

# Skin Cancer AI

*A capstone project report submitted in partial fulfillment of the requirement for the award of the degree  
of*

Bachelor of Engineering

in

Electronics and Computers Engineering

Submitted By

Tanmay Garg (Regn. No. 101915001)

Avneet Singh Maingi (Regn. No. 101915005)

Aditya Chawla (Regn. No. 101915018)

Aakarshan Gupta (Regn. No. 101915050)

Akshit Gupta (Regn. No. 101915051)

(Group No – 9)

Under Supervision of

**Dr. Vinay Kumar** (Associate Professor, ECED)

**Dr. Rahul Upadhyay** (Assistant Professor, ECED)



Department of Electronics and Communication Engineering

THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY, PATIALA, PUNJAB

December 2022

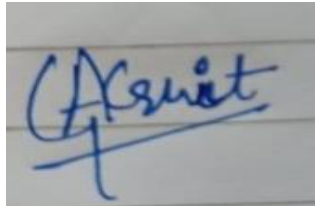
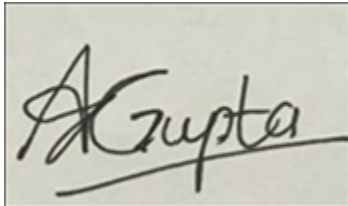
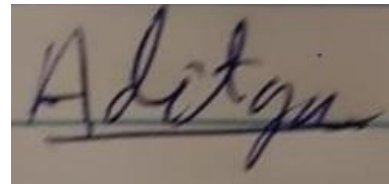
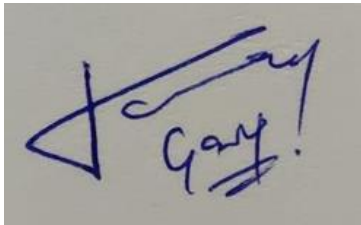
## DECLARATION

---

We hereby declare that the capstone project group report titled “Skin Cancer A.I.” is an authentic record of our work carried out at “Thapar Institute of Engineering and Technology, Patiala” as a Capstone Project in the seventh semester of B.E. (Electronics & Communication Engineering), under the guidance of “**Dr. Vinay Kumar and Dr. Rahul Upadhyay,**” during January to December 2022.

Date: 30<sup>th</sup> November 2022

Signature of Students:



## ACKNOWLEDGEMENT

---

We would like to express our thanks to our mentors, **Dr. Vinay Kumar and Dr. Rahul Upadhyay**. They have been of great help in our venture and an indispensable resource of technical knowledge. They are truly an amazing mentor to have.

We are also thankful to the entire faculty and staff of the Electronics and Communication Department, and also our friends who devoted their valuable time and helped us in all possible ways towards the successful completion of this project. We thank all those who have contributed either directly or indirectly to this project.

Lastly, we would also like to thank our families for their unyielding love and encouragement. They always wanted the best for us, and we admire their determination and sacrifice.

Date: 10<sup>th</sup> August 2022

Roll No.	Name
101915001	Tanmay Garg
101915005	Avneet Singh Maingi
101915018	Aditya Chawla
101915050	Aakarshan Gupta
101915051	Akshit Gupta

## ABSTRACT

---

This project aims at building a device that is used for detecting cancer at earlier stages. The survival rate for early detection of skin cancer is almost 98 percent, but it falls to 62 percent when cancer reaches the lymph node and 18 percent when it metastasizes to distant organs. Skin cancer develops primarily on areas of sun-exposed skin, including the scalp, face, lips, ears, neck, chest, arms, hands, and legs in women.

The building block of this detector is deep learning which is used to process an image, then recognize it and show the output accordingly. The hardware implementation is done using Raspberry Pi Model 3B+, Intel® Movidius™ Neural Compute Stick, SD card, Camera, USB cables, etc. The picture will be received by the camera, and processing will be done inside the Raspberry Pi Model 3B+ & the Intel® Movidius™ Neural Compute Stick will do the heavy lifting, and the output will be displayed on the monitor. The goal is to build a machine-learning algorithm that can detect cancer images and pair them with mentioned hardware hence providing a portable solution for the aforementioned problem.

# TABLE OF CONTENTS

---

<b>DECLARATION.....</b>	<b>i</b>
<b>ACKNOWLEDGEMENT.....</b>	<b>ii</b>
<b>ABSTRACT.....</b>	<b>iii</b>
<b>TABLE OF CONTENTS.....</b>	<b>iv</b>
<b>LIST OF TABLES .....</b>	<b>vii</b>
<b>LIST OF FIGURES .....</b>	<b>viii</b>
<b>LIST OF ABBREVIATIONS .....</b>	<b>ix</b>

<b>1</b>	<b>INTRODUCTION.....</b>	<b>1</b>
1.1	Project Overview.....	1
1.2	Motivation.....	2
1.3	Assumptions and Constraints.....	3
1.4	Novelty Of Work .....	4
<b>2</b>	<b>LITERATURE SURVEY.....</b>	<b>5</b>
2.1	Literature Survey.....	5
2.1.1	Theory Associated with Problem Area .....	6
2.1.2	Sample Table for Literature Survey.....	6
2.1.3	The Problem that has been Identified .....	11
2.2	Research Gaps.....	12
2.3	Problem Definition and Scope .....	13
2.4	Requirements Specification .....	14
2.4.1	Introduction.....	14
2.4.1.1	Purpose.....	14
2.4.1.2	Intended Audience and Reading Suggestions.....	14
2.4.1.3	Project Scope.....	14
2.4.2	Overall Description .....	15
2.4.2.1	Product Perspective.....	15
2.4.2.2	Product Features.....	15
2.4.3	External Interface Requirements.....	15
2.4.3.1	User Interface .....	16
2.4.3.2	Hardware Interfaces .....	16
2.4.3.3	Software Interfaces .....	16
2.4.4	Other Non-Functional Requirements .....	17
2.4.4.1	Performance Requirements .....	17
2.5	Approved Objectives.....	17

2.6	Project Outcomes and Deliverables .....	18
2.7	Cost Analysis .....	18
2.8	Risk Analysis .....	19
<b>3</b>	<b>Flow Chart .....</b>	<b>20</b>
3.1	System Architecture .....	20
3.2	Analysis.....	21
3.3	Tools and Technology.....	22
3.3.1	Hardware.....	22
3.3.2	Machin Learning Models .....	22
<b>4</b>	<b>PROJECT DESIGN AND DESCRIPTIONS .....</b>	<b>23</b>
4.1	Description .....	23
4.2	U.G. Subjects .....	24
4.3	Standards Used.....	24
4.4	Survey of Tools and Technologies.....	25
4.4.1	Raspberry Pi 3B+ .....	25
4.4.1.1	Working Principle .....	26
4.4.1.2	Specification.....	26
4.4.2	Intel Neural Compute Stick 2.....	27
4.4.2.1	Working Principle .....	27
4.4.2.2	Specifications .....	28
4.4.3	TensorFlow .....	29
4.4.3.1	Working .....	29
4.4.3.2	TensorFlow Technical Architecture.....	30
4.4.4	Keras .....	30
4.4.4.1	Description .....	30
4.4.5	OpenVINO .....	31
4.4.5.1	Description .....	31
4.4.5.2	OpenVINO Workflow.....	32
4.4.6	Convolutions Neural Network .....	32
4.4.6.1	Description .....	33
<b>5</b>	<b>IMPLEMENTATION &amp; EXPERIMENTAL RESULT .....</b>	<b>34</b>
5.1	Circuit Simulation and Output .....	34
5.2	Python Code Simulation and Output .....	39
5.2.1	Flow Chart of Code Approach .....	39
5.2.2	Functions used in Code and Explanation.....	40
5.2.2.1	Dataset Description .....	40
5.2.2.2	Pre-processing using ImageDataGenerator.....	41
5.2.2.3	Elements of the Deep Neural Network Model.....	41
5.2.2.4	Training the model .....	44

	5.2.2.5 Learning Curves .....	44
	5.2.2.6 Evaluating Model Performance .....	45
	5.2.2.7 Making predictions .....	47
	5.2.2.8 Saving Model as Frozen Graph.....	47
<b>6</b>	<b>OUTCOME AND PROSPECTIVE LEARNING .....</b>	<b>49</b>
6.1	Scope and Outcomes .....	49
6.2	Prospective Learning.....	50
6.3	Conclusion .....	50
<b>7</b>	<b>PROJECT TIMELINE.....</b>	<b>51</b>
7.1	Work Breakdown and Gantt Chart.....	51
7.2	Project Timeline .....	53
7.3	Individual Gantt Chart .....	54
	<b>REFERENCES.....</b>	<b>55</b>

## LIST OF TABLES

Table No.	Caption	Page No.
Table 1.1	Constraints while building the project	03
Table 1.2	Assumptions while building the project	03
Table 2.1	Survey Table	08
Table 2.2	Cost Analysis	18
Table 2.3	Risk Analysis	19
Table 4.1	Specifications of Neural Compute Stick	28
Table 7.1	Gantt Chart(Jan-June)	52
Table 7.2	Gantt Chart(July-Nov)	52
Table 7.3	Individual Gantt Chart(Jan-June)	54
Table 7.4	Individual Gantt Chart(July-Nov)	54



## LIST OF FIGURES

Figure No.	Caption	Page No.
Figure No. 1.1	Skin Cancer Statistics	01
Figure No. 1.2	Traditional ML vs. Deep Learning	04
Figure No. 1.3	Training on the server and deploying on NCS2	04
Figure No. 2.1	Benign Vs. Malignant	05
Figure No. 3.1	System Architecture	20
Figure No. 4.1	Raspberry Pi 3B+	25
Figure No. 4.2	Neural Compute Stick 2	27
Figure No. 4.3	TensorFlow Working	29
Figure No. 4.4	OpenVINO Toolkit	31
Figure No. 4.5	OpenVINO Workflow	32
Figure No. 4.6	Typical CNN network	33
Figure No. 4.7	Neurons of a Convolution layer, connected to their receptive	33
Figure No. 5.1	Flowchart of conversion of model	36
Figure No. 5.2	Flow Chart of ML model	39
Figure No. 5.3	ISIC skin cancer dataset preview	40
Figure No. 5.4	ImageDataGenerator Augmented Images	41
Figure No. 5.5	MobilenetV2 architecture	42
Figure No. 5.6	MobilenetV2 parameters for the input image	42
Figure No. 5.7	Model Summary	43
Figure No. 5.8	Model Training	44
Figure No. 5.9	Learning Curve	45
Figure No. 5.10	Confusion Matrix	46
Figure No. 5.11	Classification Report	47
Figure No. 5.12	Making Predictions	47

## LIST OF ABBREVIATIONS

<b>CNN</b>	Convolutional Neural Network
<b>DNN</b>	Deep Neural Network
<b>VM</b>	Virtual Machine
<b>GUI</b>	Graphic User Interface
<b>UI</b>	User Interface
<b>MSRP</b>	Manufacturer Suggested Retail Price
<b>ML</b>	Machine Learning
<b>UV</b>	Ultra-Violet
<b>VPU</b>	Vision Processing Unit
<b>GPU</b>	Graphical Processing Unit
<b>CPU</b>	Central Processing Unit
<b>AI</b>	Artificial Intelligence
<b>IEEE</b>	Institute of Electrical and Electronics Engineers
<b>OS</b>	Operating System
<b>SVM</b>	Support Vector Machine
<b>NCS</b>	Neural Compute Stick
<b>HDMI</b>	High-Defenition Multimedia Interface
<b>XML</b>	Extensible Markup Language
<b>API</b>	Application Programming Interface
<b>PoE</b>	Power Over Ethernet

# 1. INTRODUCTION

## 1.1 Project Overview

Skin cancer is seen as one of the most hazardous forms of cancer found in humans. Skin cancer is the abnormal growth of skin cells. It's common cancer that can form on any part of the body, but it often occurs on sun-exposed skin[1]. The survival rate for early detection in the case of skin cancer is almost 98 percent, but it falls to 62 percent when cancer reaches the lymph node and 18 percent when it metastasizes to distant organs. We want to resolve this problem by detecting whether the cancer is malignant or benign in earlier stages. The building block of the Skin Cancer A.I. detector is a deep learning model which is used to process an image, then recognize it and show the output accordingly and is paired with hardware(mentioned below) to reduce time, increase efficiency and result in an optimized output.

TensorFlow and Keras are used along with a Convolutional Neural Network(CNN) architecture called MobileNet v2 to build and create a machine-learning model. Using Convolutional Neural Networks, we developed algorithms and models to distinguish between benign and malignant skin cancers.



Figure 1.1- Skin Cancer Statistics Source: Blue Scan Labs

Hardware Used:

- Raspberry Pi Model 3B+
- Intel® Movidius™ Neural Compute Stick 2
- S.D. card, Camera, USB cables, Mouse, and Keyboard

## 1.2 Motivation

One of the major reasons behind the high fatality rate due to skin cancer is the lack of detection at earlier stages which in turn leads to late diagnosis, and then it becomes almost impossible to cure it. When the cell become cancerous, they can slowly spread to the other parts of the body if left untreated for a long period.

Melanoma, also known as ‘Black Tumor,’ is one of the deadliest skin cancers, which rapidly spreads to other parts of the body and be fatal if not diagnosed early. Early diagnosis is extremely important for effective treatment. The survival rate for early detection in the case of skin cancer is almost 98 percent, but it falls to 62 percent when cancer reaches the lymph node and 18 percent when it metastasizes to distant organs. For the treatment of melanoma, even at earlier stages, various processes are involved, like surgery, radiation, medication, or in some cases, chemotherapy. One of the other major motivations was that more than 1 million cases of melanoma are encountered in India on a yearly basis.

In remote areas, it is hard to diagnose skin cancer due to the inaccessibility of qualified doctors and lack of internet. This is further hammered by the reason that people living in remote areas(like tribes) are most likely to be affected by cancer due to extended time in direct sunlight.

There have been several experiments and studies to demonstrate and prove the fact that A.I. support can improve the overall accuracy of dermatologists in the dichotomous image-based discrimination between melanoma and nevus. This supports the argument for AI-based tools to aid clinicians in skin lesion classification and provides a rationale for studies of such classifiers in real-life settings. A comparatively new strategy falling within the realm of computer-aided diagnosis (CAD) is the use of trained convolutional neural networks (CNNs) to analyze macroscopic images of suspicious lesions. Studies have shown that, within certain limitations and considering a purely image-based setting, artificial intelligence (A.I.) can achieve on-par or superior performance to dermatologists, thereby highlighting its potential as a decision-support system with immediate clinical implications.

We found a great rising in the trends of skin cancer A.I. detection systems, but still, there is space for further improvement in present measures of skin cancer A.I. detection systems.

### 1.3 Assumptions and Constraints

Assumptions and constraints associated with the project are discussed in the following table:

Table 1.1

S.No.	Constraints
1.	We were able to gather around 5000 images for training and testing of our model, which can be improved upon further if more images were readily available.
2.	To build a CNN model with thousands of images, training and testing demand a high computation power. Access to such a computer is one of the constraints.
3.	A piece of specific technical knowledge (the user needs to write a line of code at the terminal) is required to get the result displayed on the output device.
4.	The Skin cancer detection system A.I. software can only detect if a given image is cancerous or not. If the mole is not tumourous in nature, then the built model will not function as expected.
5.	The price and availability of hardware proved to be a major hassle as NCS2's production has declined, and MSRP is relatively high for a student to afford. The same goes for Raspberry Pi.

