**INSTITUTE OF AERONAUTICAL ENGINEERING**

DUNDIGAL, HYDERABAD TELANGANA 500043

(AUTONOMOUS)

**JUNE 2023**

****

**ON**

**DETECTING PLANT SKIN DISEASE**

**TWO WEEKS SUMMER INTERNSHIP**

**Submitted by**

**CIRIPURAM ABHIGNA [20951A1202]**

**INFORMATION TECHNOLOGY**

**Under the guidance of**

**INV Suryanarayana sir**

**(Associate professor,Department of computer science and eng****ineering)**

**DECLARATION**

I declare that this project report titled ‘DETECTING PLANT DISEASE’ Submitted in fulfilment of the requirement for the award of Bachelor of Technology in Information Technology is a record of original work carried out by me under the supervision of MR.INV SURYANARAYANA, and has not formed the basis of the award of any other degree or diploma, in this or any other Institute or University. In Keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

**CIRIPURAM ABHIGNA [20951A1202]**

**ABSTRACT**

Melanoma, also referred to as malignant melanoma, is a type of skin cancer Caused by Abnormal multiplication of pigment producing cells that give color to the skin.Despite significant death rate, early stage detected melanoma is curable in most cases. Meanwhile differentiation between melanoma and other benign moles in their initial growth phases is a challenging task even for experienced dermatologists. Computerised algorithms are being developed for this purpose. But professional decision making, in this regard, requires sophisticated algorithms and equipment. There are various methods in dermatology such as ABCDE (asymmetry, border irregularity, color patterns, and diameter, evolution) rule that guide physicians in this task.

**CONTENTS**

**Candidate’s Declaration** i

**Acknowledgement** ii

**Abstract** iii

**Contents**  iv

**List of Figures** v

**1. INTRODUCTION**

1.1 Motivation 1

1.2 Problem Definition 2

1.3 Objective of Project 2

1.4 Limitations of Project 2

1.5 Organization of Documentation 3

**2. LITERATURE SURVEY**

2.1 Introduction 4

* + 1. Deep Learning in Healthcare 4

2.2 Existing System 5

2.3 Disadvantages of Existing system 6

2.4 Proposed System 6

2.4.1 EfficientNet Architecture

**3. ANALYSIS**

3.1 Introduction 9

3.2 Software Requirement Specification 9

3.2.1 User requirements 9

3.2.2 Software requirements 10

3.2.3 Hardware requirements 10

3.3 Content Diagram of Project 10

3.4 Flowcharts 11

**4. DESIGN**

4.1 Introduction 12

4.2 UML diagrams 12

4.2.1 DFD Diagram 12

4.2.2 Use Case Diagram 15

4.2.3 Sequence Diagram 16

4.2.4 Class Diagram 17

4.2.5 Activity Diagram 19

4.3 Module design and organization 19

**5. IMPLEMENTATION & RESULTS**

5.1 Introduction 20

5.2 Explanation of Key functions 20

5.2.1 Dataset Description 20

5.2.2 Source Code 22

5.2.3 Result Analysis 23

**6. TESTING & VALIDATION**

6.1 Introduction 26

6.2 Design of test cases and scenarios 28

6.3 Validation 29

**7. CONCLUSION & FUTURE WORK**

7.1 Conclusion 31

7.2 Future Work 31

**8. REFERENCES 32**

**LIST OF FIGURES**

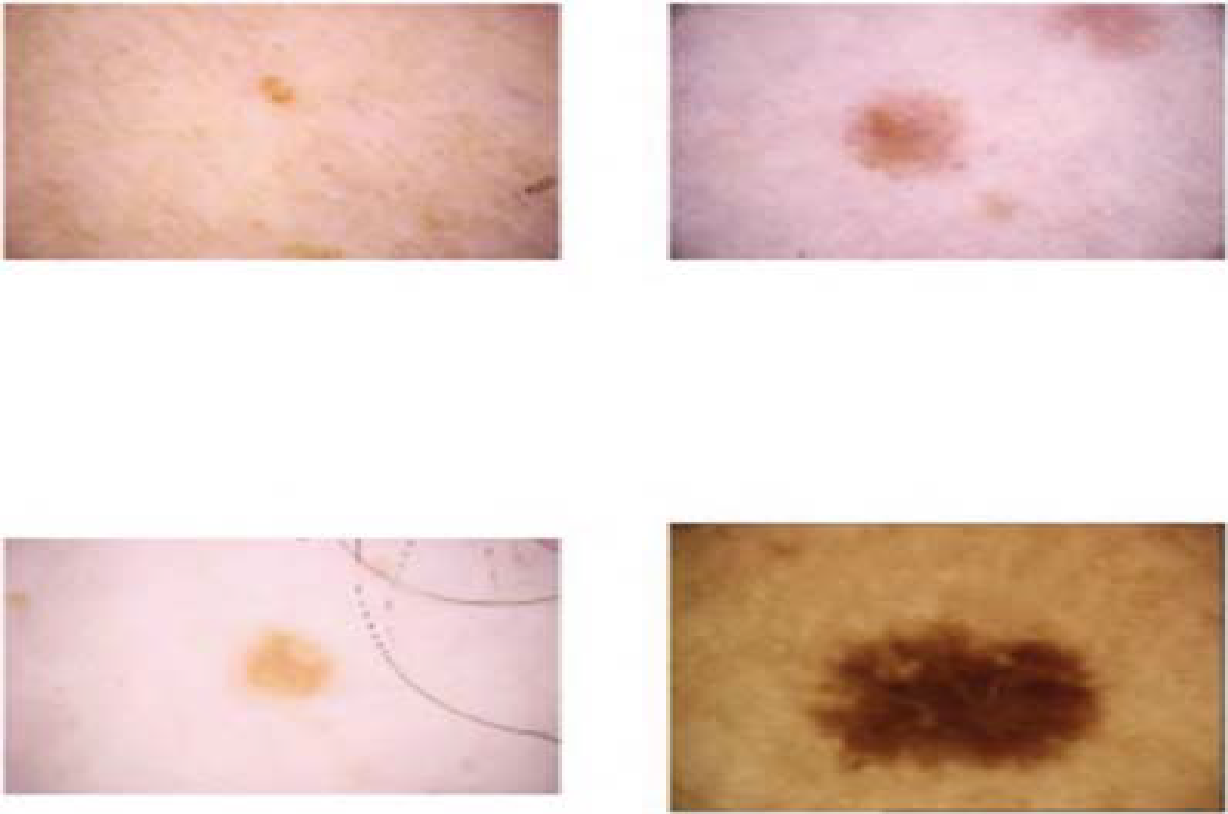
|  |  |  |
| --- | --- | --- |
| **Figure No** | **Figure Name** | **Page No** |
| Figure 1.1a  Figure 1.1b  Figure 1.1c  Figure 1.1d  Figure 2.1 | Slight mole of a patient  Early stage of malignant melanoma lesion  Slight melanoma lesion  Malignant melanoma lesion  Deep Learning | 1  1  1  2  5 |
| Figure 3.1 | Architecture of Melanoma Detection System | 11 |
| Figure 3.2 | Flowchart of project | 11 |
| Figure 4.1 | Process notation in DFD | 13 |
| Figure 4.2 | Datastore notation in DFD | 13 |
| Figure 4.3 | Dataflow notation in DFD | 13 |
| Figure 4.4 | External Entity notation in DFD | 14 |
| Figure 4.5 | Level 0 Dataflow Diagram | 14 |
| Figure 4.6 | Level 1 Dataflow Diagram | 15 |
| Figure 4.7 | Use Case Diagram | 15 |
| Figure 4.8 | Sequence Diagram | 16 |

