

Transfer Learning for Multi-Class Skin Cancer Detection using  
Deep Convolutional Neural Networks

## DISSERTATION

Submitted in partial fulfillment of the requirements of the  
MTech Data Science and Engineering Degree programme

By

Shreya Kshirsagar  
2022AA05455

Under the supervision of

Sarang Ashtankar  
Associate Manager

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE  
Pilani (Rajasthan) INDIA

June, 2024

## 1. Acknowledgment

"I would like to express my heartfelt thanks to my evaluator, Vinaya Sathyanarayana, for providing invaluable feedback and constructive suggestions throughout the course of my project. Your guidance has played a crucial role in helping me refine my work and meet the objectives of this research. I am also deeply appreciative of my mentor, Sarang Ashtankar, whose constant support and encouragement have been a source of great motivation. Your expertise and insights were essential in overcoming the challenges of this project, and your mentorship has greatly contributed to my learning and development."

This project, titled *"Transfer Learning for Multi-Class Skin Cancer Detection using Deep Convolutional Neural Networks,"* has been a challenging yet immensely rewarding endeavor. The knowledge and skills gained through this research would not have been possible without the support and mentorship provided by both of you.

Thank you for your dedication and commitment to my academic and professional development.

## 2. BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

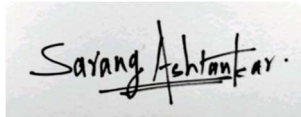
### CERTIFICATE

This is to certify that the Dissertation entitled - Transfer Learning for Multi-Class Skin Cancer Detection using Deep Convolutional Neural Networks

and submitted by Ms. \_\_\_\_\_ Shreya Kshirsagar \_\_\_\_\_ ID  
No. \_\_\_\_2022AA05455\_\_\_\_\_

in partial fulfillment of the requirements of DSECLZG628T /  
AIMLCZG628T Dissertation, embodies the work

done by her under my supervision.



Signature of the Supervisor

Place: Delhi

Date: \_\_\_\_9/9/2024\_\_\_\_

Name: Sarang Ashtankar  
Designation: Associate Manager

### **3. Abstract:**

Skin cancer is a very serious type of cancer that starts when skin cells get damaged and their DNA isn't repaired properly. This can cause genetic changes that lead to cancerous growths on the skin. Detecting skin cancer early is crucial because it's easier to treat when caught early on. Unfortunately, if it spreads to other parts of the body, it becomes more difficult to cure.

Due to the increasing number of cases, high death rates, and expensive treatments, it's really important to find ways to diagnose skin cancer as soon as possible. Researchers have developed different techniques to help with early detection. They look at features like the shape, color, size, and symmetry of skin lesions to tell if they might be cancerous, especially melanoma, which is a very dangerous form of skin cancer.

This project aims to utilize an advanced computer technology known as a deep convolutional neural network. This network will be trained using a method called transfer learning to recognize and classify three main types of skin cancer:

- (i) Basal Cell Carcinoma
- (ii) Melanoma
- (iii) Nevus

This could improve how quickly and accurately doctors can diagnose and treat these types of cancer.

## Table of Contents:

1. Acknowledgement .....	2
2. Certificate from Supervisor .....	3
3. Dissertation Abstract .....	4
4. List of Symbols and Abbreviations .....	7
5. List of Tables .....	8
6. List of Figures .....	10
7. Chapter 1: Introduction	
7.1. Problem Statement .....	11
7.2. Objective .....	12
7.3. Scope of Work .....	13
7.4. Challenges .....	15
7.5 Code outline.....	17
7.6 Flowchart.....	19
8. Chapter 2: Data Handling	
8.1. Data Collection .....	20
8.2. Source of Data .....	20
8.3. Dataset Description .....	20
8.4. Challenges with Data Collection .....	21
9. Chapter 3: Setup and Importing Libraries	
9.1. Importing Required Libraries .....	22
10. Chapter 4: Data Preparation .....	24
11. Chapter 5: Model Building .....	27
12. ROC- AUC curve .....	33