Table 1.2

S.No.	Assumptions
1	Moles on the skin must be within the range of the camera.
2	The terminal command should be accurate to get the result displayed on the output device.
3	The camera is assumed to be at a proper height.
4	The used hardware must be in working order to deduce the output correctly at the time of execution.

## 1.4 Novelty Of Work

There exist many AI-based detection systems which use Machine Learning techniques as their fundamental base for the working of the project. With the expansion of technology, image-based detection systems are attracting more attention. Machine learning (ML) requires feature extraction and model training. The major difference between traditional ML and deep learning is in feature engineering. Traditional ML uses manually programmed features, whereas deep learning does it automatically. Feature engineering is relatively difficult since it requires domain expertise and is very time-consuming. Deep learning requires no feature engineering and can be more accurate[2]. We aim to make a Skin Cancer Detection System using deep learning techniques to achieve high accuracy and to identify cancerous cells as fast and efficiently as possible.

We are implementing a unique approach to solve the problem, our approach divides the functionality into two main parts, and we merge them synchronously, resulting in a combination of both hardware and software. The first part includes model building and training on the cancer images dataset with corresponding labels. The second includes deploying on edge, using the aforementioned model on our built device are receiving the output.

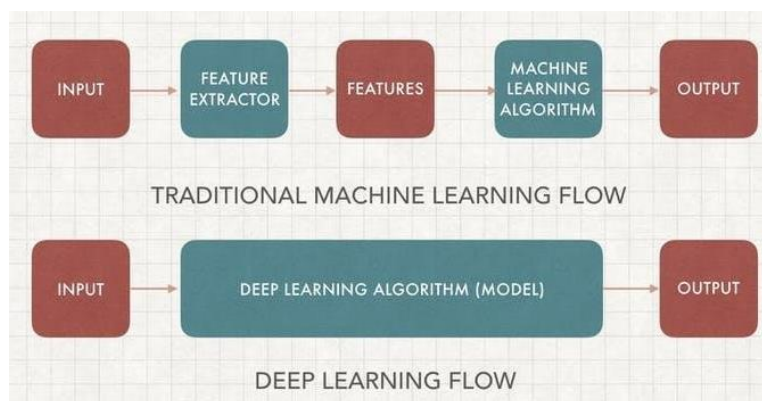


Figure 1.2- Traditional Machine Learning vs. Deep Learning

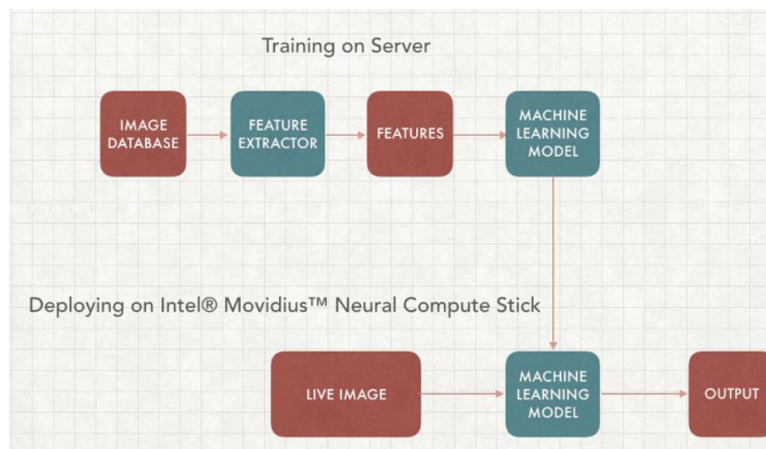


Figure 1.3- Training on the server and deploying on NCS2

## 2. LITERATURE SURVEY

### 2.1 Literature Survey

Skin cancer — the abnormal growth of skin cells — most often develops on skin exposed to the sun. But this common form of cancer can also occur in areas of your skin not ordinarily exposed to sunlight. There are three major types of skin cancer — basal cell carcinoma, squamous cell carcinoma, and melanoma. You can reduce your risk of skin cancer by limiting or avoiding exposure to ultraviolet (U.V.) radiation. Checking your skin for suspicious changes can help detect skin cancer at its earliest stages. Early detection of skin cancer gives you the most excellent chance for successful skin cancer treatment. Skin cancer develops primarily on areas of sun-exposed skin, including the scalp, face, lips, ears, neck, chest, arms, hands, and legs in women. But it can also form on areas that rarely see the light of day — your palms, beneath your fingernails or toenails, and your genital area. Skin cancer affects people of all skin tones, including those with darker complexions. When melanoma occurs in people with dark skin tones, it's more likely to occur in areas not generally exposed to the sun, such as the palms of the hands and soles of the feet. Most skin cancers are preventable[3]. To protect yourself, follow these skin cancer prevention tips:

- Avoid the sun during the middle of the day
- Wear sunscreen year-round
- Wear protective clothing
- Avoid tanning beds
- Be aware of sun-sensitizing medications

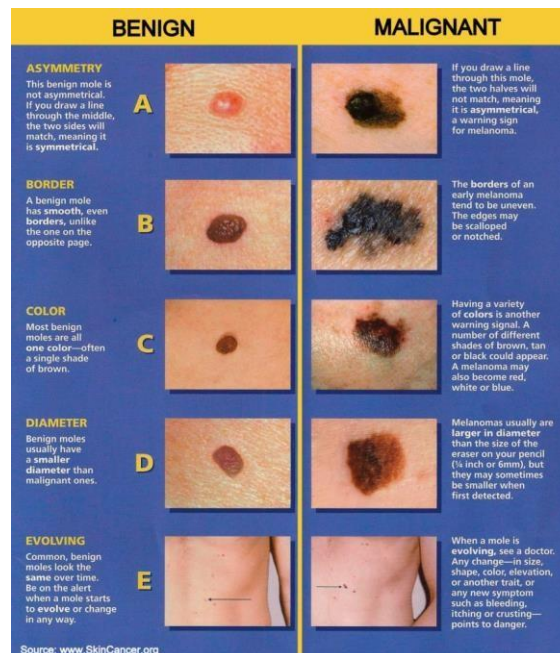


Figure 2.1- Benign Vs. Malignant

### **2.1.1 Theory Associated with Problem Area**

- Skin cancer is seen as one of the most hazardous forms of cancer found in humans. Skin cancer is the abnormal growth of skin cells.
- It's common cancer that can form on any part of the body, but it often occurs on sun-exposed skin.
- According to the Skin Cancer Foundation, half of the population in the United States is diagnosed with some form of skin cancer by age 65. The survival rate for early detection is almost 98%, but it falls to 62% when cancer reaches the lymph node and 18% when it metastasizes to distant organs. With Skin Cancer A.I., we want to use the power of artificial intelligence to provide early detection as widely as possible.

### **2.1.2 Sample Table for Literature Survey**

#### **Survey Introduction to Skin Cancer**

Skin cancer global statistics are equally alarming. Recent reports show that from 2008 to 2018, there has been a 53% increase in new melanoma cases diagnosed annually. The mortality rate of this disease is expected to rise in the next decade. The survival rate is less than 14% if diagnosed in later stages. However, if skin cancer is detected at early stages, then the survival rate is nearly 97%. This demands the early detection of skin cancer.

#### **Models and Algorithms**

The steps for constructing a deep learning model are listed below:

- Select the input block, drag and drop it into the workspace.
- Perform some normalization (if required, drag & drop related blocks)
- Apply Convolution blocks (Choose activation ReLU preferably)
- Apply Pooling Blocks
- Apply Dropout block
- Repeat 3, 4, & 5 as many times as optimally needed
- Use some flattening block
- Use some core layers (Dense blocks preferably). Ensure here that the final core layer before the output block has the proper output value parameter as per the data (output) categories.
- Apply the output block.
- Set the model parameters (Hyper parameter setting).
- Finally, ensure that the Model construction checks Green "OK"; else, fine-tune your selections.



## **Methodology**

The DLS, Model Driven Architecture Tool, provides Neural Network Modeling components as a stack of drag-drop & develop art. The sequence of critical general steps involved as a research methodology in this paper is as follows:

- Data Preparation
- Creating a project and loading the dataset
- Building a deep learning classifier
- Tuning the model
- Checking result
- Drawing Inferences
- Code Access

## **Training**

To train the model, one needs to start the compute instance and push the redpower button. The power button will go green, and the training instance using the remote GPU will create. The dashboard shows the progress while training, accuracy changes, and completion of epochs. Training will stop if there is any error in the run. If everything is correct, training will stop after the stopping criteria.

## Table

Table 2.1

S. No.	Roll No.	Name	Paper Title	Tools/Technology	Findings	Citation
1	101915051	Akshit Gupta	Artificial Intelligence in skin cancer diagnostics: The Patients Perspective	Artificial Intelligence (A.I.)	The use of A.I. can lead to more precise, impartial, and faster diagnostics, particularly in skin cancer diagnostics. People had a positive attitude toward the use of A.I. in the medical field in general and melanoma diagnostics.	Jutzi, T. B., Utikal, J. S., Hauschild, A., Sondermann, W., Fröhling, S., Hekler, A., Schmitt, M., Maron, R. C., & Brinker, T. J. (2020). Artificial Intelligence in Skin Cancer Diagnostics: The Patients' Perspective [4]
2	101915050	Aakarshan Gupta	Machine Learning and Its Application in Skin Cancer	Machine Learning, A.I.	ML can be of significant use in early detection as it is critical for effective treatment and better skin cancer outcomes.	Das, K., Cockerell, C. J., Patil, A., Pietkiewicz, P., Giulini, M., Grabbe, S., & Goldust, M. (2021). Machine Learning and Its Application in Skin Cancer. [5]
3	101915051	Akshit Gupta	Artificial Intelligence for Skin Cancer Detection: Scoping Review	A.I. Based Techniques	A.I. tools are being used, including shallow and deep machine learning-based methodologies that are trained to detect and classify skin cancer using computer algorithms and deep neural networks.	Takiddin, A., Schneider, J., Yang, Y., Abd-Alrazaq, A., & Househ, M. (2021). Artificial Intelligence for Skin Cancer Detection: Scoping Review [6]

4	101915050	Aakarshan Gupta	Comparative Analysis of Skin Cancer (Benign vs. Malignant) Detection Using Convolutional Neural Networks	CNN, deep learning	The convolutional neural network (CNN) helps find skin cancer more accurately through image classification.	Hasan, M. R., Fatemi, M. I., Khan, M. M., Kaur, M., & Zaguia, A. (2021). Comparative Analysis of Skin Cancer (Benign vs. Malignant) Detection Using Convolutional Neural Networks.[7]
5	101915001	Tanmay Garg	Skin Cancer Detection Using Convolutional Neural Network	Convolutional Neural Network (CNN)	The features of the affected skin cells are extracted after the segmentation of the dermoscopic images using the feature extraction technique. A deep learning-based method convolutional neural network classifier is used for the stratification of the extracted features. class	Hasan, Mahamudul & Barman, Surajit & Islam, Samia & Reza, Ahmed Wasif. (2019). Skin Cancer Detection Using Convolutional Neural Network.[5]
6	101915005	Avneet Singh Maingi	Research on Skin Cancer Cell Detection Using Image Processing	Image pre-processing, ML, Deep learning	The skin cancer detection technology is broadly divided into four basic components, viz., image pre-processing, which includes hair removal, de-noise, sharpening, resizing of the given skin image, segmentation which is used for segmenting out the region of interest from the given image.	E. Jana, R. Subban, and S. Saraswathi, "Research on Skin Cancer Cell Detection Using Image Processing," 2017 <i>IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)</i> , 2017 [8]

7	101915001	Tanmay Garg	Classification of skin cancer images using TensorFlow and inception v3	Tensor Flow	Deep convolutional neural network (DCNN), a machine learning classification technique, is used to classify skin cancer images. DCNN algorithm is implemented with Tensor Flow	V, Bhavya & G, Narasimha & M, Ramya & Y, Sujana & AnuRadha, T.. (2018). Classification of skin cancer images using TensorFlow and inception v3.[9]
8	101915005	Avneet Singh Maingi	Skin Cancer Classification from Dermoscopic Images using Feature Extraction Methods	Feature Extraction methods	Various feature descriptors like local binary pattern (LBP), complete LBP (CLBP), and their variants, which are based on histogram mapping, such as uniform, rotation invariant, and rotation consistent uniform patterns.	A. Gautam and B. Raman, "Skin Cancer Classification from Dermoscopic Images using Feature Extraction Methods," <i>2020 IEEE REGION 10 CONFERENCE (TENCON)</i> [10]
9	101915018	Aditya Chawla	Cure skin - Skin Disease Prediction using MobileNet Model	MobileNet, CNN	MobileNet model is pre-trained by feeding thousands of images, including images that are not diseased. This approach is simple, fast, and inexpensive and does not require massive equipment for the diagnosis. It is found that the MobileNet model gives the best accuracy.	Divya, N. & Dsouza, Deepthi & Hariprasad (2021). Cure skin - Skin Disease Prediction using MobileNet Model. [11]
10	101915018	Aditya Chawla	MobileNet Neural Network skin disease detector with Raspberry pi	MobileNet, Raspberry Pi, Keras	The skin disease detector employs the MobileNet convolutional neural network on Raspberry pi for the classification of skin lesions utilizing the Keras architecture for training.	Z. N. Fikile Gasa, P. A. Owolawi, T. Mapayi, and K. Odeyemi, "MobileNet Neural Network skin disease detector with Raspberry pi," <i>2020</i> [12]

### **2.1.3 The problem that has been Identified**

- Usage Limitation- There may be many skin cancer detection systems, but most fail to classify the type of skin cancer further, limiting its usage.
- Training barrier- The training of A.I. systems presents an even more significant obstacle. Hundreds of thousands of photos confirmed as benign or malignant are used to teach the technology to recognize skin cancer, but all of these images were captured in optimal conditions.
- Technology barrier- Unlike A.I. technology, board-certified dermatologists don't just look at one mole to determine whether it's problematic. They consider several additional factors, including the other spots on the patient's body and the evolution of the lesion in question, as well as the individual's skin type, skin cancer history and risk factors, and sun protection habits.

## 2.2 Research Gaps

Although we were able to implement the CNN model and configure the hardware required for the proper working and functioning of our project, we still believe there are some areas on which we can improve further.