***CHAPTER 1***

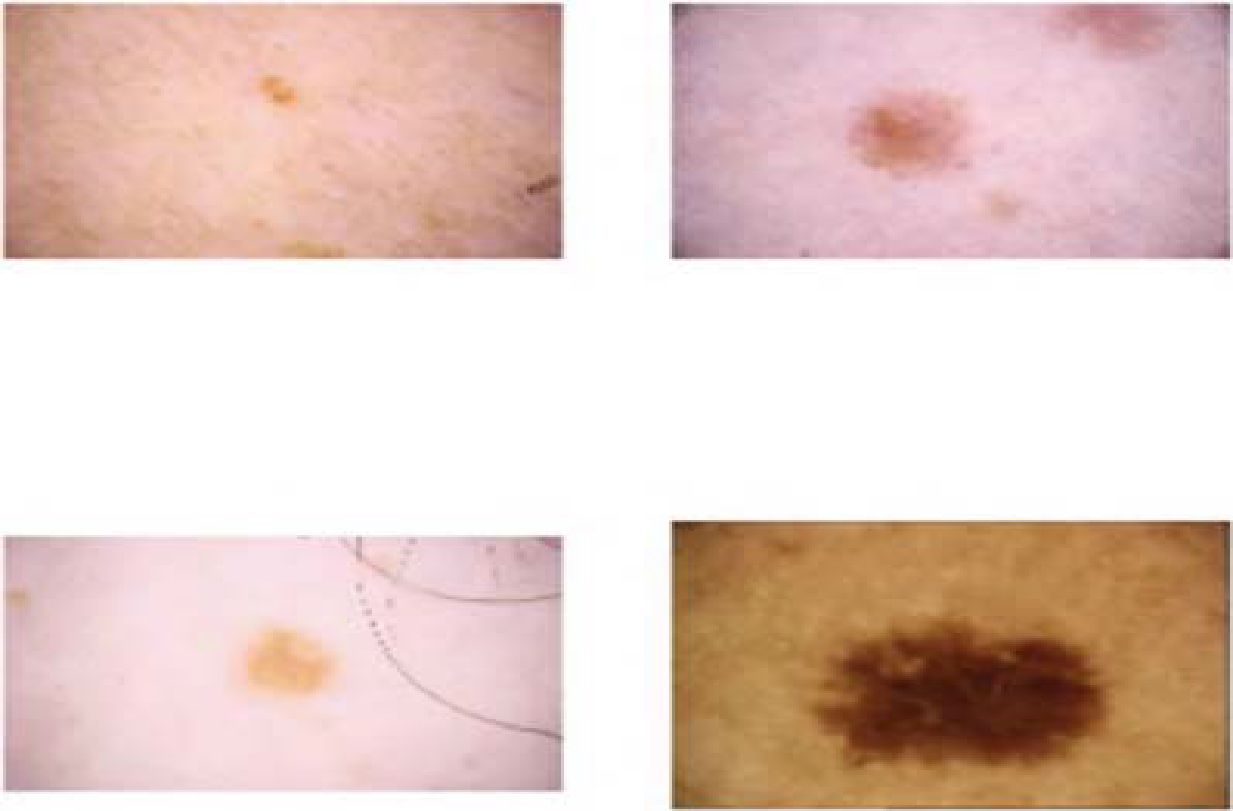
**Introduction**

* 1. **Motivation**

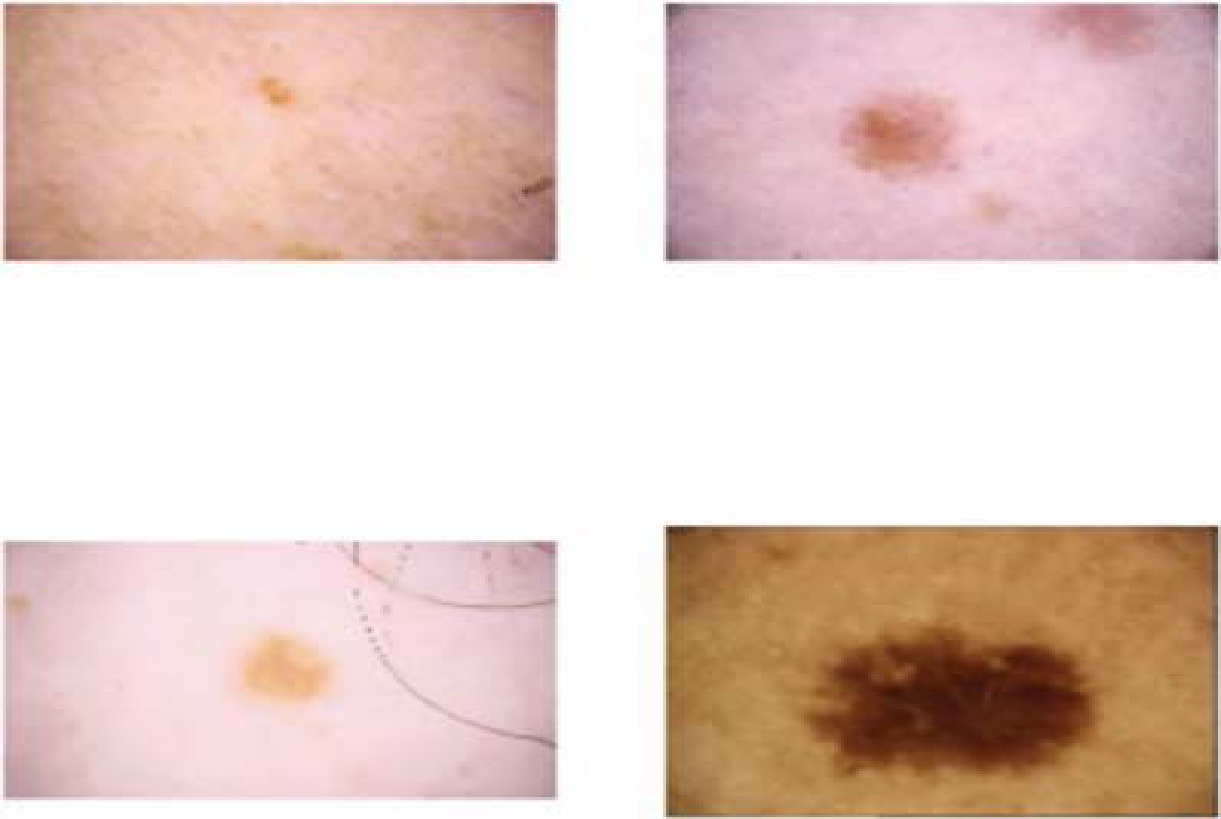
Melanoma, also referred to as malignant melanoma, is a type of skin cancer caused by abnormal multiplication of pigment producing cells that give color to the skin. Despite significant death rate, early stage detected melanoma is curable in most cases. Meanwhile differentiation between melanoma and other benign moles in their initial growth phases is a challenging task even for experienced dermatologists. Computerized algorithms are being developed for this purpose. But professional decision making, in this regard, requires sophisticated algorithms and equipment. There are various methods in dermatology such as ABCDE (asymmetry, border irregularity, color patterns, and diameter, evolution) rule that guide physicians in this task.