13.	Confusion matrix .....	35
14.	Classification report .....	37
15.	Conclusion .....	38
16.	References -----	39

#### 4. List of symbols and abbreviations:

Symbol/Abbreviation	Description
CNN	Convolutional_Neural_Network
DNN	Deep_Neural_Network
DL	Deep_Learning
Transfer Learning	A technique which is a pre-trained model used for fine-tuning on a new dataset
MobileNetV2	A lightweight deep learning model designed for mobile and edge devices
Grad-CAM	This is a technique to visualize where a model is focusing in an image
ROC	Receiver Operating Characteristic; a curve plotting true positive rate vs. false positive rate
AUC	Area Under the ROC Curve; a measure of model performance
FPR	False Positive Rate
TPR	True Positive Rate
BS	Batch Size
CNN	Convolutional Neural Network
ImageNet	A large visual database designed for use in visual object recognition software research
Keras	An advanced neural network API built in Python, designed to operate on top of TensorFlow.
BS	Batch Size
True Positive (TP)	Correctly predicted positive cases
False Positive (FP)	Incorrectly predicted positive cases

True Negative (TN)	Correctly predicted negative cases
False Negative (FN)	Incorrectly predicted negative cases
Precision	The proportion of correctly identified positive instances out of all predicted positive cases.
Recall	The proportion of correctly predicted positive instances to the total number of actual positive instances
F1-Score	The harmonic mean of precision and recall
One-Hot Encoding	A technique for transforming categorical variables into a binary matrix format.
Softmax	A function used to normalize the output of a neural network to a probability distribution over multiple classes
Grad-CAM	Gradient-weighted Class Activation Mapping; a technique to visualize where a model is focusing in an image
HDF5	Hierarchical Data Format version 5 (HDF5) is a file format designed for storing vast quantities of data.
TPR	True Positive Rate
FPR	False Positive Rate
AUC	Area Under the Curve; a metric for evaluating classification performance
BS	Batch Size
Acc	Accuracy

|



## 5. List of Tables

Table 1	Dataset Overview	It tells us the count of samples used
Table 2	Confusion Matrix Table	The confusion matrix represents the true labels versus the predicted labels.
Table 3	ROC Curve Data Table	The ROC curve visually depicts a classifier's performance at different threshold levels.

## 6. List of Figures

1. Flowchart	.....19
--------------	---------

## **7.Introduction**

### **7.1. Problem Statement:**

Skin cancer is among the most prevalent cancers globally. Timely detection and precise diagnosis are essential for successful treatment and better patient outcomes. The three predominant forms of skin cancer are Basal Cell Carcinoma (BCC), Melanoma, and Nevus (a benign condition). Dermatologists typically perform manual diagnoses through visual inspection of skin lesions, a method that can be subjective and susceptible to inaccuracies.

Hence prime focus of this project is to develop an automated skin cancer detection system using Transfer Learning and Deep Convolutional Neural Networks (CNNs). The system aims to classify skin lesions into one of three categories: Basal Cell Carcinoma (BCC), Melanoma, and Nevus, based on images of skin lesions. By leveraging pre-trained models and fine-tuning them on a specialized dataset, the goal is to achieve high accuracy and reliability in classification.

## 7.2. OBJECTIVES:

### 1. Develop a Skin Cancer Classification System

- Create and implement a deep learning system that accurately identifies three major types of skin cancer (melanoma, basal cell carcinoma, and squamous cell carcinoma) from skin lesion images. Explore transfer learning techniques to leverage pre-trained convolutional neural networks, focusing on achieving high performance, robust generalization to new data, and efficient computation.

### 2. Evaluate System Performance:

Assess the developed classification system using standard metrics like accuracy sensitivity, and area under the ROC curve (AUC). Compare the system's performance against existing methods and benchmarks in skin cancer detection.

### 3. Document Methodology and Results

Document the entire process of developing, training, and evaluating the deep learning model.

