(list(data_dir_train.glob(classes[i]+'/*.jpg')))
  total += count
print("total training image count = {} \n".format(total))
print("-------------------------------------")
for i in range(num_classes):
  count = len(list(data_dir_train.glob(classes[i]+'/*.jpg')))
  print("Class name = ",classes[i])
  print("count      = ",count)
  print("proportion = ",count/total)
  print("-------------------------------------")
  all_count.append(count)
  class_name.append(classes[i])

temp_df = pd.DataFrame(list(zip(all_count, class_name)), columns = ['count', 'class_name'])
sns.barplot(data=temp_df, y="count", x="class_name")
plt.xticks(rotation=90)
plt.show()
```

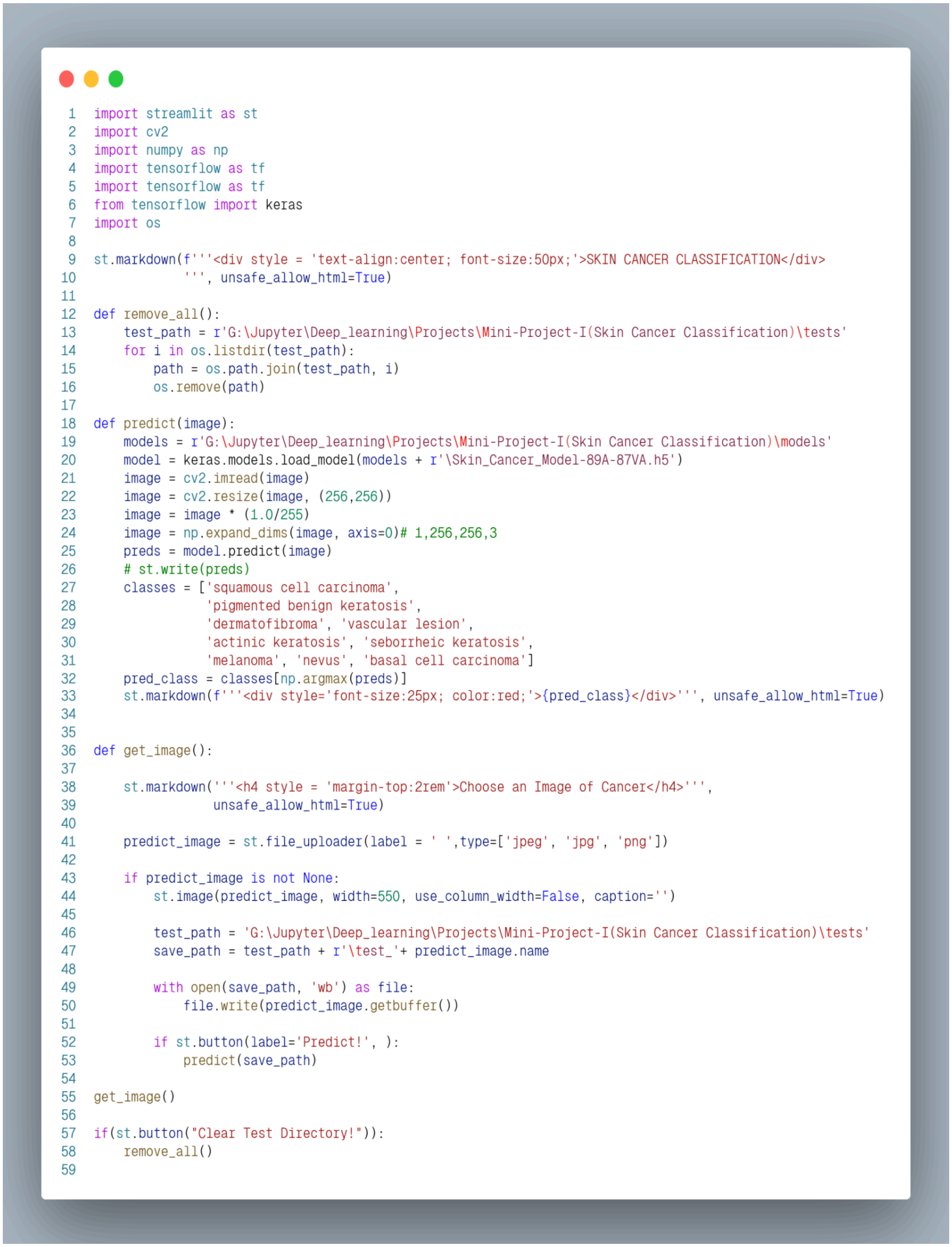