- The camera used in the implementation of the project, i.e., the Raspberry Pi 5MP camera, was the only hardware available that was also affordable for this project. Using a high-resolution camera of around ₹ 3000 will result in better images of moles and skin lesions compared to the images in the growing database.
- There's a huge problem in getting AI data for medicine, but amazing results are possible. The more people share, the more accurate the system becomes. Having a limited data set is the only problem that is present while training the model. If the data set is easily and widely available, the accuracy can improve even further. Due to this, the model at a time may even fail to predict the correct result.
- Availability of camera at proper height and place. Placing the camera every time in the accurate position to get the perfect image is itself a task. There is always a chance of human error in such cases, thereby affecting the desired output.
- The unavailability of live test subjects with skin tumors prevents us from using and testing projects in real life. So accuracy can only be checked on pictures available online, which are most of the time unlabelled, and make for an unsuitable test case.
- Due to the price and supply of compute sticks like Nvidia Jetson, we couldn't acquire it. So only one image can be processed at a time which in itself takes around 5 seconds, as Intel Neural Compute Stick 2 has less processing power than Nvidia Jetson. Also, Nvidia Jetson has better long-term support making it a better option for future improvements.
- A better microprocessor like Raspberry Pi 4 decreases computation time and makes testing easier. Even simpler processes like installing O.S. and booting up the device are slower in the development board used. Also, Raspberry Pi 3B+ is a discontinued device, and constraints of running the above-mentioned compute stick required us to install an older version of the O.S., which makes matters harder to resolve.
- Lack of experience in deep learning algorithms limited us to using better algorithms that can increase the accuracy of our project. So we were compelled to use models of which the team members had prior knowledge.

## 2.3 Problem Definition and Scope

Skin cancer is seen as one of the most hazardous forms of cancer found in humans. It's common cancer that can form on any part of the body, but it often occurs on sun-exposed skin. Existing solutions such as image pre-processing, which includes hair removal, de-noise, sharpening, and resizing of the given skin image, and segmentation- which is used for segmenting out the region of interest from the given image. Some commonly used algorithms are k-means, the threshold in the histogram, etc., features extraction from the image, and classification of the image from the features set extracted from the segmented image. A variety of classification algorithms are used for this purpose. The recent skin cancer detection technology uses machine learning and deep learning-based algorithms for the classification of its type and whether a tumor is cancerous or not. The most commonly used classification algorithms are support vector machine (SVM), feed-forward artificial neural network, and deep convolutional neural network.

These methods exist, but we need something that can detect cancer at a very early stage so that it is easily curable. Therefore, it is necessary to establish a system such as a skin cancer A.I. detection system. The base of the Skin Cancer A.I. detector is a deep learning model which is used to process an image, then recognize it and show the output accordingly[19]. TensorFlow and Keras are used along with a Convolutional Neural Network(CNN) architecture called MobileNet v2 to build and create a machine-learning model. Using CNNs, we developed algorithms and models to distinguish between benign and malignant types of skin cancers. On detection of a mole to whether it is cancerous or not.

We can even further identify which type of cancers are there, which represents the future scope of the project, which are as follows nevus, melanoma, and seborrheic keratosis. If we can detect the type of cancer, it will be a great added advantage as patients can be made aware at a very early stage. The addition of GUI will make the application easier to access and use for lesser tech-savvy customers, hence resulting in a larger customer base. The development of a mobile application can make the project affordable and even portable, as everyone has a phone. In the future, the project can be expanded to classify not only skin cancer but also other skin-related diseases like warts, atopic dermatitis, sunburn, etc.

## **2.4 Requirements Specification**

The objective of this report is to provide a full description of the hardware and software components that will be utilized in our Skin Cancer Detection as well as how they should work.

### **2.4.1 Introduction**

This section documents the specific requirements (functional and non-functional), performance requirements, design constraints, and external interface requirements.

#### **2.4.1.1 Purpose**

The objective of this report is to provide a full description of how we are doing our project, the features, and requirements that will be utilized in our 'Skin Cancer A.I. Detection System,' as well as how it should work. It describes the system's goal and characteristics, as well as what it will perform, the limitations it must work under, and how it will react to external stimuli. This paper is intended for both stakeholders and developers of the system.

#### **2.4.1.2 Intended Audience and Reading Suggestions**

This project is a prototype of a Skin Cancer A.I. detection system using image processing, and it is targeting all those who develop moles and want to be sure at a very early stage whether the grown moles are cancerous or not so that they can be sure about what treatment they need before consulting a specialist. This has been implemented under the supervision and guidance of our mentors.

#### **2.4.1.3 Project Scope**

The building block of the Skin Cancer A.I. detector is deep learning which is used to process an image, then recognize it and show the output accordingly. Using Convolutional Neural Networks, we developed algorithms and models to distinguish between benign and malignant skin cancers. On detection of a mole as cancerous, we can even further identify which type of cancer is there, which are as follows: nevus, melanoma, and seborrheic keratosis. If we can detect the type of cancer, it will be a great added advantage as patients can be made aware at a very early stage.



## **2.4.2 Overall Description**

The description of the project is mentioned below under different sections. The section mainly explains the Product perspective and Product features.

### **2.4.2.1 Product Perspective**

We aim to build a Skin Cancer A.I. detection system using image processing to get greater accuracy than the ones already existing. The system will notify the supervisor/user by sending him/her a notification on the display screen. The User Interface is worked by writing a line of code that should be accurate to get the output.

The building block of this detector is deep learning which is used to process an image, then recognize it and show the output accordingly. The hardware implementation is done using Raspberry Pi Model 3B+, Intel® Movidius™ Neural Compute Stick, SD card, Camera, USB cables, etc. The picture will be received by the camera, and processing will be done inside the Raspberry Pi Model 3B+ & the Intel® Movidius™ Neural Compute Stick will do the heavy lifting, and the output will be displayed on the monitor[18]. The goal is to build a machine-learning algorithm that can detect cancer images and pair them with mentioned hardware.

### **2.4.2.2 Product Features**

- Power and cost-efficient Raspberry Pi is used.
- Cancer can be detected in earlier stages hence resulting in averting possible death.
- Low-power NCS 2 makes computation faster and has long-lasting internal components.
- No connectivity to the internet is required to run the project to make it more portable and accessible.
- The end product's small size also helps in making it more portable.
- Easy command to run the files.
- A camera for real-time detection is available, giving it more functionality.

## **2.4.3 External Interface Requirements**

The section explains the interface requirements under the project. Some light is thrown on the user, hardware, and software interfaces.

#### **2.4.3.1 User Interface**

- The skin cancer A.I. detection system is simple to understand and use, making it user-friendly.
- The User Interface is worked by writing a line of code that should be accurate to get the output.
- The user can see the result on the output screen.

#### **2.4.3.2 Hardware Interfaces**

- The hardware part of our system consists of Raspberry-pi, a camera module, Intel® Movidius™ Neural Compute Stick, and a keyboard.
- The user can interact with the hardware directly as they have to write some line of code before viewing the final result on the output display.

#### **2.4.3.3 Software Interfaces**

- Google Colab is used to provide an integrated environment and is a cloud-based software. Colab allows anybody to write and execute arbitrary python code through the browser and is especially well suited to machine learning, data analysis, and education[13].
- ‘Classification\_sample.py’ pre-built API used to help in the classification of images given model’s and image’s file location.
- TensorFlow is an open-sourced end-to-end platform, a library for multiple machine learning tasks, while Keras is a high-level neural network library that runs on top of TensorFlow.

## **2.4.4 Other Non-Functional Requirements**

Non-functional requirements are important as all the requirements are considered below in this section. Basic Non - functional requirements are:

- The performance of the model should be good enough to recognize skin cancer images.
- The response time of the model should be fast
- The model should be robust to any ambiguities
- The model should be reliable in terms of power failure etc.
- The device should be portable and small in size.
- Reliability on the internet should be minimal or absent (if possible).

### **2.4.4.1 Performance Requirements**

The performance of the system highly depends on the Skin cancer A.I. detection algorithm. It also depends on the efficiency of the devices used in the system.

- The resolution of the camera must be high, and it should be at an appropriate distance to capture clear pictures of the skin moles data sample.
- We need to ensure that every device, i.e., Raspberry Pi and Neural Compute Stick 2, is in perfect condition and helps in accurate & fast detection.
- Enough power must be given to the device to prevent power throttling which will highly impact performance.
- The device must be operated in a cool and dry area for a prolonged period to prevent thermal throttling, which will result in damage to its components.

## **2.5 Approved Objectives**

- To study existing skin cancer detection devices and techniques.
- To develop a skin cancer A.I. detection system with good accuracy using image analysis.
- Testing and validating results by comparing with the existing data set.
- Implementation of the model on Raspberry Pi using NCS.
- Optimized the accuracy with the help of the proposed ML model.

## 2.6 Project Outcomes and Deliverables

The proposed product is expected to be able to accomplish the following task after completion:

- Our project will detect the type of moles, whether it is malignant (cancerous) or benign (non-cancerous), using raspberry Pi and Neural Compute Stick 2.
- The fabricated device would be a one-time purchase.
- Saving a life by detecting even the smallest of cancerous moles (early detection survival rate of skin cancer is 98 percent).
- The project is practically applicable.
- Detection of skin cancer with the help of a camera.
- The project can be extended by upgrading the model to describe which types of cancer it is and how they can be treated.
- An eco-friendly method of implementation.

## 2.7 Cost Analysis

For the product to work properly, the hardware needs to be good and reliable. A proper camera is necessary to capture the smallest of moles, whose results will be shown on screen and handled by the supervisor. Hardware cost is explained in the table mentioned below :

Table 2.1

Hardware Required	Cost (in ₹)
Raspberry Pi Model 3B+	3500
Raspberry Pi 5MP Camera	380
Intel® Movidius™ Neural Compute Stick 2	18,000
Keyboard + Mouse	500
Cables(HDMI, power adapter and cables)	500
<b>Total Cost</b>	<b>22,880</b>

## 2.8 Risk Analysis

There are always some chances of risk while designing a new software and hardware system. Most of them can be easily mitigated. The below table gives a detailed analysis of risks that can be faced.

Table 2.2

Major modules of failure	Risk involved	Risk mitigation
Image capture failure (Hardware error)	Inability to capture the image clearly(far distant images)	Use a high-resolution camera.
Image processing failure (Software error)	Inability to process image correctly as per trained model(Overfitting / Underfitting)	Use a different, better, suitable model.
Low contrast failure (Software error)	Failure to capture images based on contrast	Image enhancement during pre-processing.
Loose connection (Hardware error)	Inability to run the device	Check all the connections to ensure reliability.
NCS failure (Hardware error)	NCS fails to run or give high enough accuracy	Buy a better Scompute Stick like Nvidia Jetson.
Raspberry Pi Failure (Hardware error)	Raspberry Pi used has low computation power and is outdated.	Acquire a better version of Raspberry Pi or any other high-end development board.
Compatibility Failure (Software error)	Check the compatibility between devices and software.	Read the documentation thoroughly and install the correct version.

### 3. FLOW CHART

#### 3.1 System Architecture

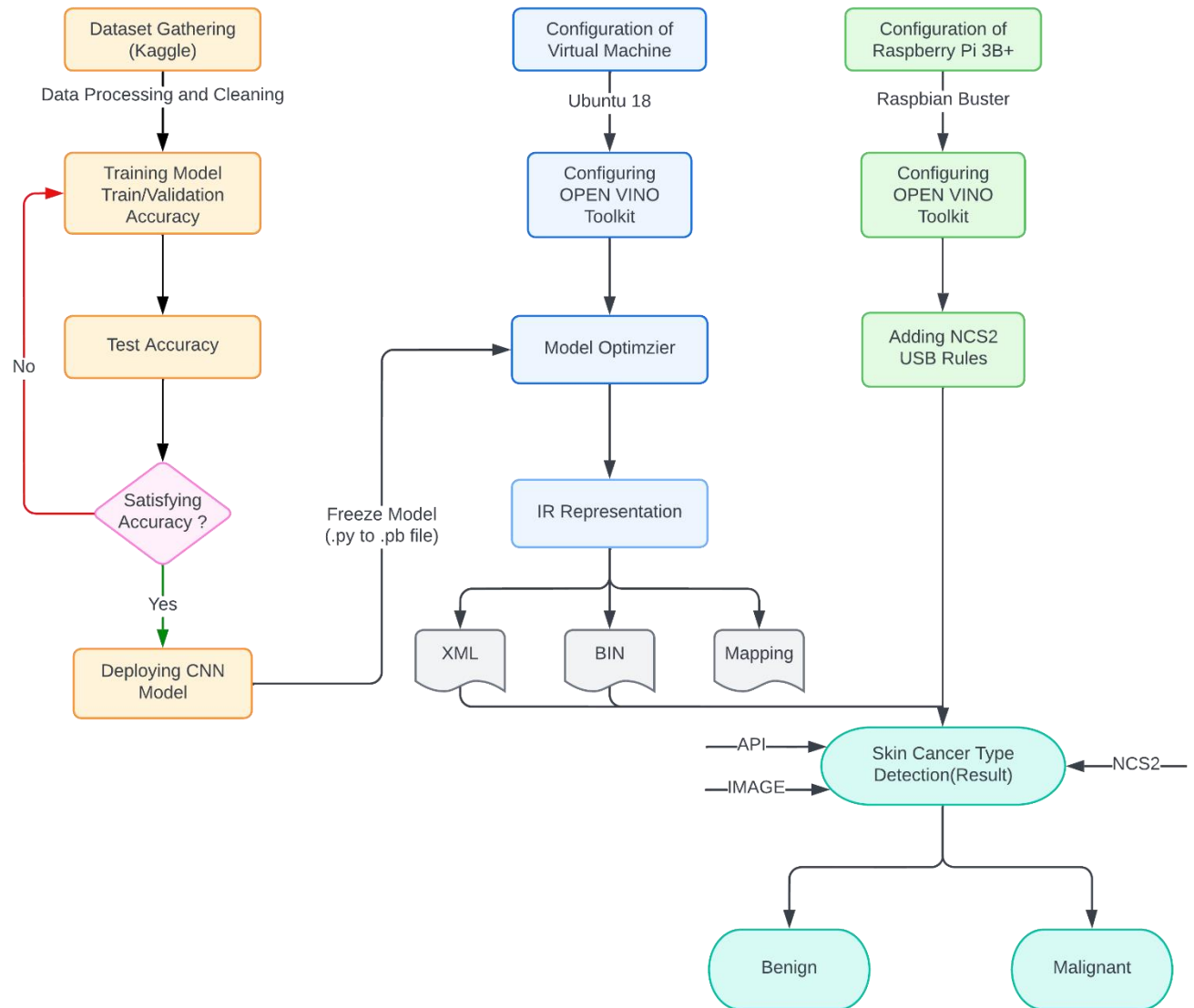


Figure 3.1- System Architecture

## 3.2 Analysis

The flow of the project majorly consists of 3 parts:

### **Model:**

1. The dataset is collected, and features are extracted.
2. The images and pre-built model(MobileNet v2) are used to train the model(train and validation accuracy).
3. Testing accuracy is calculated. If the numbers are unsatisfying, parameter tuning is performed, and the model has trained again. Else final model is deployed.
4. The final model is frozen and fed to the model optimizer.

### **Virtual Machine :**

1. Ubuntu 18(LTS) is installed and configured.
2. Open Vino toolkit is set up in V.M. Libraries are installed, and the environment is initialized.
3. NCS2 USB rules are added, and the hypervisor is configured.
4. The frozen model is fed into the model optimizer giving an Intermediate Representation as output. I.R. consists of XML, bin, and mapping files.
5. Test model using images and API. If the results are unsatisfying model is tuned and converted again.
6. XML and bin, along with API(classification\_sample.py) and photos, are then transferred to Raspberry Pi.