### 4. Explore Augmentation Strategies

Explore data augmentation techniques to enhance the model's ability to generalize to different skin lesion variations and conditions. Investigate how augmenting the training data with variations like rotations, flips, and color adjustments impacts classification performance.

### 5. Consider Computational Efficiency:

Optimize the deep learning model for computational efficiency without compromising classification accuracy. Evaluate the trade-offs between model complexity and inference speed to ensure practical deployment in real-world clinical settings.

### 6. Address Class Imbalance:

- Address any imbalance in the distribution of skin cancer types in the dataset. Investigate methods such as class weighting, oversampling, or undersampling to improve the model's ability to correctly classify underrepresented classes. These objectives expand on the initial goals by focusing on interpretability, data augmentation, computational efficiency, and handling class imbalance, all of which are critical for developing a robust and effective skin cancer classification system.

## 7.3 Scope of Work

### 1. Objective

The primary objective of this project is to develop and evaluate a deep learning model that accurately classifies skin lesions into multiple classes (Basal Cell Carcinoma, Melanoma, and Nevus) using transfer learning with convolutional neural networks (CNNs). This involves leveraging pre-trained models, specifically MobileNetV2, to improve classification performance and generalization.

### 2. Dataset Preparation

**Collection:** Utilize the ISIC 2019 dataset or other relevant datasets containing labeled images of skin lesions.

**Preprocessing:** Implement preprocessing steps such as resizing, normalization, and augmentation to prepare the images for training. Address class imbalances by using techniques such as data augmentation or synthetic data generation.

### 3. Model Development

**Transfer Learning:** Apply MobileNetV2, a pre-trained CNN model, as a feature extractor. Fine-tune the model by training the final layers to adapt to the skin cancer classification task.

**Architecture:** Design and implement a deep learning architecture that includes MobileNetV2 for feature extraction and additional layers for classification.

**Training:** Train the model on the prepared dataset, employing techniques such as early stopping, regularization, and learning rate scheduling to enhance performance.

### 4. Evaluation Metrics

**Accuracy:** Measure the overall classification accuracy of the model.

**Precision, Recall, and F1-Score:** Evaluate the model's performance for each class individually to assess its ability to correctly identify Basal Cell Carcinoma, Melanoma, and Nevus.

**ROC AUC Curve:** Plot and analyze the ROC AUC curve for each class to understand the model's performance in distinguishing between different classes.

**Confusion Matrix:** Generate and analyze the confusion matrix to identify misclassifications and understand the model's strengths and weaknesses.

## 5. Interpretability

Grad-CAM Heatmaps: Use Grad-CAM to visualize which parts of the image the model is focusing on when making predictions. This helps in understanding how the model arrives at its conclusions and ensures transparency.

## 6. Real-World Application

Deployment: Explore potential deployment strategies for integrating the model into clinical settings or diagnostic tools.

Usability: Assess the model's performance and reliability in practical scenarios.

Ensure that it can handle various image qualities and conditions encountered in real-world applications.

## 7. Challenges and Limitations

Data Quality and Variability: Address issues related to image quality and variability in the dataset. Develop strategies to handle noisy or inconsistent data.

Computational Resources: Manage computational resources effectively for training and evaluating the model. Consider cloud-based solutions if necessary.

## 8. Ethical Considerations

Data Privacy: Ensure that patient data is handled in accordance with privacy regulations and ethical standards.

Bias and Fairness: Evaluate the model for any potential biases and ensure that it provides fair and equitable results across different demographic groups.

## 9. Future Work

Model Improvement: Explore advanced techniques such as ensemble methods, additional data sources, or other deep learning architectures to further improve model performance.

Clinical Validation: Plan for clinical trials or collaborations with medical professionals to validate the model's effectiveness in real-world diagnostic settings.