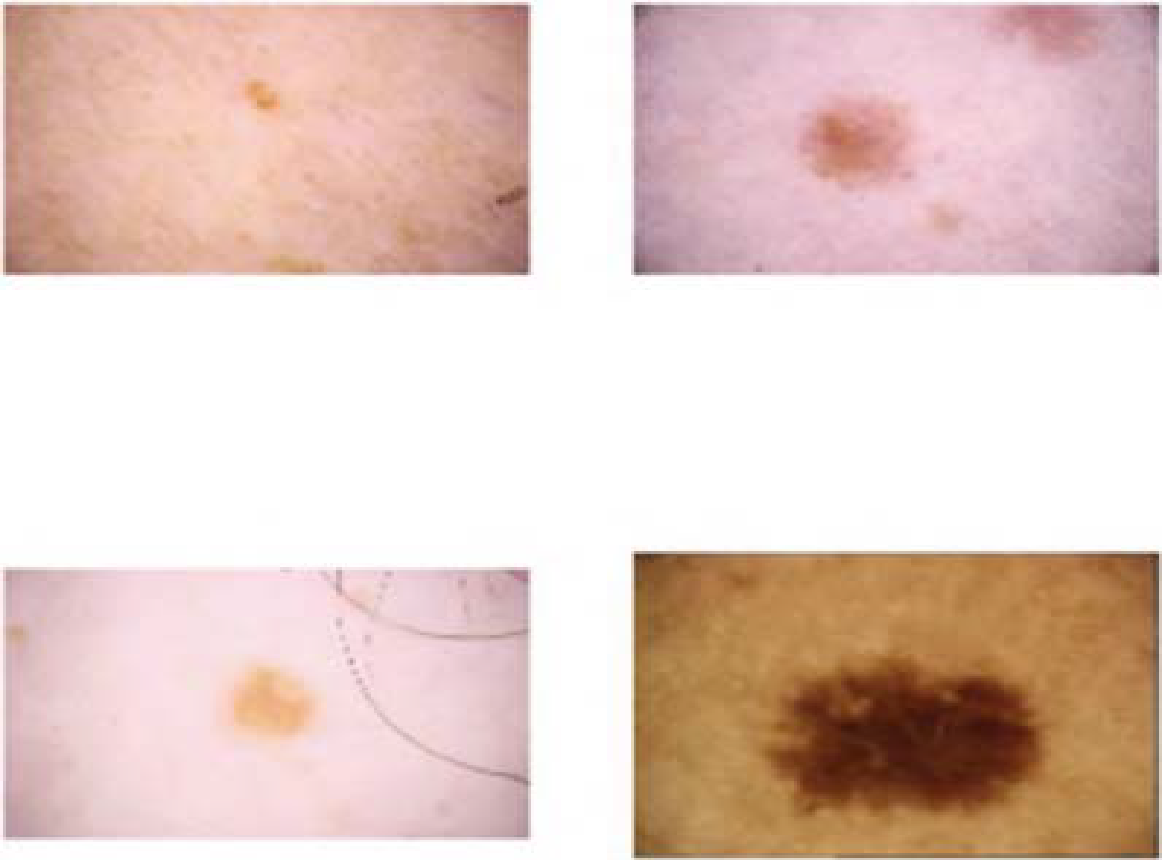
**Fig 1.1(a)**: slight mole of a patient



**Fig 1.1(b)**: early stage of malignant melanoma lesion



**Fig 1.1(c)**: slight melanoma lesion



**Fig 1.1(d)**: malignant melanoma lesion.

**1.2 Problem Definition**

To create a reliable model that can categorize various input photos as benign or melanoma instances. The development of such a system is highly helpful because it takes time to determine if a skin lesion is cancerous or not.

* 1. **Objective of Project**

The main objective of the project is to develop a framework to analyze and assess the risk of melanoma using dermatological photographs taken with a standard consumer-grade camera.

**1.4 Limitations of project**

The collection(HAM Dataset) contains about 10,000 examples of photos of skin conditions. Each category has a varied number of photographs. When photos from one category outnumber those from all other categories, the classification is skewed in favour of that category. As a result, unbalanced data makes the model less accurate. These restrictions are largely addressed during the data pre-processing stage. The only limitation on this project is therefore this.

* 1. **Organisation of documentation**

Organization of documentation explains about gist of what other sections contains. Chapter 2 explains about the background work done. Chapter 3 gives information regarding analysis to be done, Chapter 4 about is regarding designing process, Chapter 5 about implementation details, Chapter 6 discusses about testing methodology, Chapter 7 is about conclusions and future work which can be done in the project. The final Chapter 8 contains the References that have been utilized to implement this project.

***CHAPTER 2***

**Literature Survey**

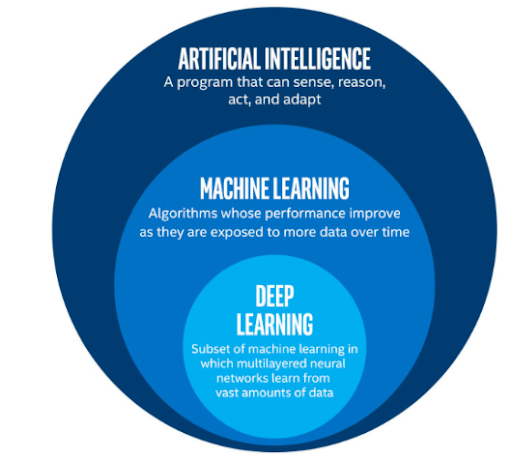
**2.1 Introduction**

Deep learning is a branch of [machine learning](https://www.netapp.com/artificial-intelligence/what-is-machine-learning). Unlike traditional machine learning algorithms, many of which have a finite capacity to learn no matter how much data they acquire, deep learning systems can improve their performance with access to more data: the machine version of more experience. After machines have gained enough experience through deep learning, they can be put to work for specific tasks such as driving a car, detecting weeds in a field of crops, detecting diseases, inspecting machinery to identify faults, and so on. Deep learning networks learn by discovering intricate structures in the data they experience. By building computational models that are composed of multiple processing layers, the networks can create multiple levels of abstraction to represent the data. Deep learning neural networks, or artificial neural networks, attempts to mimic the human brain through a combination of data inputs, weights, and bias. These elements work together to accurately recognize, classify, and describe objects within the data.

Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called *visible*layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made.

**2.1.1 Deep Learning in Healthcare**

Deep learning has been particularly effective in medical imaging, due to the availability of high-quality data and the ability of convolutional neural networks to classify images. For example, deep learning can be as effective as a dermatologist in classifying skin cancers, if not more so. Several vendors have already received FDA approval for deep learning algorithms for diagnostic purposes, including image analysis for oncology and retina diseases. Deep learning is also making significant inroads into improving healthcare quality by predicting medical events from electronic health record data.