```python
train_data = keras.utils.image_dataset_from_directory(data_dir_train,
                                                      seed = 123,
                                                      validation_split = 0.2,
                                                      subset = 'training',
                                                      image_size = (256,256),
                                                      batch_size = 32)

val_data = keras.utils.image_dataset_from_directory(data_dir_train,
                                                    seed = 123,
                                                    validation_split = 0.2,
                                                    subset = 'validation',
                                                    image_size = (256,256),
                                                    batch_size = 32)

test_data = keras.utils.image_dataset_from_directory(data_dir_test,
                                                     image_size = (256,256),
                                                     batch_size = 32)

model = keras.Sequential([
    keras.Input(shape=(256,256,3)),

    layers.Conv2D(32, (3,3), padding = 'valid'),
    layers.Activation('elu'),
    layers.Conv2D(32, (3,3), padding = 'valid'),
    layers.Activation('elu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(),

    layers.Conv2D(64, (3,3)),
    layers.Activation('elu'),
    layers.Conv2D(64, (3,3)),
    layers.Activation('elu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(),

    layers.Conv2D(128, (3,3)),
    layers.Activation('elu'),
    layers.Conv2D(128, (3,3)),
    layers.Activation('elu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(),

    layers.Conv2D(256, (3,3)),
    layers.Activation('elu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(),

    layers.Flatten(),

    layers.Dense(256),
    layers.Activation('relu'),
    layers.Dropout(0.25),

    layers.Dense(128),
    layers.Activation('relu'),

    layers.Dense(9, activation = 'softmax')
])
```



```python
checkpoint = callbacks.ModelCheckpoint('Skin_Cancer_Model.h5',
                             monitor='val_loss',
                             mode='min',
                             save_best_only=True,
                             verbose=1)

# earlystop = callbacks.EarlyStopping(monitor='val_loss',
#                            patience=3,
#                            verbose=1,
#                            restore_best_weights=True
#                            )

reduce_lr = callbacks.ReduceLROnPlateau(monitor='val_loss',
                              factor=0.2,
                              patience=3,
                              verbose=1,
                              min_delta=0.0001)

model_callbacks = [checkpoint, reduce_lr]

model.summary()
# from keras.utils import plot_model
# plot_model(model, show_shapes = True, show_layer_activations = True)

model.compile(optimizer = keras.optimizers.Adam(learning_rate = 0.001),
              loss = keras.losses.SparseCategoricalCrossentropy(from_logits = False),
              metrics = ['accuracy'])

training = model.fit(train_data, epochs = 25,
                     validation_data = val_data,
                     callbacks =  model_callbacks)

acc = training.history['accuracy']
val_acc = training.history['val_accuracy']

loss = training.history['loss']
val_loss = training.history['val_loss']

epochs_range = range(25)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()

(eval_loss, eval_accuracy) = model.evaluate(train_data, batch_size=1, verbose=1)

model.save('model-90A-87VA.h5')

eval_accuracy
```

server.py:



```python
import streamlit as st
import cv2
import numpy as np
import tensorflow as tf
import tensorflow as tf
from tensorflow import keras
import os

st.markdown(f'''<div style = 'text-align:center; font-size:50px;'>SKIN CANCER CLASSIFICATION</div>
            ''', unsafe_allow_html=True)

def remove_all():
    test_path = r'G:\Jupyter\Deep_learning\Projects\Mini-Project-I(Skin Cancer Classification)\tests'
    for i in os.listdir(test_path):
        path = os.path.join(test_path, i)
        os.remove(path)

def predict(image):
    models = r'G:\Jupyter\Deep_learning\Projects\Mini-Project-I(Skin Cancer Classification)\models'
    model = keras.models.load_model(models + r'\Skin_Cancer_Model-89A-87VA.h5')
    image = cv2.imread(image)
    image = cv2.resize(image, (256,256))
    image = image * (1.0/255)
    image = np.expand_dims(image, axis=0)# 1,256,256,3
    preds = model.predict(image)
    # st.write(preds)
    classes = ['squamous cell carcinoma',
               'pigmented benign keratosis',
               'dermatofibroma', 'vascular lesion',
               'actinic keratosis', 'seborrheic keratosis',
               'melanoma', 'nevus', 'basal cell carcinoma']
    pred_class = classes[np.argmax(preds)]
    st.markdown(f'''<div style='font-size:25px; color:red;'>{pred_class}</div>''', unsafe_allow_html=True)