## 7.4 Challenges in Skin Cancer Detection

### 1. Data Imbalance

- Explanation: In your dataset, there might be more images of one type of skin cancer compared to others. For example, you might have many images of Melanoma but fewer images of Basal Cell Carcinoma.
- Challenge: This imbalance can make it harder for the model to learn to correctly identify the less common types of cancer because it might become biased towards the more frequent types.

### 2. Variability in Image Quality

- Explanation: Images of skin lesions can vary greatly in terms of quality, lighting, and resolution. Some images might be blurry, poorly lit, or taken from different angles.
- Challenge: These variations can affect the model's ability to accurately classify images. The model needs to be robust enough to handle these inconsistencies and still make accurate predictions.

### 3. Inter-Class Similarity

- Explanation: Different types of skin cancer can look quite similar, making it challenging for the model to distinguish between them. For example, Nevus and Melanoma might have similar features.
- Challenge: The model needs to learn subtle differences between classes to avoid misclassification. This requires high-quality features and a robust training process.

### 4. Overfitting

- Explanation: Overfitting happens when a model excels on training data but struggles with new, unseen data. This occurs because the model has learned the noise and specific details of the training data that do not apply to other data.
- Challenge: To overcome overfitting, you might need to use techniques like data augmentation, regularization, or cross-validation to ensure the model generalizes well to new images.

### 5. Computational Resources

- Explanation: Training deep learning models, especially on large datasets, can be resource-intensive. It requires significant computational power and

memory.

- Challenge: You need access to powerful hardware or cloud-based resources to efficiently train and test your models. Managing these resources effectively can be a challenge.

## 6. Interpretability

- Explanation: Deep learning models, especially those used in medical applications, are often considered "black boxes" because it's hard to understand how they make their decisions.
- Challenge: It's important to ensure that the model's predictions can be explained in a way that medical professionals can trust. Techniques like Grad-CAM can help visualize what parts of an image the model is focusing on.

## 7. Real-World Validation

- Explanation: Testing the model in a real-world scenario involves dealing with a variety of factors such as different patient demographics, varying image quality, and practical constraints.
- Challenge: Ensuring that the model performs well not just on a controlled dataset but in real-world settings is crucial. This often requires extensive field testing and possibly further adjustments to the model.

## 8. Ethical and Privacy Concerns

- Explanation: Handling medical data involves strict privacy regulations and ethical considerations. Ensuring that patient data is secure and used responsibly is crucial.
- Challenge: You need to comply with regulations like HIPAA (Health Insurance Portability and Accountability Act) and ensure that data privacy and security are maintained throughout the project.



## 7.5 CODE OUTLINE:

### 1. Importing Libraries:

- The code starts by importing necessary libraries like pandas, numpy, cv2, os, tqdm, random, etc. These are used for data handling, image processing, machine learning, and visualization.

### 2. Mounting Google Drive:

- The code mounts Google Drive to access and load image data from specific directories.

### 3. Loading Images:

- Images from directories ('Basal\_cell\_carcinoma', 'Melanoma', 'Nevus') are loaded and displayed as sample images using 'matplotlib.pyplot'.

### 4. Data Preparation:

- Images are loaded, resized to 224x224 pixels, converted to arrays, and appended to 'data'. Corresponding labels (0 for Basal Cell Carcinoma, 1 for Melanoma, 2 for Nevus) are stored in 'labels'.

### 5. Data Randomization:

- Data and labels are shuffled randomly to ensure a balanced distribution in training and testing datasets.

### 6 Train-Test Split:

- The dataset is divided into training and testing sets with an 80:20 ratio. The image data is normalized by dividing pixel values by 255.

### 7. Image Augmentation:

- Techniques such as rotation, zoom, and horizontal shift are used to enhance dataset diversity and improve the model's ability to generalize.

### 8. Model Building:

- Transfer learning is used with MobileNetV2 as a base model. Additional layers are added for classification. The model is compiled with categorical cross-entropy loss and Adam optimizer.