**Fig 2.1**: Deep Learning

**2.2 Existing System**

In the past, skin cancer was diagnosed manually using biopsy-based detection techniques. This approach was painful, and the diagnosing process took a long time. For the purpose of identifying the different types of cancer in skin images, semi-automatic based machine learning algorithms are employed to reduce the drawbacks. The Fuzzy-C-Means (FCM) approach is initially employed to categorise the constrained.

Another widely used technique for detecting cancer in medical picture diagnosis is K-Means. Using this k-means clustering technique results in the initial cluster point selection challenge. Numerous studies have employed Markov Random Field (MRF) and Conditional Random Field (CRF) for the detection of cancer. Recently, SVM based methods are performing better in skin cancer identification various cancer kinds. Real-world performance of this approach is. Murugan et al., Mane et al. and Patel et al. have proposed SVM based detection techniques.

**2.3 Disadvantages of Existing System**

Skin cancer used to be manually diagnosed using biopsy-based detection methods. This method hurt, and it took a while to diagnose the problem. Another effective strategy in the realm of cancer detection is greedy fusion. These techniques take more time and are less accurate when it comes to identifying skin cancer in the real world.

**2.4 Proposed System**

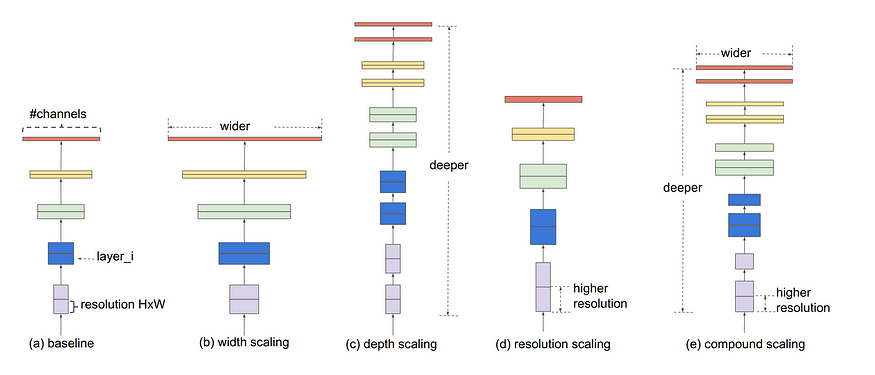
The proposed method makes use of CNN that takes an image as input taken by standard consumer-grade camera. Data augmentation step is performed which an important step to make the dataset balanced. EfficientNet architecture is used by the proposed system. The motivation behind this architecture instead of other popular CNN architecture is activated by several factors including excellent feature extraction power and efficiency of EfficientNet.

**2.4.1 EfficientNet Architecture**

**EfficientNet** is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a *compound coefficient*. Unlike conventional practice that arbitrary scales these factors, the EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients. The compound scaling method is justified by the intuition that if the input image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image.

For developing the method of compound scaling, the authors systematically studied the impacts that each scaling technique has on the model’s performance and efficiency. They figured that while scaling single dimensions helps improve model performance, balancing the scale in all the three dimensions — width, depth, and image resolution — considering the variable available resources best improve the overall model performance.

* **Depth**: it is the number of layers in the network. The deeper the network, the more likely it is to be more performant.
* **Width**: It is the number of channels in a convolutional layer. The wider the network the more it tend to capture fine-grained features and the easier it is to train.
* **Resolution**: is the resolution of the input images (width x height).



**Fig 2.2**: Different scaling methods vs. Compound scaling

***CHAPTER 3***

**Analysis**

**3.1 Introduction**

The main aim of this section is to give a detailed description about the requirements of the project and the algorithms used to develop the project.

Gathering requirements is the main attraction of the Analysis Phase. The process of gathering requirements is usually more than simply asking the users what they need and writing their answers down. Depending on the complexity of the application, the process for gathering requirements has a clearly defined process of its own. This process consists of a group of repeatable processes that utilize certain techniques to capture, document, communicate, and manage requirements.

**3.2 Software Requirement Specification**

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

A SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations.

*Functional vs Non*-*Functional Requirements:*

The difference between functional and non-functional requirements is that non-functional requirements describe how the system works, while functional requirements describe what the system should do.

**3.2.1 User requirements**

The user requirement(s) document (URD) or user requirement(s) specification is a document that specifies what the user expects the software to be able to do. The user requirements are:

Determine whether the input image is cancerous or not.

**3.2.2 Software requirements**

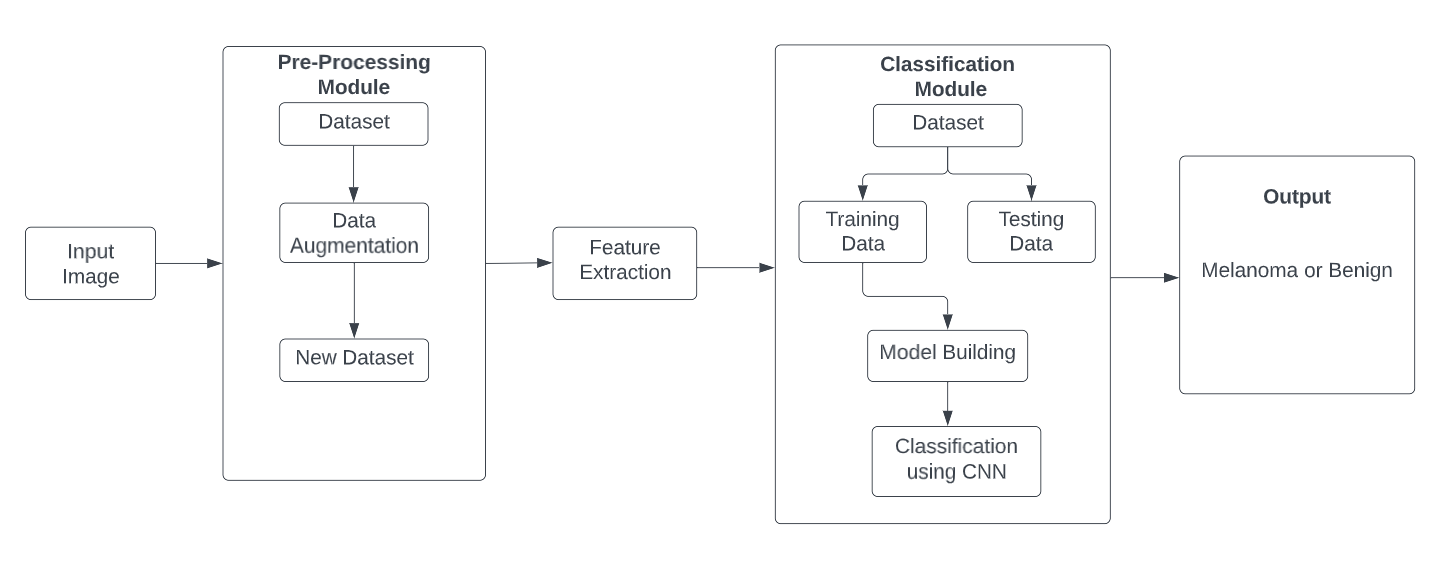
* Google Colab
* Language: Python
* Libraries:

1. Keras
2. TensorFlow
   * 1. **Hardware requirements**

* Operating System: Windows, Mac, Linux
* Processor: Intel Core i5
* RAM: 8GB(minimum)

**3.3 Content Diagram of Project**

An architecture diagram is a diagram that depicts a system that people use to abstract the software system's overall outline and build constraints, relations, and boundaries between components. It provides a complete view of the physical deployment of the evolution roadmap of the software system.