def get_image():

    st.markdown('''<h4 style = 'margin-top:2rem'>Choose an Image of Cancer</h4>''',
                unsafe_allow_html=True)

    predict_image = st.file_uploader(label = ' ',type=['jpeg', 'jpg', 'png'])

    if predict_image is not None:
        st.image(predict_image, width=550, use_column_width=False, caption='')

        test_path = 'G:\Jupyter\Deep_learning\Projects\Mini-Project-I(Skin Cancer Classification)\tests'
        save_path = test_path + r'\test_'+ predict_image.name

        with open(save_path, 'wb') as file:
            file.write(predict_image.getbuffer())

        if st.button(label='Predict!', ):
            predict(save_path)

get_image()

if(st.button("Clear Test Directory!")):
    remove_all()
```

CHAPTER – 6

TESTING

6.1 TEST CASES

Test Case to check whether the required Software is installed on the systems

Test Case ID:	1
Test Case Name:	Required Software Testing
Purpose:	To check whether the required Software is installed on the systems
Input:	Enter python command
Expected Result:	Should Display the version number for the python
Actual Result:	Displays python version
Failure	If the python environment is not installed, then the Deployment fails

Table 6.1.1 python Installation verification

Test Case to check Program Integration Testing

Test Case ID:	2
Test Case Name:	Programs Integration Testing
Purpose:	To ensure that all the modules work together
Input:	All the modules should be accessed.
Expected Result:	All the modules should be functioning properly.
Actual Result:	All the modules should be functioning properly.
Failure	If any module fails to function properly, the implementation fails.