### **Raspberry Pi 3B+ :**

1. Raspbian buster is installed and configured.
2. Open Vino toolkit is set up in Raspberry Pi, and it is a lighter version than installed in Virtual Machine. Libraries are installed, and the environment is initialized. USB rules for Neural Compute Stick 2 are added.
3. An API(classification\_sample.py) with images and XML file is used in coordination with NCS2 to classify and detect.
4. Results are presented, in terms of probability, as Benign or Malignant.

### **3.3 Tools and Technologies Used**

The project is divided into the following modules-

#### **3.3.1 Hardware**

- **Raspberry Pi 3B+:** The Raspberry Pi is a low-cost, credit-card-sized computer that plugs into a computer monitor or T.V., can be powered by a phone charger, and uses a standard keyboard and mouse.
- **Camera Module:** It is used to capture clear and accurate pictures of the given skin sample. The module is compatible with Raspberry Pi.
- **Neural Compute Stick:** Intel Neural Compute Stick 2 is powered by the Intel Movidius X VPU to deliver industry-leading performance, wattage, and power. The neural compute supports [14] OpenVINO, a toolkit that accelerates solution development and streamlines deployment.
- **Keyboard and Mouse:** To operate Raspberry Pi and connect using USB.

#### **3.3.2 Machine Learning Models**

- **Image pre-processing:** CNN
- **Feature extraction and image pre-process:** Mobile Net
- **Tools and Technology:** Python, TensorFlow, Keras, Open VINO
- **Model evaluation and visualization:** Weights and Biases (WandB)
- **Dataset- Skin Cancer:** Malignant vs. Benign (Kaggle)



## 4. PROJECT DESIGN AND DESCRIPTION

### 4.1 Description

In recent days, skin cancer has been seen as one of the most Hazardous forms of Cancers found in humans. Skin cancer is found in various types, such as Melanoma, Basal, and Squamous cell Carcinoma, among which melanoma is the most unpredictable[15]. The detection of Melanoma cancer in the early stage can be helpful in curing it. Computer vision can play an important role in Medical Image Diagnosis, and it has been proved by many existing systems. TensorFlow and Keras are used to build and create a machine-learning model. Using Convolutional Neural Networks, we developed algorithms and models to distinguish between benign and malignant skin cancers.

Melanoma, also known as 'Black Tumor,' is one of the deadliest skin cancers, which rapidly spreads to other parts of the body and be fatal if not diagnosed early. Early diagnosis is extremely important for effective treatment. The survival rate for early detection in the case of skin cancer is almost 98 percent, but it falls to 62 percent when cancer reaches the lymph node and 18 percent when it metastasizes to distant organs. For the treatment of melanoma, even at earlier stages, various processes are involved, like surgery, radiation, medication, or in some cases, chemotherapy. One of the other major motivations was that more than 1 million cases of melanoma are encountered in India on a yearly basis.

These methods exist, but we need something that can detect cancer at a very early stage so that it is easily curable. Therefore, it is necessary to establish a system such as a skin cancer A.I. detection system. The base of the Skin Cancer A.I. detector is a deep learning model which is used to process an image, then recognize it and show the output accordingly. TensorFlow and Keras are used along with a Convolutional Neural Network(CNN) architecture called MobileNet v2 to build and create a machine-learning model. Using CNNs, we developed algorithms and models to distinguish between benign and malignant types of skin cancers. On detection of a mole to whether it is cancerous or not.

## 4.2 U.G. Subjects

- **Machine Learning:** Machine learning is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead[16].
- **UEC513: Embedded Systems-** An embedded system is a computer system—a combination of a computer processor, computer memory, and input/output peripheral devices—that has a dedicated function within a larger mechanical or electronic system[17]. Embedded systems tell us about Wires and Ports, Basic Protocols of data transfer, Bus arbitration, ISA bus signals, handshaking, Memory-mapped I/O, and simple I/O, Parallel I/O, and Port-Based I/O.
- **UEC610: Computer Architecture-** Computer Architecture helps us to Evaluate the performance of a RISC-based machine with an enhancement applied and make a decision about the applicability of that respective enhancement as a design engineer (performance metrics). Display an understanding of the concept of pipelining and parallelism pipelining in a modern RISC processor and describe how hazards are resolved.
- **UEC620: Deep Learning for Computer Vision-** The automatic analysis and understanding of images and videos, a field called Computer Vision, occupies significant importance in applications including security, healthcare, entertainment, mobility, etc. [19]. The recent success of deep learning methods has revolutionized the field of computer vision, making new developments increasingly closer to deployment that benefits end users.

## 4.3 Standards Used

- **IEEE 29148** is software requirements specification in a software system to be developed. Specifies the required processes implemented in the engineering activities that result in requirements for systems and software products (including services) throughout the life cycle[19].
- **IEEE 1012** is a standard for software project management, software testing, software engineering, verification, and validation.
- **IEEE 16326** is the standard for Software project management which is the art and science of planning and leading software projects. It provides detailed discussion and advice on applying a set of technical management processes that are common to both the system and software life cycles.
- **IEEE 24748** is standard for Software documentation which is written text or illustration that accompanies computer software or is embedded in the source code. It also addresses systems concepts and life cycle concepts, models, stages, processes, process application, key points of view, adaptation, and use in various domains and by various disciplines[20].

## 4.4 Survey of Tools and Technologies

The project is divided into several modules listed below

### 4.4.1 Raspberry Pi 3B+

The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins, allowing you to control electronic components for physical computing and explore the Internet of Things (IoT).

There have been many generations of the Raspberry Pi line: from Pi 1 to 4, and even a Pi 400. There has generally been a Model A and a Model B of most generations. Model A has been a less expensive variant and tends to have reduced RAM and fewer ports (such as USB and Ethernet)[21]. We are using Raspberry Pi 3B+ for our project, which uses a standard keyboard and mouse. 1.4GHz 64-bit quad-core processor, dual-band wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and Power-over-Ethernet support (with separate PoE HAT).

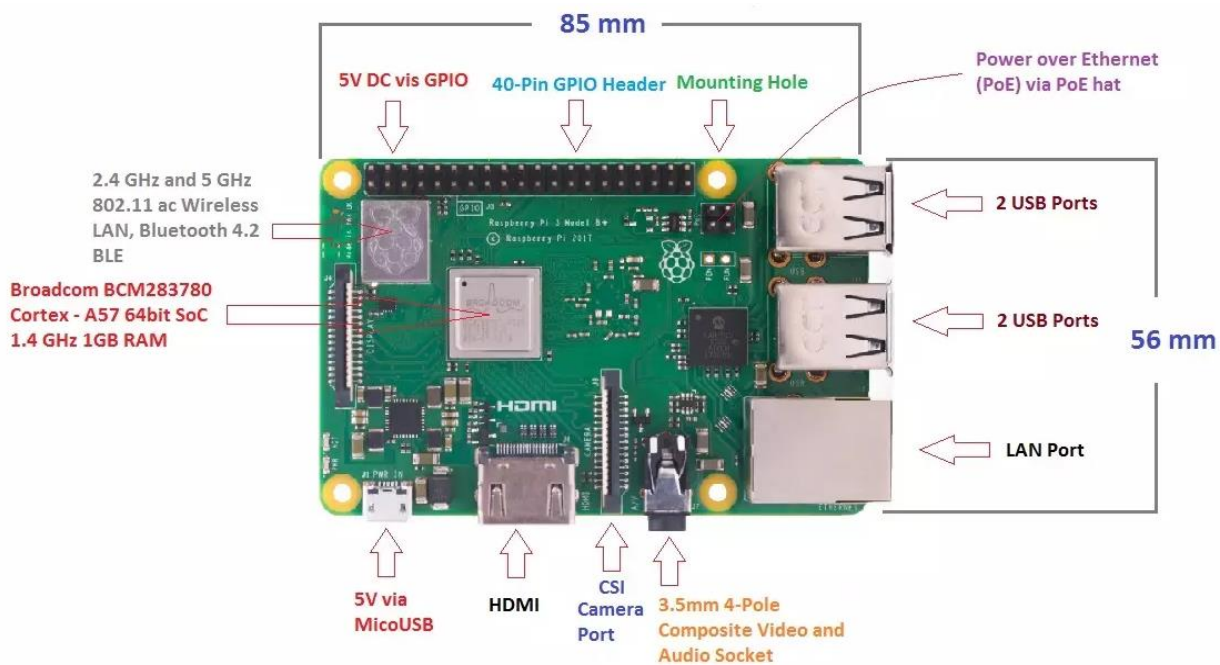


Figure 4.1- Raspberry Pi 3B+

#### 4.4.1.1 Working Principle

- It is a tiny computer board that comes with CPU, GPU, USB ports, I/O pins, Wi-Fi, Bluetooth, USB, and network boot, and it is capable of doing some functions like a regular computer.
- The SoC (system on chip) combines both CPU and GPU on a single package and turns out to be faster than the Pi 2 and Pi 3 models.
- The dual-band Wi-Fi 802.11ac runs at 2.4GHz and 5GHz providing a better range in challenging wireless environments, and Bluetooth 4.2 is available with BLE support.
- Four pin header is added on the board that resides near 40-pin headers. This allows the Power over Ethernet (PoE), i.e., provides the necessary electrical current to the device using data cables instead of power cords. It is very useful and reduces the number of cables required for the installation of a device in the relevant project[22].
- PoE works only in the presence of a PoE hat.

#### 4.4.1.2 Specification

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- Extended 40-pin GPIO header
- Full-size HDMI
- USB 2.0 ports
- CSI camera port for connecting a Raspberry Pi camera
- 4-pole stereo output and composite video port
- Micro SD port for loading your operating system and storing data
- 5V/2.5A D.C. power input
- Power-over-Ethernet (PoE) support (requires separate PoE HAT)
- DSI display port for connecting a Raspberry Pi touchscreen display

### 4.4.2 Intel Neural Compute Stick 2

It is a tiny, fanless, deep-learning device that allows you to learn A.I. programming at the edge. It is powered by the same high-performance Intel Movidius Vision Processing Unit (VPU) that can be found in millions of smart security cameras, gesture-controlled drones, industrial machine vision equipment, and more.



Figure 4.2- Movidius Neural Compute Stick

#### 4.4.2.1 Working Principle

The Movidius™ Neural Compute Stick (NCS) is a tiny fanless deep learning device that you can use to learn A.I. programming at the edge. NCS is powered by the same low-power, high-performance Movidius™ Vision Processing Unit (VPU) that can be found in millions of smart security cameras, gesture-controlled drones, industrial machine vision equipment, and more. The Movidius Neural Compute Stick enables rapid prototyping, validation, and deployment of Deep Neural Network (DNN) inference applications at the edge. Its low-power VPU architecture enables an entirely new segment of A.I. applications that aren't reliant on a connection to the cloud. The NCS combined with Movidius™ Neural Compute SDK allows deep learning developers to profile, tune, and deploy Convolutional Neural Networks (CNN) on low-power applications that require real-time inferencing.

#### 4.4.2.2 Specifications

Table 4.1

Brand	Intel
Form Factor	Stick
Item Height	2.85 Inches
Item Width	0.55 Inches
Standing screen display size	9 Inches
Product Dimensions	2.69 x 1.4 x 7.24 cm; 18.14 Grams
Item model number	NCSM2485.DK
Processor Brand	Intel
Processor Count	1
Hard Drive Interface	USB
Graphics Coprocessor	Integrated Graphics
Graphics Chipset Brand	Intel
Are Batteries Included	No
Item Weight	18.1 g

### 4.4.3 TensorFlow

TensorFlow is an open-source end-to-end platform for creating Machine Learning applications. It is a symbolic math library that uses dataflow and differentiable programming to perform various tasks focused on training and inference of deep neural networks[23]. It allows developers to create machine-learning applications using various tools, libraries, and community resources. Currently, the most famous deeplearning library in the world is Google's TensorFlow.

#### 4.4.3.1 Working

TensorFlow allows developers to create dataflow graphs—structures that describe how data moves through a graph or a series of processing nodes[23]. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multi-dimensional data array or tensor. TensorFlow applications can be run on almost any target that's convenient: a local machine, a cluster in the cloud, iOS and Android devices, CPUs, or GPUs.

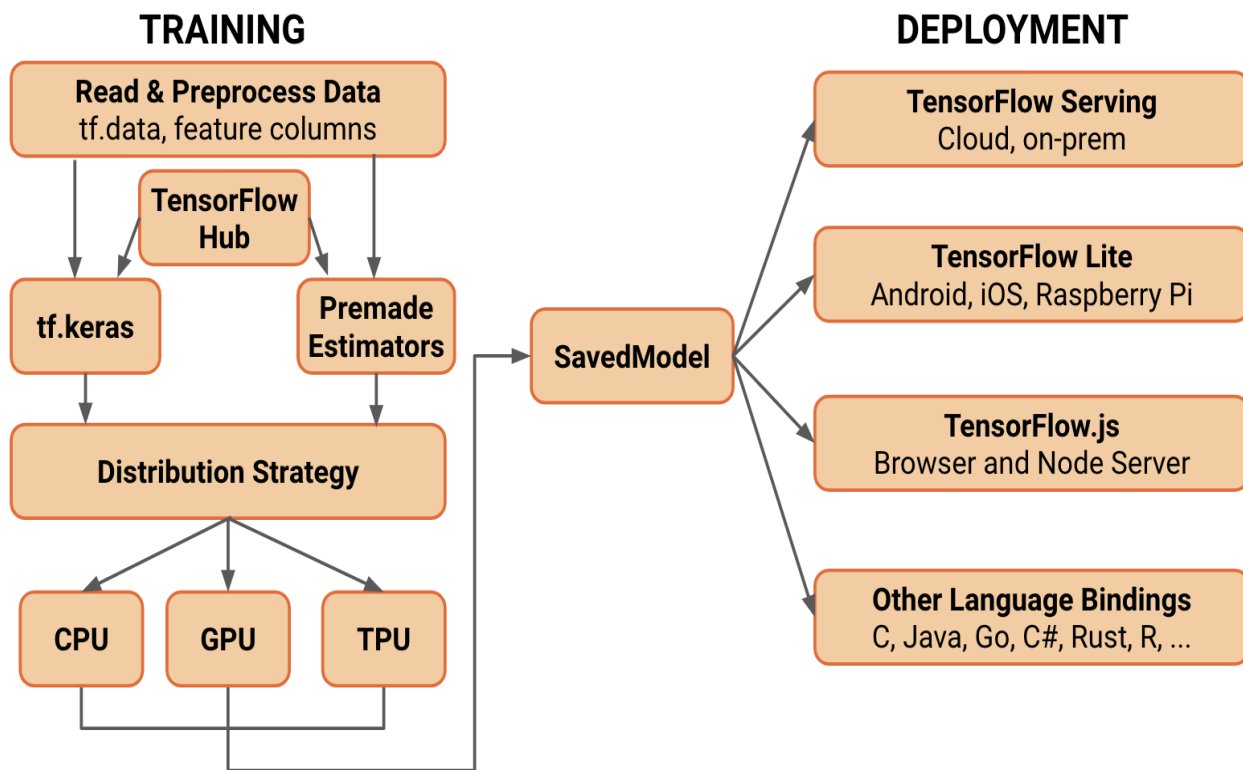


Figure 4.3- TensorFlow Working

#### 4.4.3.2 TensorFlow Technical Architecture

- Sources create loaders for Servable Versions, and then loaders are sent as Aspired versions to the Manager, which will load and serve them to client requests.
- The Loader contains metadata, and it needs to load the servable.
- The source uses a callback to convey the Manager of Aspired version.
- The Manager applies the effective version policy to determine the next action to take.
- If the Manager determines that it gives the Loader to load a new version, clients ask the Manager for the servable and specify a version explicitly or request the current version. The Manager returns a handle for servable. The dynamic Manager applies the version action and decides to load the newer version of it.
- The dynamic Manager commands the Loader that there is enough memory.
- A client requests a handle for the latest version of the model, and the dynamic Manager returns a handle to the new version of the servable[24].