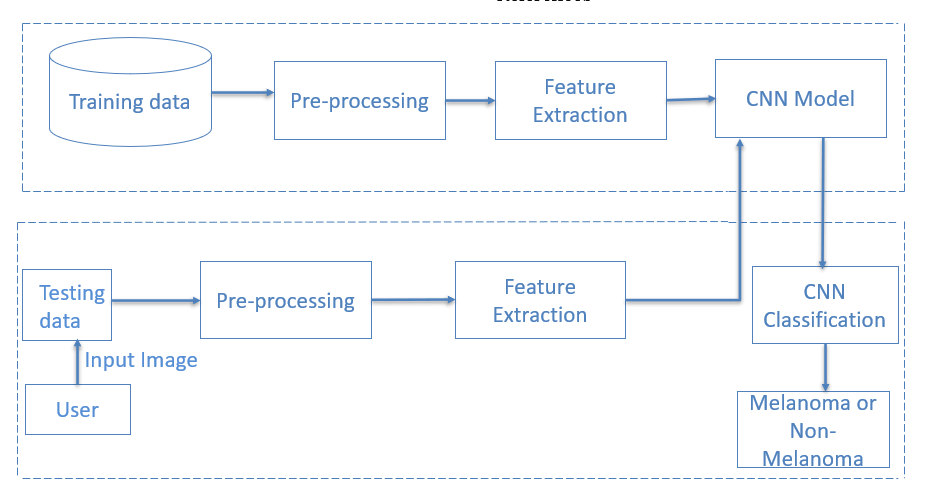


**Fig. 3.1:** Architecture of Melanoma Detection System

* 1. **Flowcharts**

A flowchart is a formalized graphic representation of a logic sequence, work or manufacturing process, organization chart, or similar formalized structure. The purpose of a flow chart is to provide people with a common language or reference point when dealing with a project or process.

Flowcharts use simple geometric symbols and arrows to define relationships.

****

**Fig. 3.2:** Flowchart of project

***CHAPTER 4***

**Design**

**4.1 Introduction**

Design is concerned with identifying software components specifying relationships among components. Specifying structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified. Design will explain software components in detail. This will help the implementation of the system.

During detailed design the internal logic of the modules specified in system design is decided. In the system design the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules.

The Unified Modelling Language is a general-purpose, developmental, modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

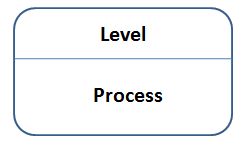
**4.2 UML diagrams**

**4.2.1 DFD Diagram**

system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it A data flow diagram (DFD) illustrates how data is processed by a gets stored.

Different notations in DFD are:

**Process Notation:** A process transforms incoming data flow into outgoing data flow.



**Fig. 4.1:** Process Notation in DFD

**Datastore Notation:** Data stores are repositories of data in the system. They are sometimes also referred to as files.

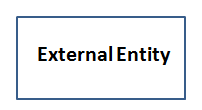


**Fig. 4.2:** Datastore Notation in DFD

**Dataflow Notation:** Data flows are pipelines through which packets of information flow. Labelled the arrows with the name of the data that moves through it.

**Fig. 4.3:** Dataflow Notation in DFD

**External Entity Notation:** External entities are objects outside the system, with which the system communicates.

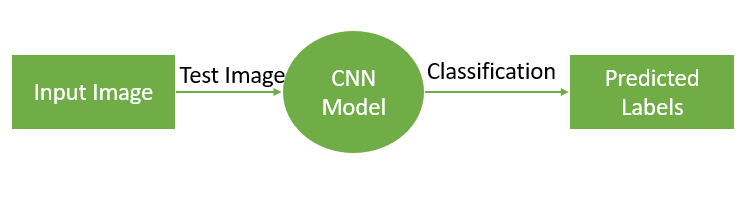




**Fig. 4.4:** External Entity Notations in DFD

**LEVEL 0 DFD:**

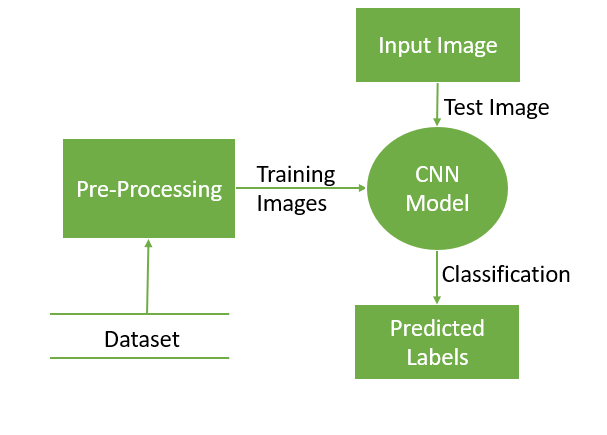
It is also known as a context diagram. It’s designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.



**Fig. 4.5:** Level 0 Dataflow Diagram for Entire Application

**LEVEL 1 DFD:**

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses.

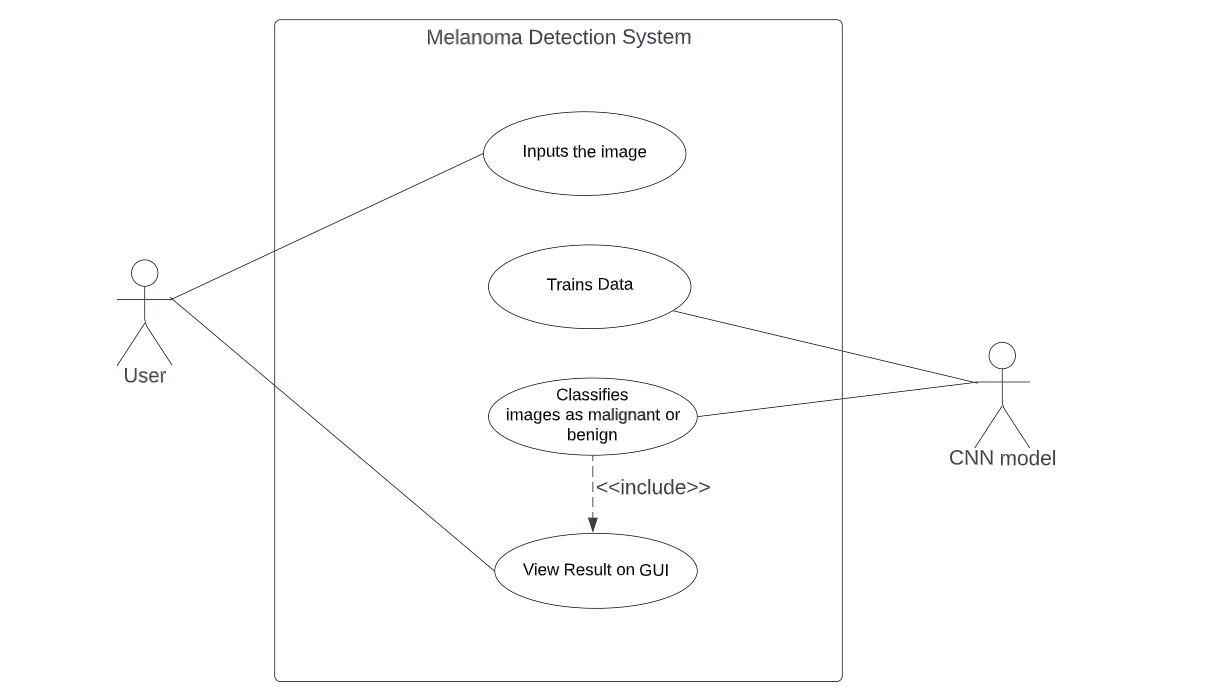


**Fig. 4.6:** Level 1 Dataflow Diagram for Entire Application

**4.2.2 Use case Diagram**

Use case Diagrams are the one of the five diagrams in the UML for modelling the dynamic aspects of systems. Use case diagrams commonly contain