Table 6.1.2 python Programs Integration Testing

Test Case to Collect Dataset and Load the Dataset

Test Case ID:	3
Test Case Name:	Collect Dataset and Load the Dataset
Purpose:	Check Dataset is collected, and the data is stored
Input:	Provide Dataset as input
Expected Result:	Dataset is collected and view the Dataset and store the Dataset
Actual Result:	Load the Dataset and view the Dataset and store
Failure	If the dataset is not loaded, it will throw an error.

Table 6.1.3 Collect Dataset and Load the Dataset

Test Case to check whether the types are recognized

Test Case ID:	4
Test Case Name:	skin cancer Recognition
Purpose:	skin cancer Recognition using CNN
Input:	Provide dataset and input an image
Expected Result:	After Evaluation I get the cancer name
Actual Result:	After Evaluation I get the cancer name
Failure	If the data is not Evaluated, it does not display the gesture

Table 6.1.4 Skin Cancer Types Recognition

CHAPTER - 7

SCREENSHOTS

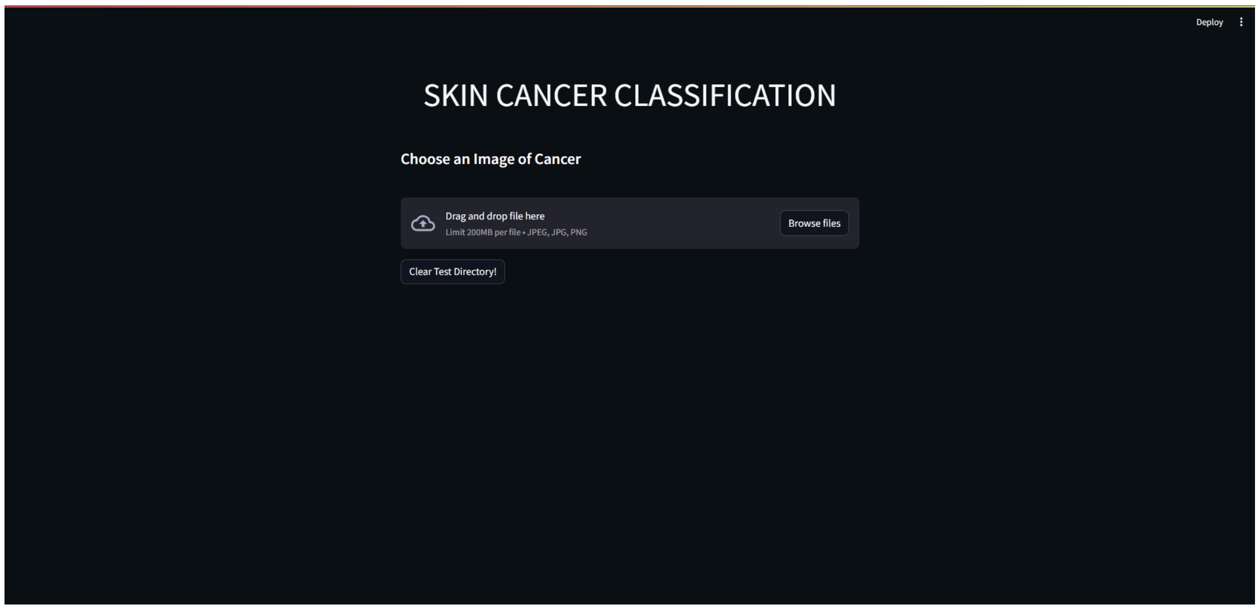

Deploy
SKIN CANCER CLASSIFICATION
Choose an Image of Cancer
Drag and drop file here
Limit 200MB per file • JPEG, JPG, PNG
Browse files
Clear Test Directory!

Figure 7.1 showcasing interface

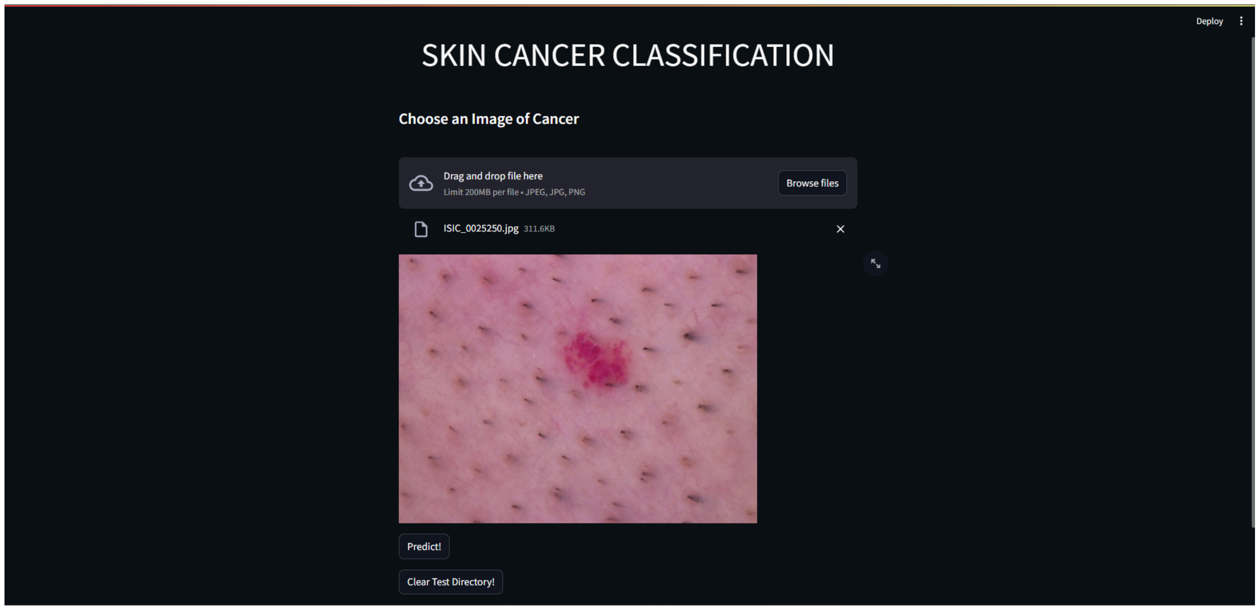
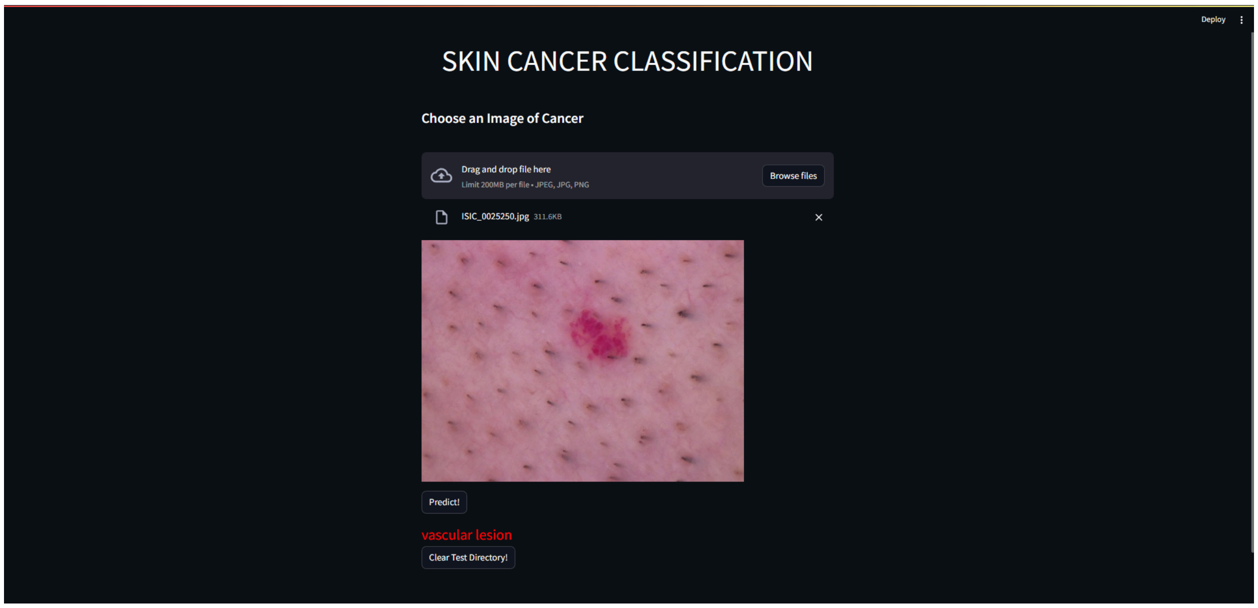


Figure 7.2 Uploading image of Vascular Lesion



SKIN CANCER CLASSIFICATION

Choose an Image of Cancer

Drag and drop file here
Limit 200MB per file • JPEG, JPG, PNG

ISIC_0025250.jpg 311.6KB

vascular lesion

Figure 7.3 detected vascular lesion

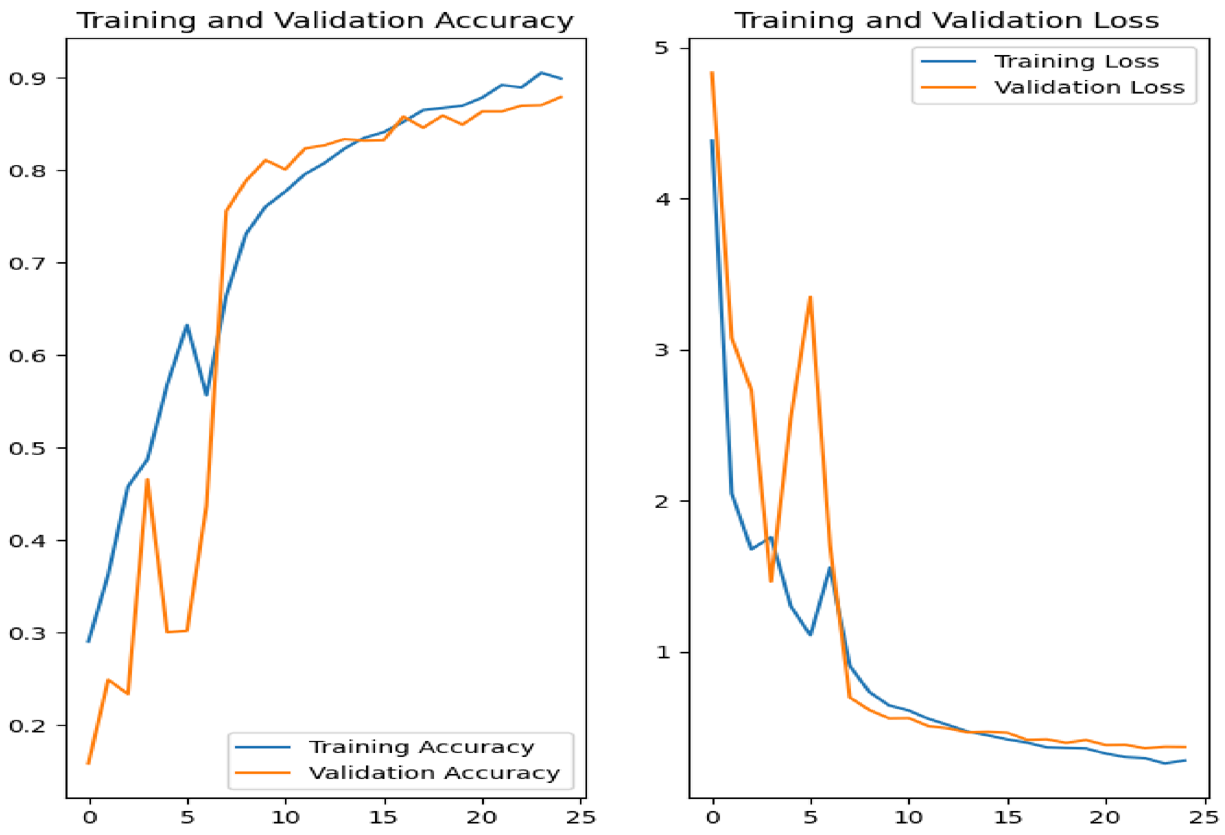


Figure 7.4 Training and Testing Accuracy and losses

CHAPTER - 8

CONCLUSION AND FUTURE SCOPE