#### 4.4.4 Keras

Keras is a deep learning API written in Python, running on top of the machinelearning platform TensorFlow. It was developed with a focus on enabling fast experimentation.

Keras is:

- Simple -- but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
- Flexible -- Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.
- Powerful -- Keras provides industry-strength performance and scalability: it is used by organizations and companies, including NASA, YouTube, and Waymo[25].

##### 4.4.4.1 Description

Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page and a Slack channel.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.



## 4.4.5 OpenVINO

Open VINO™ toolkit is a comprehensive toolkit for quickly developing applications and solutions that solve a variety of tasks, including emulation of human vision, automatic speech recognition, natural language processing, recommendation systems, and many others. Based on the latest generations of artificial neural networks, including Convolutional Neural Networks (CNNs) and recurrent and attention-based networks, the toolkit extends computer vision and non-vision workloads across Intel® hardware, maximizing performance. It accelerates applications with high-performance, A.I., and deep learning inference deployed from edge to cloud[26].

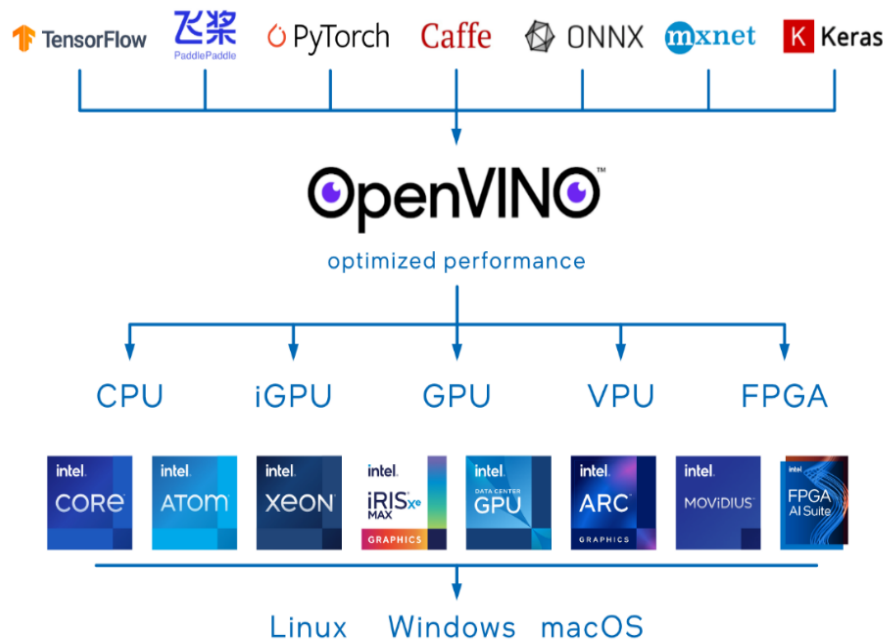


Figure 4.4- OpenVINO Toolkit

### 4.4.5.1 Description

**Easy Deployment of Model Server at Scale-** A complete Open VINO ModelServer deployment in Kubernetes and OpenShift can be managed via ModelServer resources.

**Support for Multiple Model Storage Options-** Model repositories can be stored in S3-compatible cloud storage like Amazon S3, Azure Blob, and Google Cloud Storage (GSC). It is possible to deploy models hosted in local storage on the Kubernetes Nodes or in Kubernetes Persistent Volumes.

**Configurable Resource Restrictions and Security Context-** It is possible to define OpenShift resource requirements in the model server pods. That also includes the usage of A.I. accelerators like iGPU or VPU. Security context can be tuned to adjust permissions in the model repository.

#### 4.4.5.2 OpenVINO Workflow

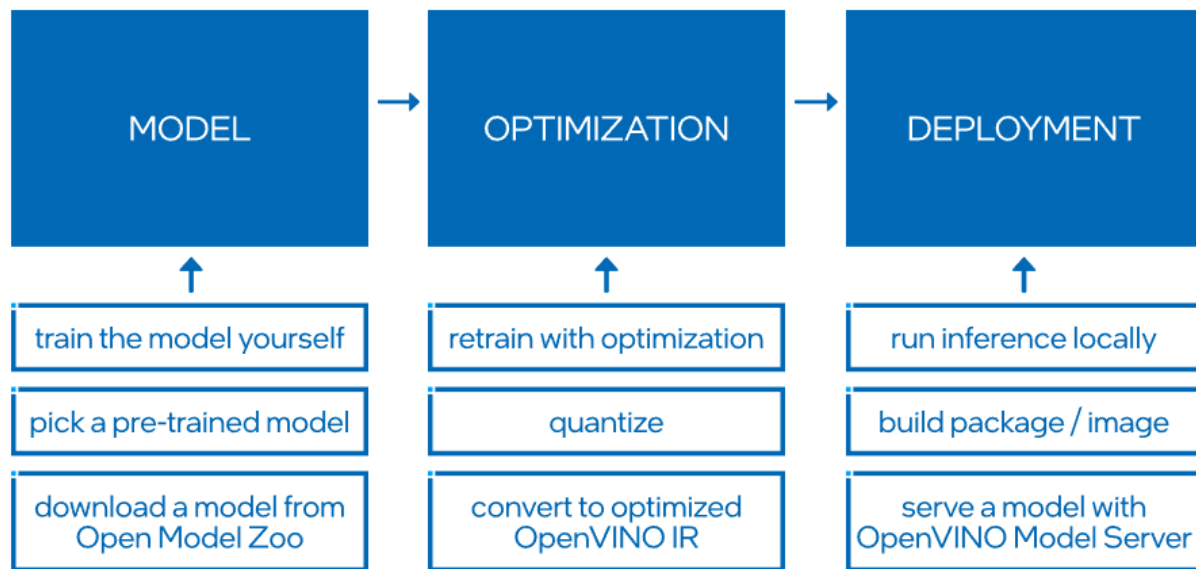


Figure 4.5- OpenVINO Workflow

#### 4.4.6 Convolution Neural Network

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of artificial neural networks (ANN) most commonly applied to analyze visual imagery. CNN's are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation-equivariant responses known as feature maps. Counter-intuitively, most convolutional neural networks are not invariant to translation due to the downsampling operation they apply to the input. They have applications in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series[27].

A convolutional neural network is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery because it is designed to emulate the behavior of biological behaviors of the animal visual cortex. It consists of convolutional layers and pooling layers so that the network can encode image properties.

#### 4.4.6.1 Description

The convolutional layer's parameters consist of a set of learnable filters (or kernels) that have a small receptive field. This way, the image can convolve across spatially, computing dot products between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. This way, the network learns filters that can activate when it detects special features on the input image's spatial feature.

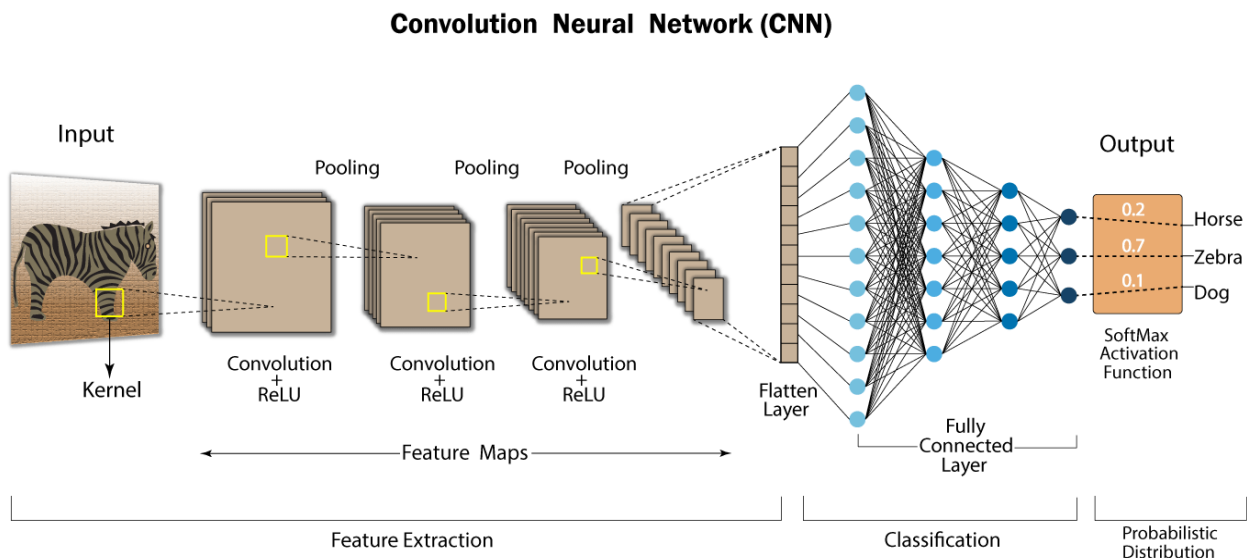


Figure 4.6-Typical CNN network

A pooling layer is a form of non-linear down-sampling. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The idea is to continuously reduce the spatial size of the input representation to reduce the number of parameters and computations in the network, so it can also control overfitting. Max pooling is the most common type of non-linear pooling.

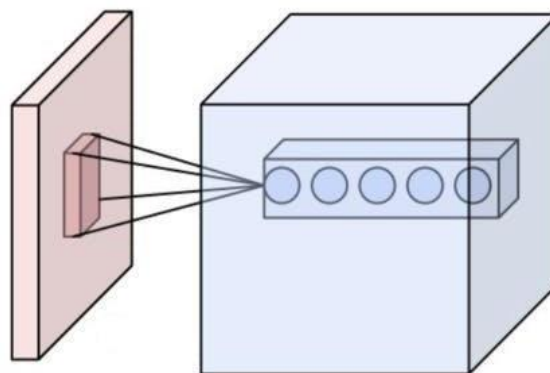


Figure 4.7- Neurons of a Convolution layer (Blue), connected to their receptive (Red)

## 5. IMPLEMENTATION & EXPERIMENTAL RESULT

### 5.1 Circuit Simulation and Output

The following section will explain the steps of the Skin Cancer A.I. Detection System. There are two main components of the system:

- Software Component
- Hardware Component

Open VINO- Open VINO toolkit is a free toolkit facilitating the optimization of a deep learning model from a framework and deployment using an inference engine onto Intel hardware. The high-level pipeline of Open VINO consists of two parts: generate I.R. (Intermediate Representation) files via Model Optimizer using your trained model or public one and execute inference on inference engine on specified plugins (CPU, Intel Processor Graphics, VPU, GNA, Multi-Device plugin, Heterogeneous plugin)

Setting up Open VINO-

Setting up of open vino on in virtual machine is primarily done so that we can run NCS2. First of all, we installed the current version of open vino. Then the detailed steps involved in its setting up are explained as follows: -

- After the download is complete, install the package. Unzip files, install dependencies and execute GUI installation as follows.
- Follow the on-screen instructions and wait till the installation is completed.
- Now we need to setup the environmental variables.
- OpenVINO toolkit environment variables are removed when you close the shell. So it is highly recommended that the environment variables be set permanently.
- Firstly, open the **.bashrc** file and append the following code at the end of the file.

**“source /opt/intel/computer\_vision\_sdk/bin/setupvars.sh”**

- Save the file and close the editor.
- To ensure that everything is working completely fine, open a new terminal, and the following message should be displayed.

**[setupvars.sh] OpenVino environment initialized.**

In the environment that has been created in Raspberry Pi, all its work has been done on the laptop, and finally, a folder will be created which will contain all classification.py files, XML files, BIN files, .mapping files, and all the photos. Now we just need to copy this folder into Raspberry Pi and run the command, and the environment will be created.

## Saving Model as Frozen Graph

When a network is defined in Python code, we have to create an inference graph file. Usually, graphs are built in a form that allows model training. That means that all trainable parameters are represented as variables in the graph. To be able to use such a graph with Model Optimizer, such a graph should be frozen. Freezing the model means producing a singular file containing information about the graph and checkpoint variables but saving these hyperparameters as constants within the graph structure. This eliminates additional information saved in the checkpoint files, such as the gradients at each point, which is included so that the model can be reloaded and training continued from where you left off. As this is not needed when serving a model purely for inference, they are discarded in freezing[28]. A frozen model is a file of the “.pb” file type. In order to generate a .pb file containing the necessary information, we first need to generate the checkpoint and graph files, which can call freeze\_graph.py, which then is used to generate the frozen graph file

```
from tensorflow.python.framework.convert_to_constants import convert_variables_to_constants_v2
import numpy as np

#path of the directory where you want to save your model
frozen_out_path = ''
# name of the .pb file
frozen_graph_filename = "frozen_graph"

# Convert Keras model to ConcreteFunction
full_model = tf.function(lambda x: model(x))
full_model = full_model.get_concrete_function(
    tf.TensorSpec(model.inputs[0].shape, model.inputs[0].dtype))

# Get frozen graph def
frozen_func = convert_variables_to_constants_v2(full_model)
frozen_func.graph.as_graph_def()

layers = [op.name for op in frozen_func.graph.get_operations()]
print("-" * 60)
print("Frozen model layers: ")
for layer in layers:
    print(layer)
print("-" * 60)
print("Frozen model inputs: ")
print(frozen_func.inputs)
print("Frozen model outputs: ")
```

```

# Get frozen graph def
frozen_func = convert_variables_to_constants_v2(full_model)
frozen_func.graph.as_graph_def()

layers = [op.name for op in frozen_func.graph.get_operations()]
print("-" * 60)
print("Frozen model layers: ")
for layer in layers:
    print(layer)
print("-" * 60)
print("Frozen model inputs: ")
print(frozen_func.inputs) print("Frozen model outputs: ")
print(frozen_func.outputs)

tf.io.write_graph(graph_or_graph_def=frozen_func.graph,
                  logdir=frozen_out_path,
                  name=f"{frozen_graph_filename}.pb",
                  as_text=False)
tf.io.write_graph(graph_or_graph_def=frozen_func.graph,
                  logdir=frozen_out_path,
                  name=f"{frozen_graph_filename}.pbtxt",
                  as_text=True)
from google.colab import files
files.download("/content/frozen_graph.pbtxt")

```

## Intermediate Representation Used in OpenVINO

The figure below illustrates the typical workflow for deploying a trained deep-learning model:

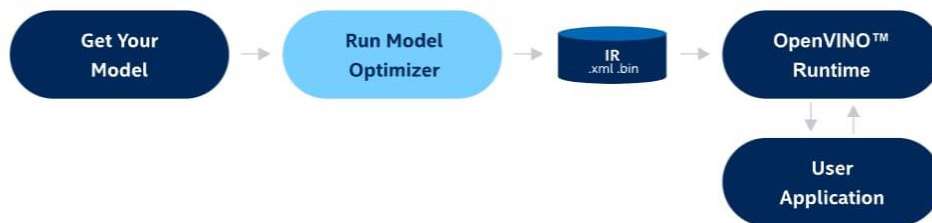


Figure 5.1- Flowchart of conversion of model

**Model Optimizer-** It is a cross-platform command-line tool that facilitates the transition between training and deployment environments, performs static model analysis, and adjusts deep learning models for optimal execution on end-point target devices. To use it, you need a pre-trained deep learning model in one of the supported formats: TensorFlow, PyTorch, PaddlePaddle, MXNet, Caffe,

Kaldi, or ONNX[29]. Here, we have used TensorFlow and MXNet deep learning models. Model Optimizer converts the model to the OpenVINO Intermediate Representation format (I.R.), which you can infer later with OpenVINO™ Runtime.