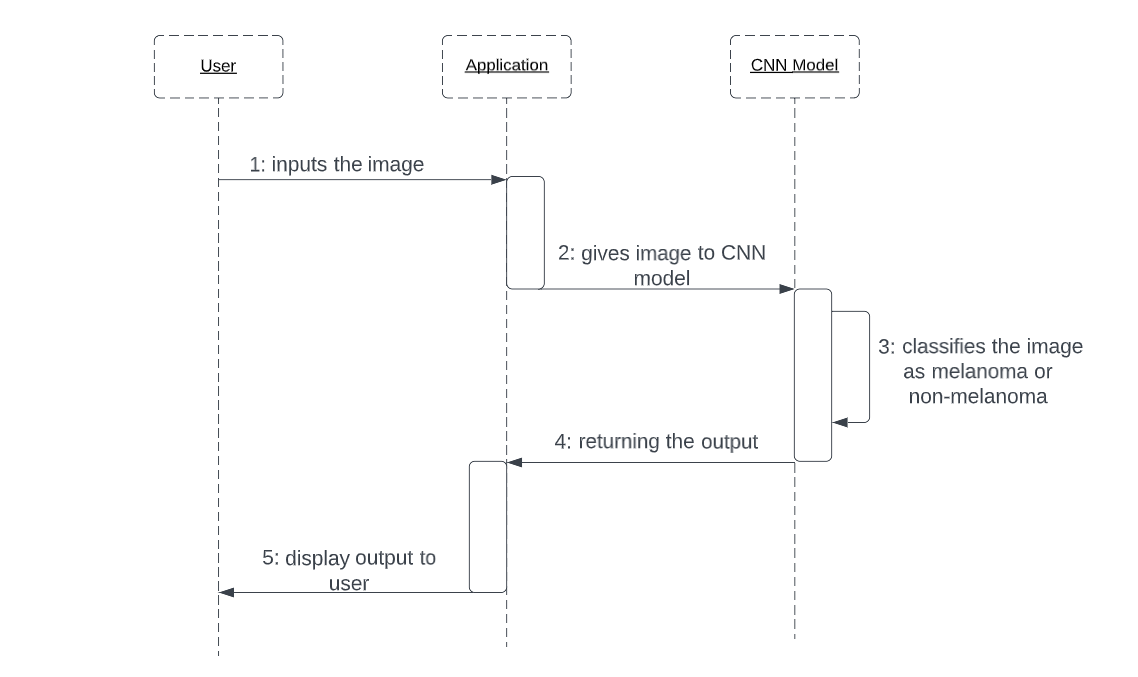
* Use cases
* Actors



**Fig. 4.7:** Use case Diagram for Entire Application

**4.2.3 Sequence Diagram**

Sequence Diagram is the most commonly used interaction diagram. A sequence diagram simply depicts interaction between objects in a sequential order. It describe how and in what order the objects in a system function.



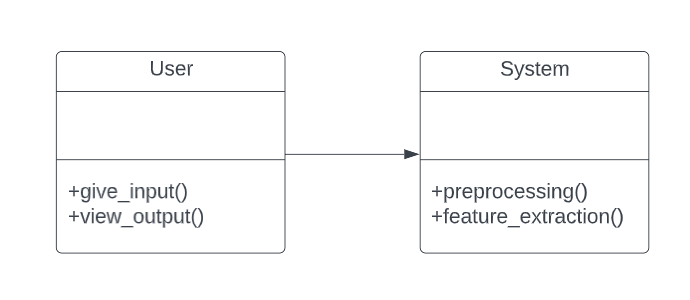
**Fig. 4.8:** Sequence Diagram for Entire Application

**4.2.4 Class Diagram**

Class Diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

**Purpose:**

* Describe responsibilities of a system.
* Forward and reverse engineering.
* Base for component and deployment diagrams.



**Fig. 4.9:** Class Diagram for Entire Application

**4.2.5 Activity Diagram**

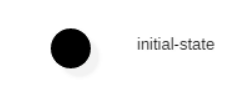
Activity Diagram is another important UML diagram to describe the dynamic aspects of the systems. It is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. It does not show any message flow from one activity to another. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering. Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination.

**Purpose:**

* Describe the sequence from one activity to another.
* Describe the parallel, branched and concurrent flow of the system.

**Notations:**

* **Initial State:** The starting stage before an activity takes place.



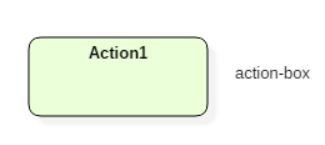
* **Final State:** The state of the system reaches when a specific process ends.

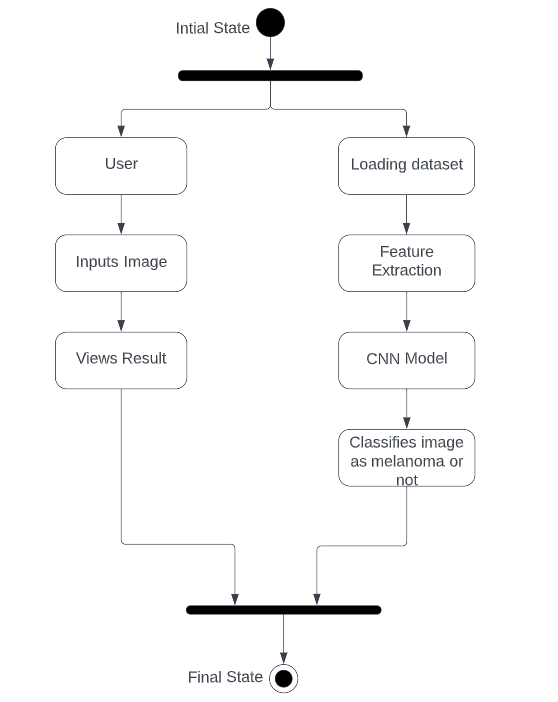
****

* **Decision Box:** It is a diamond shape box which represents a decision with alternate paths.

****

* **Action State:** It represents the non-interruptible action of objects.

****

****

**Fig. 4.10:** Activity Diagram for Entire Application

**4.3 Module Design and organization**

The various modules present are:

* User Module
* Pre-Processing Module
* Classification Module

**User module:**

The user inputs the image to the system.