In conclusion, the documentation presented herein serves as a comprehensive resource for understanding and implementing skin cancer detection systems utilizing artificial intelligence and machine learning techniques. We have addressed various aspects of skin cancer diagnosis, from data collection and preprocessing to model development, evaluation, integration, and ethical considerations. Through this documentation, healthcare professionals, researchers, and developers gain insights into the challenges and opportunities in automated skin cancer detection. By leveraging advanced algorithms and large-scale datasets, AI-based systems have demonstrated promising results in improving diagnostic accuracy and efficiency.

FUTURE SCOPE:

Enhanced Model Performance: Continuously improving the accuracy and robustness of skin cancer detection models through advanced deep learning architectures, data augmentation techniques, and transfer learning from related domains.

Multimodal Imaging: Exploring the integration of multiple imaging modalities, such as dermoscopy, reflectance confocal microscopy, and multispectral imaging, to enhance diagnostic capabilities and provide complementary information.

Real-Time Diagnosis: Developing real-time skin cancer detection systems suitable for point-of-care settings, telemedicine applications, and mobile devices to enable timely diagnosis and treatment.

Clinical Validation: Conducting rigorous clinical validation studies to assess the real-world performance of automated skin cancer detection systems and their impact on patient outcomes and healthcare workflows.

BIBLIOGRAPHY

[1] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115–118. [DOI: 10.1038/nature21056]