### 9. Training the Model:

- The model is trained using 'fit()' method on augmented data with specified batch

size and epochs. Callbacks like ModelCheckpoint and ReduceLROnPlateau are used for model saving and learning rate adjustment.

#### 10. Saving and Loading the Model:

- Trained model weights are saved and loaded for further evaluation.

#### 11. Evaluation Metrics:

- The model's performance is evaluated using confusion matrix, classification report, and ROC curves for multi-class classification.

#### 12. Testing with New Data:

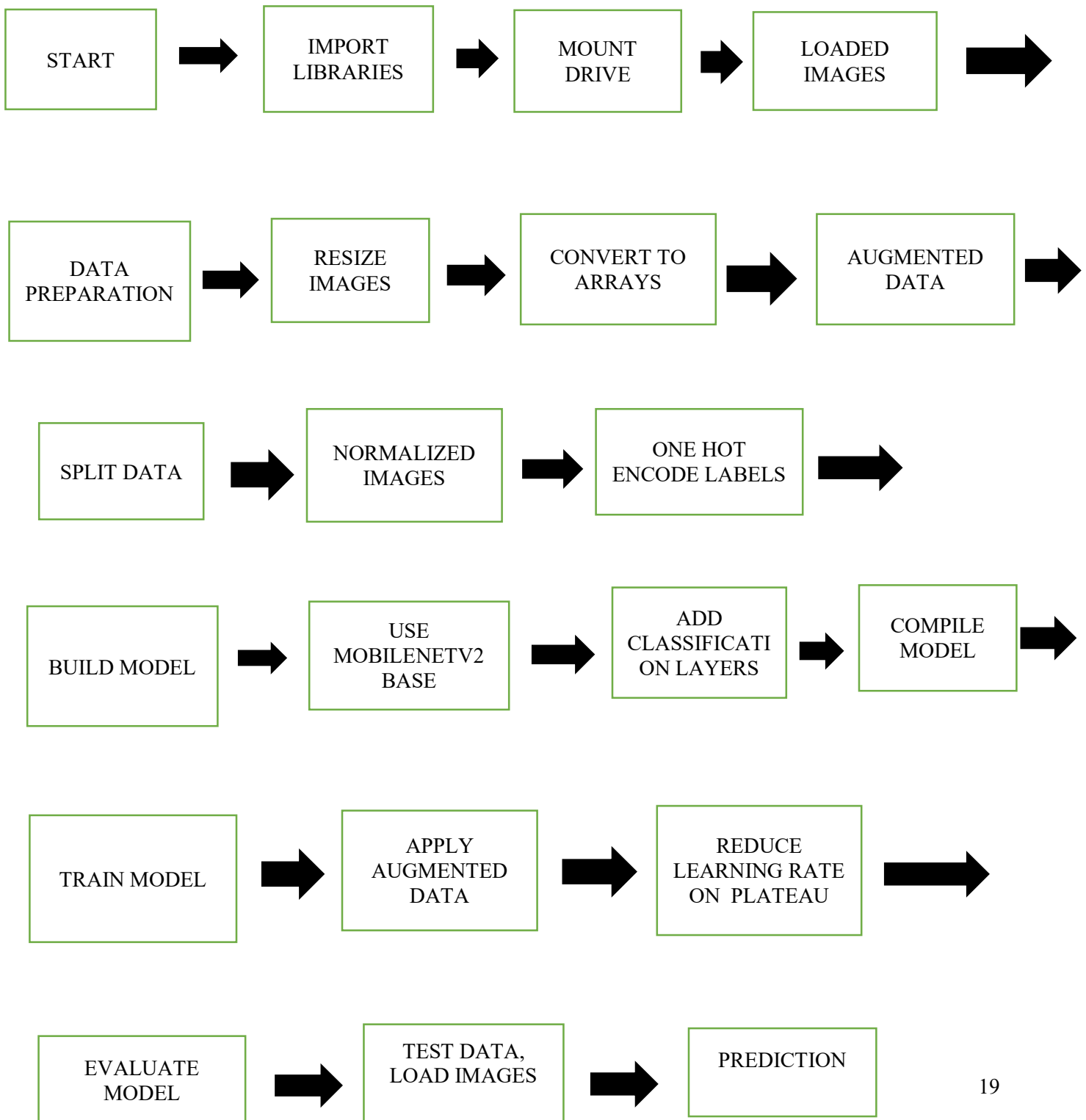
- Test data is prepared similarly to training data and used to predict classes. Images are resized, preprocessed, and predictions are made using the trained model.