**Pre-Processing module:**

Pre-processing is done in order to avoid issues caused by data imbalance. Here the main step data augmentation is done.

**Classification module:**

CNN helps in classification of the input image whether it is melanoma or non-melanoma.

***CHAPTER 5***

**Implementation And Results**

**5.1 Introduction**

Project implementation (or project execution) is the phase where visions and plans become reality. This is the logical conclusion, after evaluating, deciding, visioning, planning, applying for funds and finding the financial resources of a project.

The Project Implementation phase has two essential functions: execution of the work and proper delivery. Based on the implementation, the project’s fate is decided. The success and effectiveness can only be known after proper monitoring and feedback.

For the successful implementation of a project, you need to

1. Execute the planned details into a full-proof action plan
2. Document every little detail in this stage, from decisions to results at every step
3. Have an efficient line of communication in place across the hierarchy
4. Take quick decisions if the need for a change of plan felt instantly.
5. Form a consensus about the changes and implement immediately

**5.2 Explanation of Key Functions**

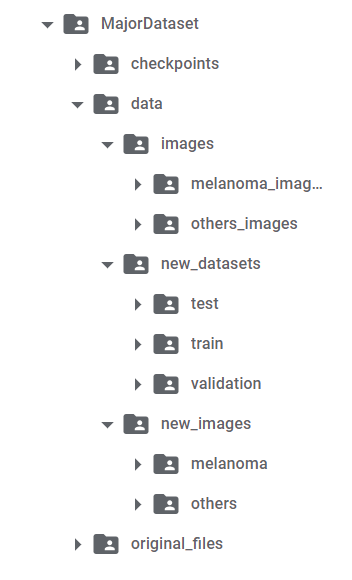
The Key feature of this application is to take the input image and predict whether it is malignant or not. So it detects the skin lesion contains melanoma or it’s just a normal mole.

**5.2.1 Dataset Description**

The dataset we are using here is HAM10000 dataset. It contains approximately 10000 skin lesions of all types. Melanoma lesions are extracted from this dataset, augmented and then divided into testing, training and validation dataset.

Project contains following files and folders:

1. original\_files folder: It contains the csv file of HAM10000 Dataset.
2. data folder: It contains three sub folders:
   * + 1. images folder: It contains 2 sub folders that contains melanoma images and other images respectively.
       2. new\_images folder: It contains 2 sub folders that contains augmented melanoma images and augmented other images respectively.
       3. new\_datasets folder: It contains train, test and validation dataset respectively.
3. checkpoints folder: It contains the checkpoints of the model.

******

**Fig 5.1**: Tree View of Dataset

**5.2.2 Source Code**

**1. Code to build the model**

if model\_id == 0:

  base\_model = efn.EfficientNetB0(weights='imagenet',

include\_top=False,

input\_shape=(224, 224, 3))

model = base\_model.output

model = GlobalAveragePooling2D()(model)

model = Dense(128, activation='relu')(model)

predictions = Dense(1, activation='sigmoid')(model)

model = Model(inputs=base\_model.input, outputs=predictions)

# Unfreeze the layers

for layer in model.layers[0:]:

   layer.trainable = True

# Set learning rate

if model\_id == 0:

  my\_lr = 0.0001

optimizer=optimizers.Adam(lr=my\_lr)

model.compile(optimizer=optimizer,

loss="binary\_crossentropy",

metrics=['accuracy'])

**2. Code to train the model**

my\_callbacks = [

                earlystop\_callback,

                checkpoint\_callback,

                # lr\_callback

]

h = model.fit(train\_it, epochs = 15,

validation\_data = val\_it,

callbacks = my\_callbacks,

shuffle = True,

steps\_per\_epoch = 100)

**3. Code to upload an image and determine whether it is malignant or not**

# Import necessary libraries

import tensorflow as tf

import numpy as np

import cv2

from google.colab.patches import cv2\_imshow

# Load pre-trained model

model = tf.keras.models.load\_model('/content/model.h5')

# Function to preprocess image

def preprocess\_image(img\_path):

    img = cv2.imread(img\_path)

    img = cv2.resize(img, (224, 224))

    img = img.astype(np.float32) / 255.0

    img = np.expand\_dims(img, axis=0)

    return img

# Upload image

from google.colab import files

uploaded = files.upload()

# Predict melanoma or not

for filename in uploaded.keys():

    img\_path = '/content/' + filename

    img = preprocess\_image(img\_path)

    prediction = model.predict(img)

    if prediction > 0.5:

        print("Melanoma")

    else:

        print("Not melanoma")

    cv2\_imshow(cv2.imread(img\_path))

**5.2.3 Result Analysis**

**Accuracy:** Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right.

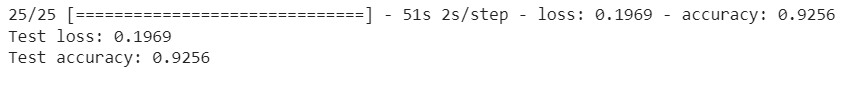
**1. Code to print the accuracy and loss of the model**

# Evaluate the model

results = model.evaluate(test\_it, batch\_size=16)

print("Test loss: {:.4f}".format(results[0]))

print("Test accuracy: {:.4f}".format(results[1]))

******

**Fig 5.2**: Accuracy of proposed system

**2. Code to plot the accuracy and validation accuracy**

# Plotting train accuracy and validation accuracy

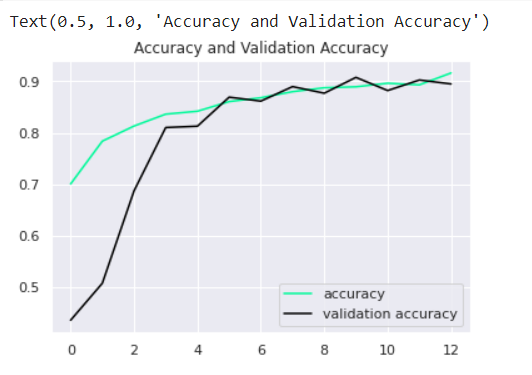
plt.figure()

plt.plot(h.history['accuracy'],color="mediumspringgreen", label='accuracy')

plt.plot(h.history['val\_accuracy'], color="black", label='validation accuracy')

plt.legend(loc='best')

plt.title(label="Accuracy and Validation Accuracy")

****

**Fig 5.3**: Accuracy v/s Validation Accuracy graph

**3. Code to plot the loss and validation loss**

# Plotting train loss and validation loss

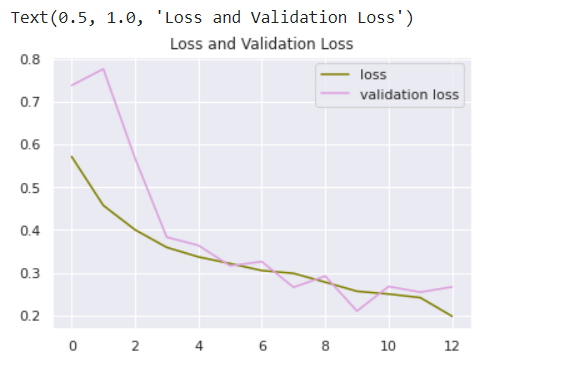
plt.figure()

plt.plot(h.history['loss'],color="olive", label='loss')

plt.plot(h.history['val\_loss'], color="plum", label='validation loss')

plt.legend(loc='best')

plt.title(label="Loss and Validation Loss")

******

**Fig 5.4**: Loss v/s Validation Loss Graph

***CHAPTER 6***

**Testing And Validation**

**6.1 Introduction**

**Software Testing** is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements. **Software Testing** is important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

There are many different types of software tests, each with specific objectives and strategies:

**1. Acceptance testing:**

Acceptance Testing is a method of software testing where a system is tested for acceptability. The major aim of this test is to evaluate the compliance of the system with the business requirements and assess whether it is acceptable for delivery or not.