OpenVINO™ toolkit has its own format of graph representation and its own operation set. It can be represented primarily in three types:

- An XML file - Describes the network topology.

Different Tags used in the XML file

- <layer> used for describing a network topology for an operation node
- <edge> used for data flow connection

- a Binary file - Contains the weights and biases of binary data.
- a Mapping file

This representation is commonly referred to as the Intermediate Representation or I.R.

Each operation has a fixed number of attributes that define operation flavor (EDIT) used for a node. XML file doesn't have big constant values, like convolution weights. Instead, it refers to a part of the accompanying binary file that stores such values in a binary format. That files will be used for predictions on Raspberry board.

OpenVINO Runtime- It is a set of C++ libraries with C and Python bindings providing a common API to deliver inference solutions on the platform of your choice. Use the OpenVINO Runtime API to read an Intermediate Representation (I.R.)[30].

Running a trained model and prediction obtained with an NCS2 device

NCS2 Device- A Neural Compute Stick 2 is the second generation of deep learning development kit from Intel that is claimed to provide eight times the performance gain compared to its predecessor Modivus NCS. The reason for such an improvement and the heart of this device is the Intel latest VPU (Vision Processing Unit) Movidius Myriad X, which includes 16 SHAVE cores (Streaming Hybrid Architecture Vector Engine). It is four pieces of more than 12 cores in Myriad 2 installed in the NCS of the first generation.

We used the classification\_sample.py script, included in OpenVINO, to evaluate the performance of the model on NCS2. After we have applied all the rules (USB Rule) to make sure that NCS2 is running smoothly, we will write the following command line

## COMMAND LINE

```
python3 classification_sample.py -m <frozen_graph.xml> -i <IMAGE> -d <MYRIAD>
```

In this, we require a . XML file and an image. The API, after processing the image, gives us the class id and probability (0 for Benign and 1 for Malignant).

```
~/Desktop/Model$ python3 classification_sample.py -m frozen_graph.xml -i 1M.jpg -d MYRIAD
```

## OUTPUT

```
[ INFO ] Creating Inference Engine
[ INFO ] Loading network files:
         frozen_graph.xml
         frozen_graph.bin
classification_sample.py:76: DeprecationWarning: 'inputs' property of IENetwork class is deprecated. To access DataPtrs user need to use 'input_data'
         by 'input_info' property.
         assert len(net.inputs.keys()) == 1, "Sample supports only single input topologies"
[ INFO ] Preparing input blobs
[ INFO ] Batch size is 1
[ INFO ] Loading model to the plugin
[ INFO ] Starting inference in synchronous mode
[ INFO ] Processing output blob
[ INFO ] Top 10 results:
Image 1M.jpg

classid probability
-----
1         0.9941406
0         0.0062027

[ INFO ] This sample is an API example, for any performance measurements please use the dedicated benchmark_app tool
```

## The Myriad Interface

All of the production ML bundles support what we call the “Myriad Interface.” This means that you can perform multiple independent machine-learning activities within a single interface invocation. Many ML algorithms require multiple sequential steps to complete an activity. Each step requires synchronization among the nodes in the cluster. The more synchronizations required, the less optimal the parallel processing within the cluster. Running multiple activities in parallel does not increase the number of synchronizations versus a single activity, though each synchronization becomes larger (i.e., more data). While the larger synchronizations do impact performance, the impact is small compared to increasing the number of synchronization steps. This is due to latency (delays) in network communication. When waiting for responses from one node to another, no work can be done. Larger messages do not create such unproductive delays because there is no waiting except for the actual time of the data transfer on the wire. The magic of the Myriad Interfaces is embedded in the core record structures used by Machine learning. Machine learning uses several main record types in its interfaces:

- NumericField is used to provide real-valued data in matrix form.
- DiscreteField is used to convey discrete (integer-valued) data.
- Layout\_Model is used to encode the model that stores everything learned about the data.



## 5.2 Python Code Simulation and Output

### 5.2.1 Flow Chart of Code Approach

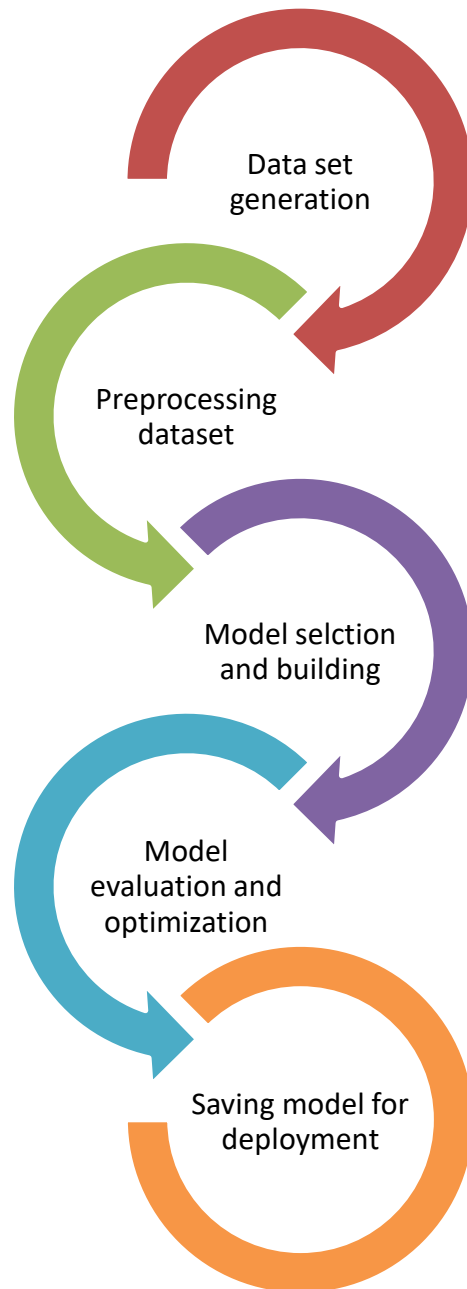


Figure 5.2- Flow Chart of ML model

## 5.2.2 Functions used in the Code and Explanation

### 5.2.2.1 Dataset Description

The dataset we used is from ISIC Archive, also available on Kaggle as the Skin Cancer: Malignant vs. Benign dataset. This dataset contains a balanced dataset of images of benign skin moles and malignant skin moles. The data consists of two folders with 1800 pictures (224x224) of the two types of moles, benign and malignant. It was collected by International Skin Imaging Collaboration (“ISIC”) as an international effort to improve melanoma diagnosis sponsored by the International Society for Digital Imaging of the Skin (ISDIS).

The ISIC Archive contains the largest publicly available collection of quality-controlled dermoscopic images of skin lesions which were collected from leading clinical centers internationally and acquired from a variety of devices within each center. Broad and international participation in image contribution is designed to ensure a representative clinically relevant sample. Most images have associated clinical metadata, which has been vetted by recognized melanoma experts. A subset of the images has undergone annotation and markup by recognized skin cancer experts.

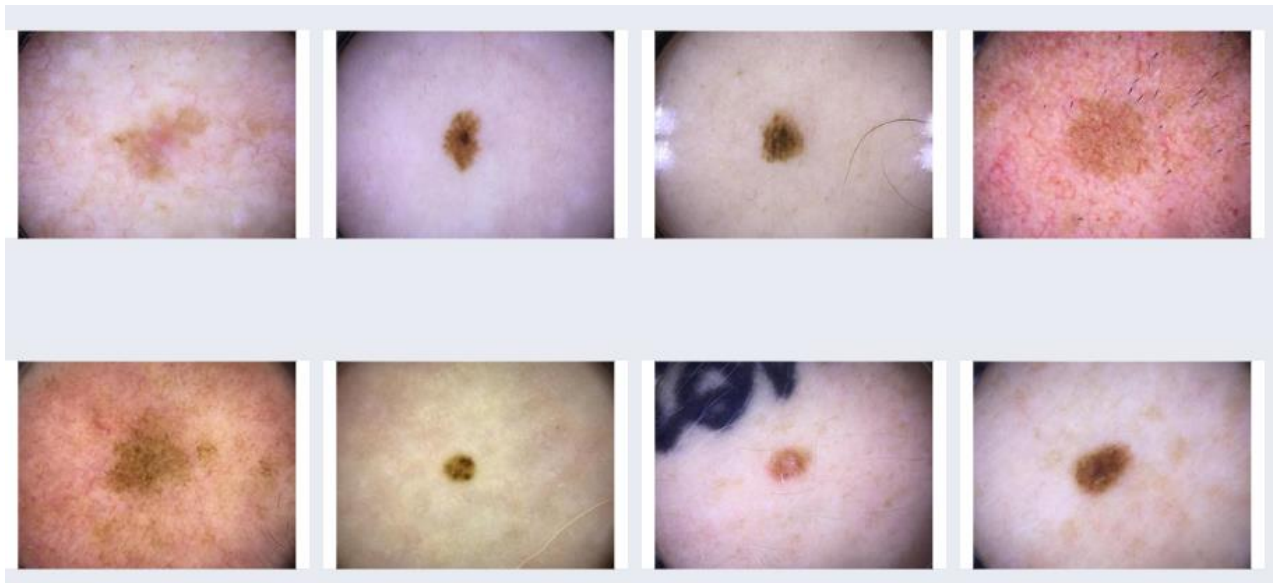


Figure 5.3- ISIC skin cancer dataset preview

### 5.2.2.2 Pre-processing using ImageDataGenerator

Image augmentation is a technique of applying different transformations to original images, which results in multiple transformed copies of the same image. Applying these small amounts of variations on the original image does not change its target class but only provides a new perspective of capturing the object in real life. Keras ImageDataGenerator class provides a quick and easy way to augment your images. It provides a host of different augmentation techniques like standardization, rotation, shifts, flips, brightness change, and many more. However, the main benefit of using the Keras ImageDataGenerator class is that it is designed to provide real-time data augmentation. Meaning it generates augmented images on the fly while your model is still in the training stage.

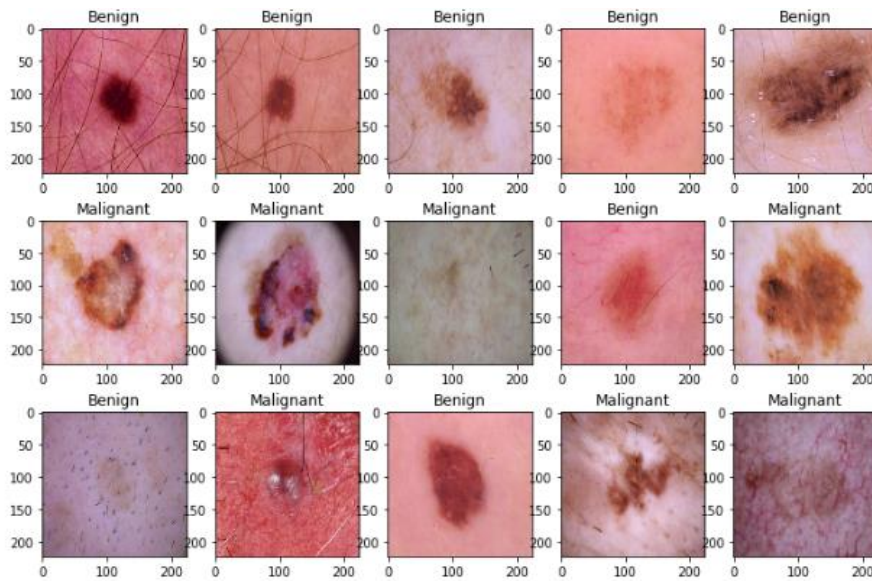


Figure 5.4- ImageDataGenerator Augmented Images

### 5.2.2.3 Elements of the Deep Neural Network Model

#### 1) Mobilenetv2

MobileNetV2 is a convolutional neural network architecture that seeks to perform well on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers. We use ReLU6 as the non-linearity because of its

robustness when used with low-precision computation. We always use kernel size three  $\times$  three as is standard for modern networks and utilize dropout and batch normalization during training. We tailor our architecture to different performance points by using the input image resolution and width multiplier as tuneable hyperparameters that can be adjusted depending on desired accuracy/performance trade-offs.

Our basic building unit has several properties that make it particularly suitable for mobile applications. It allows very memory-efficient inference and relies on utilizing standard operations present in all neural frameworks. Notably, our architecture for detection is 20 $\times$  less computation and 10 $\times$  fewer parameters than YOLOv2 on similar initial testing conditions.

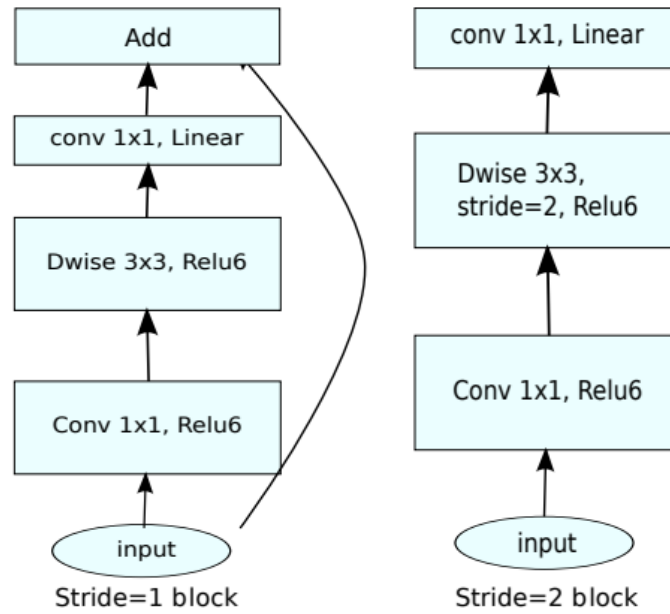


Figure 5.5- MobilenetV2 architecture

Input	Operator	$t$	$c$	$n$	$s$
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	-

Figure 5.6- MobilenetV2 parameters for the input image

## 2) Sequential Model

The main concept of transfer learning with very deep neural networks is to re-train a CNN model (on our dataset) that was previously trained on ImageNet [24] (about 1.2 million images). Since the dataset covers a wide range of objects (1000 different categories), so the model can learn diverse types of features, then they can be reused for other classification tasks.