#### 13. Prediction on New Images:

- Finally, new images are loaded, preprocessed, and fed into the model for prediction of Basal Cell Carcinoma, Melanoma, or Nevus.

## 7.6 FLOWCHART

This flowchart visualizes the sequential steps involved in training a deep learning model for skin cancer detection, from data loading and preprocessing to model evaluation and prediction. Each step builds upon the previous one, ultimately leading to the classification of new images based on the trained model.



## Chapter 2: Data Handling

### 8.1. Data Collection

- **Overview:**
  - The dataset used for this project comprises skin lesion images from multiple categories, including Basal Cell Carcinoma, Melanoma, and Nevus.
  - Images are collected from a public skin cancer dataset, which provides a substantial amount of labeled data for training, validation, and testing of the model.
- **Method:**
  - **Image Acquisition:** Images are retrieved from a specific directory structure where each class is organized into separate folders.
  - **Preprocessing:** The images are resized to a uniform size (224x224 pixels) and normalized to a range of [0, 1] by dividing pixel values by 255.
- **Tools and Libraries:**
  - Utilized libraries such as cv2 for image loading and PIL for image manipulation.
  - Data is stored and managed using file paths, and processed using libraries like numpy and tensorflow.

### 8.2. Source of Data

- **Dataset Source:**
  - The primary source of the data is the ISIC 2019 dataset, which is publicly available and widely used for skin cancer research.
  - The dataset is stored in Google Drive and accessed via mounted directories for ease of use.
- **Directory Structure:**
  - Images are organized into folders named after their respective classes (e.g., Basal\_cell\_carcinoma, Melanoma, Nevus).
  - Each folder contains test images corresponding to the class labels.

### 8.3. Dataset Description

- **Classes:**
  - **Basal Cell Carcinoma (BCC):** A type of skin cancer that originates from basal cells in the skin. Represented by label 0.

- **Melanoma:** A serious form of skin cancer that originates from melanocytes. Represented by label 1.
- **Nevus:** Non-cancerous skin lesions or moles. Represented by label 2.
- **Image Characteristics:**
  - **Size:** All images are resized to 224x224 pixels to be compatible with the MobileNetV2 model.
  - **Format:** Images are in RGB format.
  - **Data Split:** The dataset is divided into training, validation, and test sets to evaluate the model's performance.
- **Sample Data:**
  - **Number of Images:** Specific numbers of images per class and total dataset size should be mentioned if available.
  - **Example Images:** Provide examples of images from each class to illustrate the dataset.

#### 8.4. Challenges with Data Collection

- **Class Imbalance:**
  - **Issue:** Disparity in the number of images per class can lead to biased model performance.
  - **Solution:** Implement techniques such as data augmentation, resampling, or synthetic data generation to address class imbalances.
- **Image Quality and Variability:**
  - **Issue:** Variations in image quality, lighting conditions, and resolution can affect model performance.
  - **Solution:** Apply preprocessing techniques to standardize image quality and enhance robustness.
- **Data Privacy and Ethical Considerations:**
  - **Issue:** Handling medical data requires adherence to privacy regulations and ethical standards.
  - **Solution:** Ensure that patient data is anonymized and managed according to legal and ethical guidelines.
- **Data Accessibility:**
  - **Issue:** Accessing and managing large datasets can be challenging.
  - **Solution:** Use cloud storage solutions and efficient data handling practices to streamline data access and processing.