[2] Codella, N., Gutman, D., Celebi, M. E., Helba, B., & Marchetti, M. A. (2018). Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), Hosted by the International Skin Imaging Collaboration (ISIC). In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (pp. 1680–1688). [DOI: 10.1109/CVPRW.2018.00230]

[3] Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text Classification Algorithms: A Survey. Information, 10(4), 150. [DOI: 10.3390/info10040150]

[4] **Streamlit**: The fastest way to build custom ML tools. Retrieved from https://www.streamlit.io/

[5] **ISIC - International Skin Imaging Collaboration**. Retrieved from https://www.isic-archive.com/#!/topWithHeader/tightContentTop/about/isicArchive

[6] **TensorFlow**: Large-scale machine learning on heterogeneous systems. Retrieved from https://www.tensorflow.org/

[7] Chollet, F., et al. (2015). Keras. Retrieved from https://keras.io/

[8] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

APPENDIX A: TOOLS AND TECHNOLOGIES

- PYTHON V3: The Python language comes with many libraries and frameworks that make coding easy. This also saves a significant amount of time.

- JUPYTER NOTEBOOK: The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text.

- TENSORFLOW: TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML poIred applications.

- NUMPY: NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, fourier transform, and matrices.

- WINDOWS 11: Windows 11 was used as the operating system.

- KERAS: Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

- STREAMLIT: Streamlit is an open source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.