To implement transfer learning with MobileNet, we first retrieved the previously extracted bottleneck features by MobileNet, then combined them with current extracted features from our Skin cancer images by training MobileNet on existing training data. Finally, we assign another class (rather than 1000 classes as in Imagenet) at the top of the model (for classification).

In this work, we used MobileNetV2 with 52 layers. The MobileNet structure is mainly based on depth-wise separable convolutions. Each Convolution layer is succeeded by a Batch Normalization and ReLU. The down-sampling is performed with stride-Convolution. in the depth-wise Convolution and in the first layer. Then, a final average pooling is applied to reduce the spatial resolution to be one before exposed to the fully connected layer. The whole MobileNet architecture consists of 13 of these blocks, as seen in Figure 1. In our case, the last layer has only two channels (Benign or Malignant), thus soft-max activation because it is a binary classification.

Building model with [https://tfhub.dev/google/tf2-preview/mobilenet\\_v2/feature\\_vector/2](https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/2)  
Model: "sequential"

Layer (type)	Output Shape	Param #
keras_layer (KerasLayer)	(None, 1280)	2257984
flatten (Flatten)	(None, 1280)	0
dense (Dense)	(None, 512)	655872
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 2)	1026
Total params: 2,914,882		
Trainable params: 656,898		
Non-trainable params: 2,257,984		

Figure 5.7- Model Summary

#### 5.2.2.4 Training the model

Compiling the model takes three parameters: optimizer, loss, and metrics. The optimizer controls the learning rate. We will be using ‘adam’ as our optimizer. Adam is generally a good optimizer to use for many cases. The adam optimizer adjusts the learning rate throughout training. The learning rate determines how fast the optimal weights for the model are calculated. A lower learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer; thus, there is a trade-off between them. In our case, we have set the learning rate to 0.001. We will use ‘categorical\_crossentropy’ for our loss function. This is the most common choice for classification. A lower score indicates that the model is performing better. We will use the ‘accuracy’ metric to see the accuracy score on the validation set when we train the model.

Now we will train our model. To train, we will use the ‘fit()’ function on our model with the following parameters: training data, validation data, and the number of epochs and steps per epoch. The number of epochs is the number of times the model will cycle through the data. The more epochs we run, the more the model will improve up to a certain point. After that point, the model will stop improving during each epoch. For our model, we will set the number of epochs to 10.

```
Epoch 1/10
133/133 [=====] - 404s 3s/step - loss: 0.4741 - accuracy: 0.7905 - val_loss: 0.5188 - val_accuracy: 0.7500
Epoch 2/10
133/133 [=====] - 37s 278ms/step - loss: 0.3748 - accuracy: 0.8317 - val_loss: 0.4834 - val_accuracy: 0.7765
Epoch 3/10
133/133 [=====] - 37s 279ms/step - loss: 0.3562 - accuracy: 0.8435 - val_loss: 0.4335 - val_accuracy: 0.8182
Epoch 4/10
133/133 [=====] - 37s 280ms/step - loss: 0.3403 - accuracy: 0.8501 - val_loss: 0.3881 - val_accuracy: 0.8295
Epoch 5/10
133/133 [=====] - 38s 285ms/step - loss: 0.3342 - accuracy: 0.8501 - val_loss: 0.5133 - val_accuracy: 0.7614
Epoch 6/10
133/133 [=====] - 38s 287ms/step - loss: 0.3411 - accuracy: 0.8468 - val_loss: 0.4507 - val_accuracy: 0.7992
Epoch 7/10
133/133 [=====] - 37s 279ms/step - loss: 0.3106 - accuracy: 0.8591 - val_loss: 0.4331 - val_accuracy: 0.8163
Epoch 8/10
133/133 [=====] - 37s 277ms/step - loss: 0.3067 - accuracy: 0.8681 - val_loss: 0.4449 - val_accuracy: 0.7992
Epoch 9/10
133/133 [=====] - 37s 277ms/step - loss: 0.3055 - accuracy: 0.8643 - val_loss: 0.4717 - val_accuracy: 0.7973
Epoch 10/10
133/133 [=====] - 38s 287ms/step - loss: 0.2901 - accuracy: 0.8700 - val_loss: 0.4351 - val_accuracy: 0.8144
```

Figure 5.8- Model Training

#### 5.2.2.5 Learning Curves

A learning curve is a plot of model learning performance over experience or time. During the training of a machine learning model, the current state of the model at each step of the training algorithm can be evaluated. It can be evaluated on the training dataset to give an idea of how well the model is “learning.” It can also be evaluated on a hold-out validation dataset that is not part of the training dataset. Evaluation of the validation dataset gives an idea of how well the model is “generalizing.”

- Train Learning Curve: The learning curve calculated from the training dataset gives an idea of how well the model is learning.

- **Validation Learning Curve:** Learning curve is calculated from a hold-out validation dataset that gives an idea of how well the model is generalizing.

In our case of classification predictive modeling problems, where the model is optimized according to cross-entropy loss, and model performance is evaluated using classification accuracy. In this case, two plots are created, one for the learning curves of each metric, and each plot shows two learning curves, one for each of the train and validation datasets.

- **Optimization Learning Curves:** Learning curves are calculated on the metric by which the parameters of the model are being optimized, that is, loss.
- **Performance Learning Curves:** Learning curves are calculated on the metric by which the model will be evaluated and selected, that is, accuracy.

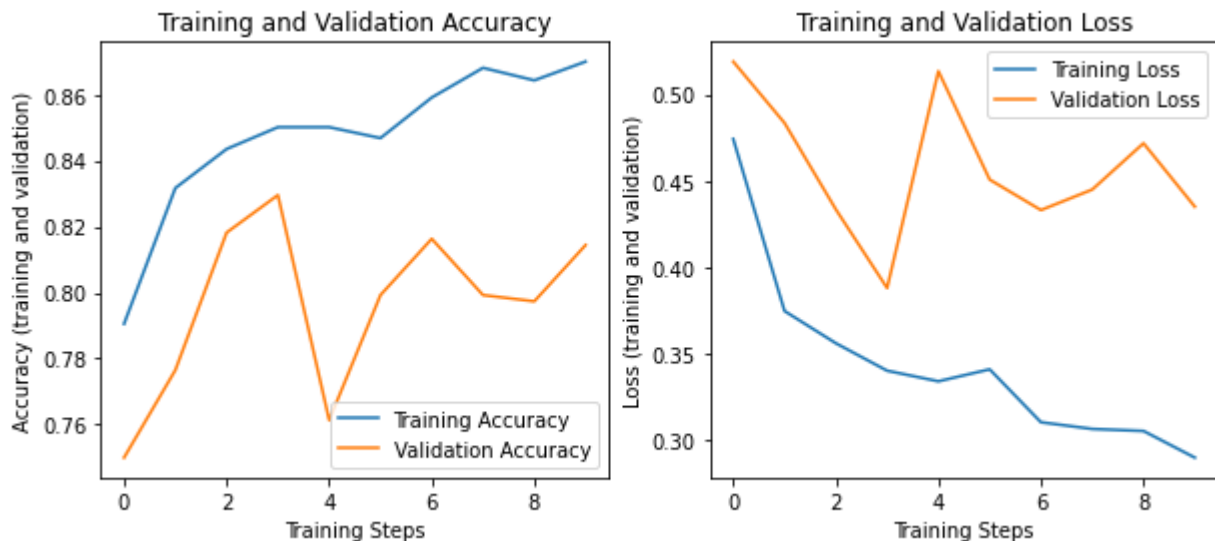


Figure 5.9- Learning Curves

### 5.2.2.6 Evaluating Model Performance

**1) Confusion matrix:** A Confusion matrix is an  $N \times N$  matrix used for evaluating the performance of a classification model, where  $N$  is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making. For a binary classification problem, we would have a  $2 \times 2$  matrix with four values as follows:

True Positive (T.P.)

- The predicted value matches the actual value
- The actual value was positive, and the model predicted a positive value

True Negative (T.N.)

- The predicted value matches the actual value
- The actual value was negative, and the model predicted a negative value  
False Positive (F.P.) – Type 1 error
- The predicted value was falsely predicted
- The actual value was negative, but the model predicted a positive value
- Also known as the Type 1 error  
False Negative (F.N.) – Type 2 error
- The predicted value was falsely predicted
- The actual value was positive, but the model predicted a negative value
- Also known as the Type 2 error

The matrix shown below depicts that the model is not biased toward any type 1 or type 2 errors, with almost equal amounts of False Positives and False Negatives. The results of the Confusion Matrix will further be used to calculate various metrics in the classification report, which forms the basis for understanding and improving the model.

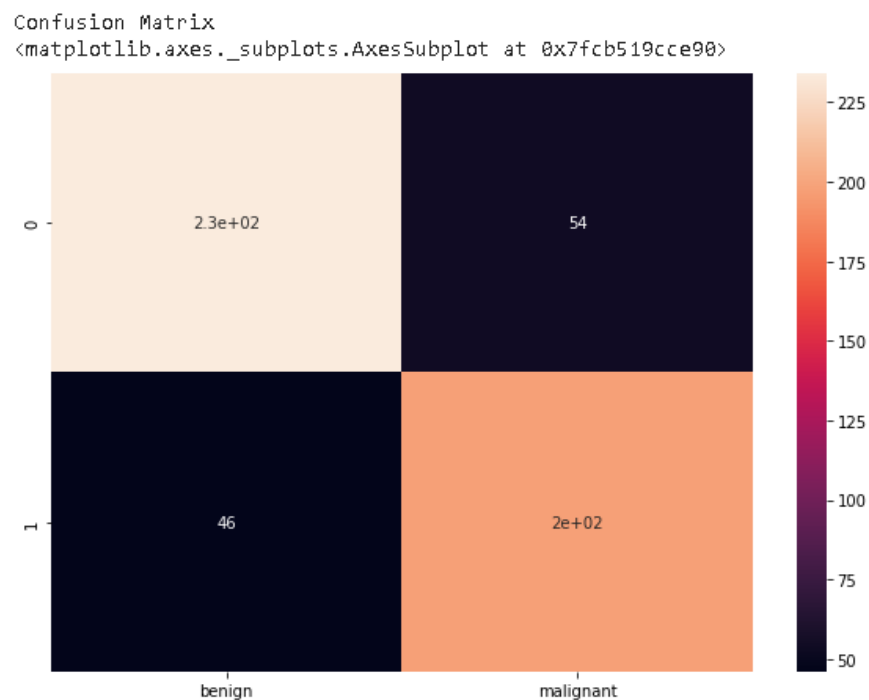


Figure 5.10- Confusion Matrix

**2) Classification Report:** The classification report visualizer displays the precision, recall, F1, and support scores for the model.

Precision:- Accuracy of positive predictions.

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

Recall:- Fraction of positives that were correctly identified.



Recall =  $TP / (TP + FN)$

F1 Score:- weighted harmonic mean of precision and recall

F1 Score =  $2 * (Recall * Precision) / (Recall + Precision)$

Support: Number of actual occurrences of the class in the specified dataset

```
34/34 [=====] - 8s 224ms/step
Classification Report
      precision    recall  f1-score   support

   benign       0.84       0.81       0.82        288
  malignant       0.79       0.81       0.80        244

   accuracy                   0.81        532
  macro avg       0.81       0.81       0.81        532
 weighted avg       0.81       0.81       0.81        532

34/34 [=====] - 7s 213ms/step - loss: 0.4250 - accuracy: 0.8271
[0.42497560381889343, 0.8270676732063293]
```

Figure 5.11: Classification Report

The above report clearly depicts that both classes are accurately classified, with an accuracy of 82.71%. F1 scores depict the weighted mean of precision and recall with 80% and 82% scores for benign and malignant, respectively, which indicates that our model is balanced and is not tipped towards either of the classes. These factors indicate that the model developed is realistic and can be deployed for classification.

### 5.2.2.7 Making Predictions

To predict the values for the untrained data, we use predict() function enables us to predict the labels of the data values on the basis of the trained model. It accepts only a single argument which is the image data to be tested. We tested on randomly selected images, an equal amount for each class from the ISIC skin cancer testing dataset consisting of 16000 images. The average confidence is 86% and 270m/s on average to evaluate each image.

```
133/133 [=====] - 36s 270ms/step - loss: 0.3004 - accuracy: 0.8610
```

Figure 5.12: Making Predictions

### 5.2.2.8 Saving Model as Frozen Graph

Freezing the model means producing a singular file containing information about the graph and checkpoint variables but saving these hyperparameters as constants within the graph structure. This eliminates additional information saved in the checkpoint files, such as the gradients at each point, which is included so that the model can be reloaded and training continued from where you left off. As this is not needed when serving a model purely for

inference, they are discarded in freezing. A frozen model is a file of the “.pb” file type. In order to generate a .pb file containing the necessary information, we first need to generate the checkpoint and graph files, which to can call freeze\_graph.py, which then is used to generate the frozen graph file.

## 6. OUTCOME AND PROSPECTIVE LEARNING

### 6.1 Scope and Outcomes

Skin cancer is seen as one of the most hazardous forms of cancer found in humans. It's common cancer that can form on any part of the body, but it often occurs on sun-exposed skin. Existing solutions such as image pre-processing, which includes hair removal, de-noise, sharpening, and resizing of the given skin image, and segmentation- which is used for segmenting out the region of interest from the given image. Some commonly used algorithms are k-means, the threshold in the histogram, etc., features extraction from the image, and classification of the image from the features set extracted from the segmented image.

A variety of classification algorithms are used for this purpose. The recent skin cancer detection technology uses machine learning and deep learning-based algorithms for the classification of its type and whether a tumor is cancerous or not. The most commonly used classification algorithms are support vector machine (SVM), feed-forward artificial neural network, and deep convolutional neural network.

The building block of the Skin Cancer A.I. detector is deep learning which is used to process an image, then recognize it and show the output accordingly. Using Convolutional Neural Networks, we developed algorithms and models to distinguish between benign and malignant skin cancers. On detection of a mole as cancerous, we can even further identify which type of cancer is there, which are as follows nevus, melanoma, and seborrheic keratosis. If we can detect the type of cancer, it will be a great added advantage as patients can be made aware at a very early stage.

These methods exist, but we need something that can detect cancer at a very early stage so that it is easily curable. Therefore, it is necessary to establish a system such as a skin cancer A.I. detection system. The base of the Skin Cancer A.I. detector is a deep learning model which is used to process an image, then recognize it and show the output accordingly. TensorFlow and Keras are used along with a Convolutional Neural Network(CNN) architecture called MobileNet v2 to build and create a machine-learning model. Using CNNs, we developed algorithms and models to distinguish between benign and malignant types of skin cancers. On detection of a mole to whether it is cancerous or not.

## 6.2 Prospective Learning

During this project, we faced many difficulties and challenges in terms of knowledge, implementation of software & hardware, gathering requirements, making action plans accordingly and, managing the team, and allocating the work according to team members' caliber, etc.