**Use of Acceptance Testing:**

• To find the defects missed during the functional testing phase.

• How well the product is developed.

• A product is what actually the customers need.

• Feedback helps in improving the product performance and user

experience.

• Minimize or eliminate the issues arising from the production.

**Advantages of Acceptance Testing:**

• This testing helps the project team to know the further requirements from the users directly as it involves the users for testing.

• Automated test execution.

• It brings confidence and satisfaction to the clients as they are directly involved in the testing process.

**2. Unit Testing:**

Unit Testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

**3. Integration Testing:**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**4. Functional Testing:**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

**Valid Input:** identified classes of valid input must be accepted.

**Invalid Input:** identified classes of invalid input must be rejected.

**Functions:** identified functions must be exercised.

**Output:** identified classes of application outputs must be exercised.

**5. System Testing:**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

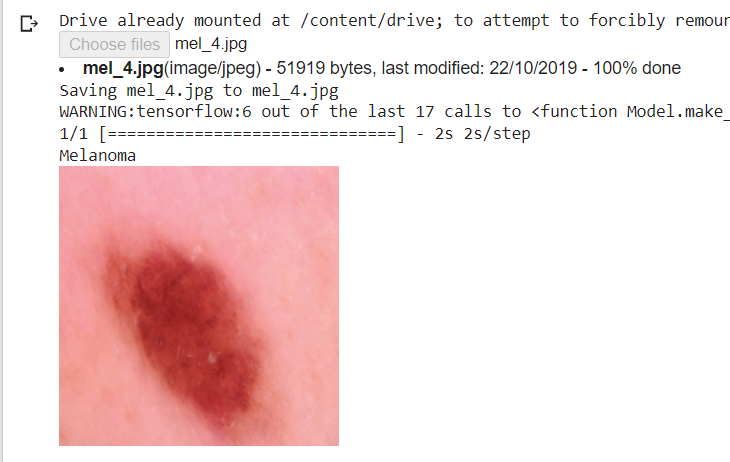
**a. White Box Testing:** It is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**b. Black Box Testing:** It is a testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs.

**The main objectives of software testing**

* To find any defects or bugs that may have been created when the software was being developed
* To increase confidence in the quality of the software
* To prevent defects in the final product
* To ensure the end product meets customer requirements as well as the company specification.
* **6.2Design of test cases and scenarios**

**Test Case 1:** An input image that is malignant is fed to the system.



**Fig 6.1**: When malignant image is given to input

**Test Case 2:** An input image that is benign is fed to the system. 

**Fig 6.2**: When benign image is given to input

**6.3 Validation**

Validation means observing the behavior of the system. The verification and validation mean, that will ensure that the output of a phase is consistent with its input and that the output of the phase is consistent with the overall requirements of the system. This is done to ensure that it is consistent with the required output. If not, apply certain mechanisms for repairing and there by achieved the requirements.

**Advantages of Validation:**

* Any bugs missed during verification will be detected while running validation tests.
* If specifications were incorrect and inadequate from the beginning, validation tests would reveal their inefficacy. Teams will have to spend time and effort fixing them, but it will prevent a bad product from hitting the market.
* Validation tests ensure that the product actually matches and adheres to customer demands, preferences, and expectations under different conditions (slow connectivity, low battery, etc.)
* These tests are also required to ensure that the software functions flawlessly across different browser-device-OS combinations.

Validation is basically done by the testers during the testing. While validating the product if some deviation is found in the actual result from the expected result then a bug is reported or an incident is raised. Not all incidents are bugs. But all bugs are incidents. Incidents can also be of type ‘Question’ where the functionality is not clear to the tester.

Hence, validation helps in unfolding the exact functionality of the features and helps the testers to understand the product in much better way. It helps in making the product more user friendly.

***CHAPTER 7***

**Conclusion and Future Work**

**7.1 Conclusion**

The proposed project is implemented in Google Colab for application development and python language is being used. The application is efficient and also has user friendly interface. It lets the user to know whether the uploaded image is malignant or not. We made use of EfficientNet architecture to build the model that helps in determining the mole is malignant or not. The proposed system gives an accuracy score of 92%.

**7.2 Future Work**

1. In our work we have used EfficientNetB0 model. Other models of EfficientNet can also be used to increase the accuracy.
2. We could further improve the performance of our models with hyperparameter optimization, moreover a richer and more divergent dataset would also provide new opportunities.
3. A web application can be developed to show the results.

***CHAPTER 8***

**References**

1. Medhat, Sara, et al. "Skin cancer diagnosis using convolutional neural networks for smartphone images: A comparative study." *Journal of Radiation Research and Applied Sciences, Elsevier.*  Vol. 15.1 (2022): PP:262-267.
2. Popescu, Dan, et al. "New Trends in Melanoma Detection Using Neural Networks: A Systematic Review." *Sensors, MDPI.* Vol. 22.2 (2022): PP:496.
3. A. Javaid, M. Sadiq, and F. Akram, “Skin cancer classification using image processing and machine learning,” in *Proceedings of the 2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*, pp. 439–444, IEEE, Islamabad, Pakistan, 12-16 Jan 2021.
4. Ashraf, Rehan, et al. "Region-of-interest based transfer learning assisted framework for skin cancer detection." *IEEE Access,*  Vol. 8 (2020): PP:147858-147871.
5. R. Rokhana, W. Herulambang, and R. Indraswari, “Deep convolutional neural network for melanoma image classification,” in *Proceedings of the 2020 International Electronics Symposium (IES)*, pp. 481–486, IEEE, Surabaya, Indonesia, 29-30 Sept 2020.
6. Nida N, Irtaza A, Javed A, Yousaf MH, Mahmood MT. “Melanoma lesion detection and segmentation using deep region based convolutional neural network and fuzzy C-means clustering”. Int J Med Inform. 2019.
7. M. Q. Khan, A. Hussain, S. U. Rehman et al., “Classification of melanoma and nevus in digital images for diagnosis of skin cancer,” *IEEE Access*, vol. 7, pp. 90132–90144, 2019.
8. TAN, Mingxing et LE, Quoc. Efficientnet: Rethinking model scaling for convolutional neural networks. In : International conference on machine learning. PMLR, 2019. p. 6105-6114.
9. P. Tschandl, C. Rosendahl, and H. Kittler, “The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions,” *Scientific Data*, vol. 5, no. 1, pp. 180161–180169, 2018.
10. Bakheet, Samy. "An SVM framework for malignant melanoma detection based on optimized HOG features." *Computation* , MDPI. Vol.5.1 (2017): 4.
11. <https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T>