## 9. Chapter 3: Setup and Importing Libraries

### 9.1 Importing Required Libraries

In this project, several libraries are used for handling data, pre-processing images, building a neural network model, and evaluating its performance. Below is a description of the main libraries imported and their purposes:

#### 1. TensorFlow:

- **Module:** tensorflow.keras
- **Purpose:** Used for building, training, and saving deep learning models. Specifically, the MobileNetV2 model is imported from TensorFlow's Keras API to leverage transfer learning.
- **Functions/Classes:**
  - MobileNetV2: Pre-trained convolutional neural network model used for feature extraction in the Transfer Learning setup.
  - Model, layers: To customize and build the final model.
  - load\_model: For loading a pre-trained model from disk.
  - preprocess\_input: Preprocessing images to feed them into the network.
  - image: Used for image processing.
  - to\_categorical: For one-hot encoding of class labels.
  - GradientTape: Used for calculating gradients to generate Grad-CAM heatmaps for model explainability.

#### 2. PIL (Python Imaging Library):

- **Module:** PIL.Image
- **Purpose:** Used for opening, manipulating, and converting image files. In this project, it's used to resize images to a format suitable for input into the neural network model.

#### 3. Matplotlib:

- **Module:** matplotlib.pyplot
- **Purpose:** Used for plotting and visualizing the data, model results, and Grad-CAM heatmaps. It's a powerful tool to create static, animated, and interactive visualizations.

#### 4. Numpy:

- **Module:** numpy
- **Purpose:** Numpy is used for efficient manipulation and processing of multi-dimensional arrays, which is essential for handling image data

and neural network outputs. It's also used to normalize image data (i.e., rescaling pixel values).

○

#### 5. **tqdm:**

- **Module:** tqdm
- **Purpose:** Provides a progress bar to track the completion of loops, especially useful for data loading, training iterations, and predictions.

#### 6. **Google Colab Drive:**

- **Module:** google.colab.drive
- **Purpose:** Allows you to mount Google Drive on Google Colab, which is useful for loading the dataset and saving the trained model in a cloud environment. It helps ensure that large datasets and models are efficiently managed.

#### 7. **Sklearn (Scikit-learn):**

- **Module:** confusion\_matrix, classification\_report
- **Purpose:** Provides tools for creating and analyzing classification models. It's used for evaluating the model's performance via confusion matrix and generating classification reports with precision, recall, and F1-score.

#### 8. **Random:**

- **Module:** random
- **Purpose:** Used for shuffling data to avoid any bias in the training/testing phases.

### **Purpose of Using Libraries:**

- **Image Processing:** PIL and TensorFlow's image functions help load and preprocess images into the correct format.
- **Modeling:** TensorFlow and Keras provide the deep learning framework, enabling you to implement the MobileNetV2 model for transfer learning.
- **Data Handling:** Numpy and Random are used to manipulate image arrays and shuffle data.
- **Evaluation and Visualization:** Matplotlib is used to visualize heatmaps and predictions. Sklearn helps in calculating performance metrics.
- **Progress Monitoring:** tqdm is used to display progress bars, making it easier to monitor long-running processes such as loading images.

## Chapter 4: 10. Data Preparation

### 1. Image Loading and Resizing

The images in your dataset are loaded and resized to a fixed size (224x224 pixels), which matches the input size expected by the pre-trained MobileNetV2 model.

Python code:

```
img = Image.open(img_path).resize((224,224))
```

**Purpose:** The MobileNetV2 model, like most convolutional neural networks, requires input images to be of a specific size. Resizing ensures consistency in the data dimensions and allows the model to process the images correctly.

### 2. Image Preprocessing

Once the images are resized, they are converted into arrays and normalized. This step converts the image from its raw format to a format the model can work with.