The findings and learnings from the project are listed below:

1. We got to know how to work in a team in a timely manner, divide work among teammates and how to manage and help each other on the way
2. We got to know how to implement an ML algorithm to solve a real-world problem that has an impact on millions of people around the globe each and every year.
3. We got to know how to synchronize hardware with software in a harmonious way and get the expected end results.
4. We got to know how to use APIs alongside Raspberry Pi to get the output classification.
5. We got to find out the number of people that are suffering from skin cancer on a daily basis, their lifespan, and the severity of the tumor.

## 6.3 Conclusion

Skin cancer is seen as one of the most hazardous forms of cancer found in humans. It's common cancer that can form on any part of the body, but it often occurs on sun-exposed skin. Existing solutions such as image pre-processing, which includes hair removal, de-noise, sharpening, and resizing of the given skin image, and segmentation- which is used for segmenting out the region of interest from the given image. The synchronization of the hardware lock embedded system and the server-based application is crucial for the system to operate seamlessly. A device has been developed which uses images of skin tumors and classify between whether an image is benign or malignant. The device primarily consists of a camera and Raspberry Pi 3B+, and computation power to execute the model is provided by Intel® Movidius™ Neural Compute Stick 2. It uses less power than most mobile phones these days without compromising on time. The core of execution is Open VINO toolkit, which helps in running the model with NCS2. The model was developed using TensorFlow. The Convolution Neural Network used to make a lightweight model works successfully individually and can be used in mobile applications, web applications, etc. But we decided to make a portable device to remove dependence on the internet. The device can be transported to remote locations for easy implementation. The model conversion and API used are provided by Open VINO toolkit. In summary, building this multi-layered system requires the application of multi-disciplinary engineering skills, and its implementation offers the potential to add ease, value, and convenience to a large body of students and users.

## 7. GANTT CHART

### 7.1 Work Breakdown and Gantt Chart

This project was completed by a team of 5 ambitious members who have equally contributed to each aspect of the project. The absence of even one member would have been detrimental to the progress of this project. The tasks completed in the span of 11 months are specified below.

1. Team formation & Mentor Section: It was the process of finding compatible minds who were ready to embark upon this journey. Here we also put preferences for the mentor who will be our guardians throughout.
2. Brainstorm & Idea Selection: Out of numerous, we chose the project which was ideal for us based on numerous parameters.
3. Research & Literature Survey: The most difficult part of the project where we read multiple papers and tried to find out which road to take based on accessibility and accuracy. Here multiple options were thoroughly studied, and the most optimized version of the project was selected.
4. Final Selection of ML Model & Hardware: The hardware version and model and their interconnectivity were selected carefully, taking into account various constraints that we faced.
5. Configuration of OpenVINO across devices and other requirements: OpenVINO was the backbone of our project, and it was configured and installed in Virtual Machine and Raspberry Pi 3B+.
6. Dataset Formation & Pre-processing: The dataset of Benign and Malignant skin cancers was formed, and pre-processing and data cleaning were executed.
7. Formation of CNN model: A pre-trained model called MobileNet v2, which is also known as transfer learning, was used to train on the above-mentioned dataset.
8. Conversion to IR: To run the model using the selected devices, we had to convert the model to a frozen graph and then to IR, which was our desirable state.
9. Integration of Hardware and Software: The model which was created earlier and configured hardware were combined, making sure that compatible versions were used. The device was functional at this point.
10. Testing and Fine Tuning: The final device, which is now fully compiled, was tested extensively, and based on accuracies, the model was fine-tuned till accuracy reached desirable numbers.
11. Report Formation(Documentation): Here, we documented our progress and presented it in a detailed document. This was by far the longest-running part of this project creation.

Below we have shown the Gantt chart throughout the year and the breakdown based on the above-mentioned tasks.

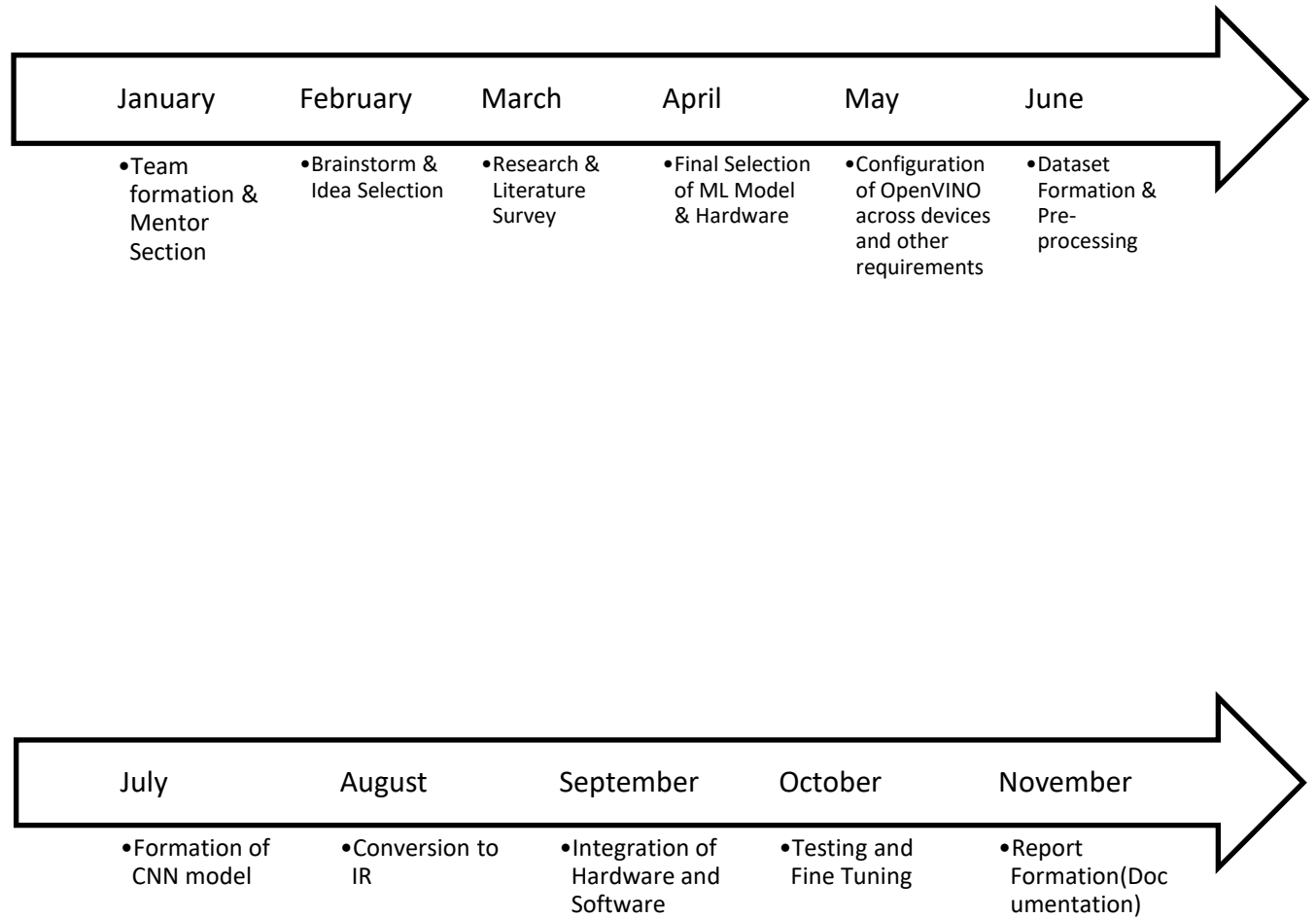
Table 7.1

Task \ Month	Jan 2022	Feb 2022	March 2022	April 2022	May 2022	June 2022
<i>Team formation &amp; Mentor Selection</i>						
<i>Brainstorming &amp; Idea Selection</i>						
<i>Research &amp; Literature Survey</i>						
<i>Final selection of ML Model &amp; Hardware</i>						
<i>Configuration of Open VINO across devices and other requirements</i>						
<i>Dataset Formation &amp; Pre-processing</i>						

Table 7.2

Task \ Month	July 2022	Aug 2022	Sep 2022	Oct 2022	Nov 2022
<i>Formation of CNN Model</i>					
<i>Conversion of model to IR</i>					
<i>Integration of HW and SW</i>					
<i>Testing &amp; Fine Tuning</i>					
<i>Report Formation (Documentation)</i>					

## 7.2 Project Timeline



## 7.3 Individual Gantt Chart

Table 7.4

Member \ Month	Jan 2022	Feb 2022	March 2022	April 2022	May 2022	June 2022
Tanmay Garg	BrainStorming & Idea Selection	Research	Hardware Configuration		Open VINO env. initialization and Configuration	
Avneet Singh Maingi	BrainStorming & Idea Selection	Research	Hardware Configuration		Open VINO env. initialization and Configuration	
Aditya Chawla	BrainStorming & Idea Selection	Research	ML Model Selection	DataSet Gathering	Dataset PreProcessing	
Aakarshan Gupta	BrainStorming & Idea Selection	Literature Survey		DataSet Gathering	DataSet Formation	Mid-Term Report Formation
Akshit Gupta	BrainStorming & Idea Selection	Literature Survey		DataSet Gathering	DataSet Formation	Mid-Term Report Formation

Table 7.5

Member \ Month	July 2022	Aug 2022	Sep 2022	Oct 2022	Nov 2022
Tanmay Garg	Bug Fixes & Troubleshooting	Model Conversion to IR	Integration of HW and SW		Final Report Formation
Avneet Singh Maingi	Bug Fixes & Troubleshooting	Model Conversion to IR	Integration of HW and SW		Final Report Formation
Aditya Chawla	CNN Model Formation	CNN Model Formation	Testing & Fine Tuning	Final Report Formation	Final Report Formation
Aakarshan Gupta	Mid-Term Report Formation	Bug Fixes & Troubleshooting	Testing & Fine Tuning	Final Report Formation	Final Report Formation
Akshit Gupta	Mid-Term Report Formation	Bug Fixes & Troubleshooting	Testing & Fine Tuning	Final Report Formation	Final Report Formation



## REFERENCES

- [1] Can You Die from Skin Cancer? Survival Rates and Prevention - Healthline.  
<https://www.healthline.com/health/can-you-die-from-skin-cancer>
- [2] Clean Water AI - Arduino Project Hub.  
<https://create.arduino.cc/projecthub/clean-water-ai/clean-water-ai-e40806>
- [3] Skin cancer - Symptoms and causes - Mayo Clinic.  
<https://www.mayoclinic.org/diseases-conditions/skin-cancer/symptoms-causes/syc-20377605>
- [4] Jutzi, T. B., Utikal, J. S., Hauschild, A., Sondermann, W., Fröhling, S., Hekler, A., Schmitt, M., Maron, R. C., & Brinker, T. J. (2020). Artificial Intelligence in Skin Cancer Diagnostics: The Patients' Perspective
- [5] Das, K., Cockerell, C. J., Patil, A., Pietkiewicz, P., Giulini, M., Grabbe, S., & Goldust, M. (2021). Machine Learning and Its Application in Skin Cancer.
- [6] Takiddin, A., Schneider, J., Yang, Y., Abd-Alrazaq, A., & Househ, M. (2021). Artificial Intelligence for Skin Cancer Detection: Scoping Review
- [7] Hasan, M. R., Fatemi, M. I., Khan, M. M., Kaur, M., & Zaguia, A. (2021). Comparative Analysis of Skin Cancer (Benign vs. Malignant) Detection Using Convolutional Neural Networks.
- [8] E. Jana, R. Subban, and S. Saraswathi, "Research on Skin Cancer Cell Detection Using Image Processing," 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 2017
- [9] V, Bhavya & G, Narasimha & M, Ramya & Y, Sujana & AnuRadha, T.. (2018). Classification of skin cancer images using TensorFlow and inception v3
- [10] A. Gautam and B. Raman, "Skin Cancer Classification from Dermoscopic Images using Feature Extraction Methods," 2020 IEEE REGION 10 CONFERENCE (TENCON)
- [11] Divya, N. & Dsouza, Deepthi & Hariprasad (2021). Cure skin - Skin Disease Prediction using MobileNet Model.
- [12] Z. N. Fikile Gasa, P. A. Owolawi, T. Mapayi, and K. Odeyemi, "MobileNet Neural Network skin disease detector with Raspberry pi," 2020

- [13] Google Colab.  
<https://research.google.com/colaboratory/faq.htm>
- [14] Intel Movidius Neural Compute Stick 2 – FactoryForward India.  
<https://www.factoryforward.com/product/intel-movidius-neural-compute-stick-2/>
- [15] Computer-Aided Detection of Melanoma, A Case.  
[https://194.149.136.10/bitstream/20.500.12188/21019/1/Computer\\_Aided\\_Detection\\_of\\_Melanoma\\_\\_A\\_Case\\_Study.pdf](https://194.149.136.10/bitstream/20.500.12188/21019/1/Computer_Aided_Detection_of_Melanoma__A_Case_Study.pdf)
- [16] All about Machine Learning - c-sharpcorner.com.  
<https://www.c-sharpcorner.com/article/a-complete-machine-learning-tutorial/>
- [17] Embedded system - Wikipedia.  
[https://en.wikipedia.org/wiki/Embedded\\_system](https://en.wikipedia.org/wiki/Embedded_system)
- [18] GitHub - TanmayGarg001/SkinCancerDetectionAI.  
<https://github.com/TanmayGarg001/SkinCancerDetectionAI>
- [19] Deep Learning for Computer Vision - Class Central.  
<https://www.classcentral.com/course/swayam-deep-learning-for-computer-vision-19838>
- [20] ISO/IEC/IEEE 24748-1:2018.  
<https://www.iso.org/standard/72896.html>
- [21] What is a Raspberry Pi? | Opensource.com.  
<https://opensource.com/resources/raspberry-pi>
- [22] Introduction to Raspberry Pi 3 B+ - The Engineering Projects.  
<https://www.theengineeringprojects.com/2018/07/introduction-to-raspberry-pi-3-b-plus.html>
- [23] What is TensorFlow? How it Works? Introduction & Architecture - Guru99.  
<https://www.guru99.com/what-is-tensorflow.html>
- [24] Architecture of TensorFlow - Javatpoint. <https://www.javatpoint.com/tensorflow-architecture>
- [25] keras/README.md at master · keras-team/keras · GitHub. <https://github.com/keras-team/keras/blob/master/README.md>
- [26] OpenVINO™ Toolkit Overview - OpenVINO™ Toolkit.  
<https://docs.openvino.ai/2021.2/index.html>

- [27] Convolutional neural network - Wikipedia.  
[https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)
  
- [28] Freezing a Keras model. How to freeze a model for serving and... | by ....  
<https://towardsdatascience.com/freezing-a-keras-model-c2e26cb84a38>
  
- [29] Model Optimizer Usage — OpenVINO™ documentation.  
[https://docs.openvino.ai/latest/openvino\\_docs\\_MO\\_DG\\_Deep\\_Learning\\_Model\\_Optimizer\\_DevGuide.html](https://docs.openvino.ai/latest/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html)
  
- [30] Inference with OpenVINO Runtime — OpenVINO™ documentation.  
[https://docs.openvino.ai/latest/openvino\\_docs\\_OV\\_UG\\_OV\\_Runtime\\_User\\_Guide.html](https://docs.openvino.ai/latest/openvino_docs_OV_UG_OV_Runtime_User_Guide.html)