Python code:

```
img = image.img_to_array(img)
```

```
img = np.expand_dims(img, axis=0)
```

```
img = img.astype('float32') / 255
```

- **Steps:**

- **img\_to\_array:** Converts the image from a PIL format to a NumPy array, which is the required format for deep learning models.
- **np.expand\_dims:** Adds an extra dimension to create a batch, as models expect inputs in batches, even if it's just a single image.
- **Normalization:** The pixel values are scaled to a range of [0, 1] by dividing by 255 (the maximum value for a pixel in an 8-bit image). This improves model performance by making the data more manageable for the network.



### 3. One-Hot Encoding

The class labels are encoded as one-hot vectors. Since the problem involves multi-class classification, the labels need to be represented in a format where each class is mapped to a unique vector.

python code

```
class_dict = {0: "Basal_Cell_Carcinoma (Cancer)",  
              1: "Melanoma (Cancer)",  
              2: "Nevus (Non-Cancerous)"}
```

- **Purpose:** Each label is mapped to a specific class, and the model's output will correspond to one of these classes (i.e., Basal Cell Carcinoma, Melanoma, or Nevus). One-hot encoding ensures that the labels are represented in a form suitable for multi-class classification.

### 4. Data Augmentation

Although data augmentation is not explicitly used in the current version of your code, it could be a useful step. Augmentation involves techniques like rotating, flipping, or zooming images to artificially increase the dataset size. In transfer learning tasks, data augmentation is often used to improve model generalization and avoid overfitting.

### 5. Prediction Preparation

When preparing an image for prediction, similar steps are repeated (resizing, normalization), followed by the model's prediction function.

Python code

```
preds = model.predict(img)[0]
```

- **Purpose:** The predict function outputs probabilities for each class, indicating how confident the model is about its classification. This prediction is then used to determine which class the image most likely belongs to.

## 6. Grad-CAM Heatmap Preparation

For model interpretability, you also use the Grad-CAM (Gradient-weighted Class Activation Mapping) technique to visualize which parts of the image the model is focusing on when making a prediction. The image needs to be preprocessed in a similar way before Grad-CAM can be applied.

**Purpose:** The same preprocessing steps are applied as in the earlier image loading process, ensuring consistency in how images are handled before creating a heatmap that shows the areas the model deems most important.

# Chapter 5 -- 11. Model Building

## 1. Importing Necessary Modules

```
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
```

```
from tensorflow.keras import Model, layers
```

**MobileNetV2:** A compact convolutional neural network design tailored for mobile and embedded vision applications

- **Model and layers:** To build and customize your neural network model.

## 2. Loading the Pre-trained Model

```
conv_base = MobileNetV2(  
    include_top=False,  
    input_shape=(224, 224, 3),  
    weights='imagenet')  
  
for layer in conv_base.layers:  
    layer.trainable = True
```

Downloading data from [https://storage.googleapis.com/tensorflow/keras-applications/mobilenet\\_v2/mobilenet\\_v2\\_weights\\_tf\\_dim\\_ordering\\_tf\\_kernels\\_1.0\\_224\\_no\\_top.h5](https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5)  
9406464/9406464 [=====] - 0s 0us/step

- **include\_top=False:** Excludes the top (fully connected) layers of the pre-trained model. This allows you to add your own classification layers tailored to your dataset.
- **input\_shape=(224, 224, 3):** Specifies the input shape of your images (224x224 pixels with 3 color channels).
- **weights='imagenet':** Loads weights pre-trained on the ImageNet dataset.

## 3. Fine-Tuning the Model

for layer in conv\_base.layers:

layer.trainable = True

- **Purpose:** Sets all layers in the convolutional base to be trainable. This means the weights in these layers will be updated during training.
- **Consideration:** Fine-tuning the entire model can lead to better performance but requires more computational resources and data to avoid overfitting.

## 4. Building Custom Classification Layers

We're adding new layers on top of the convolutional base to adapt the model to your specific classification problem.

```
x = conv_base.output
```

- **conv\_base.output:** The output of the last layer in the pre-trained MobileNetV2 model.

#### 4.1. Global Average Pooling

```
x = layers.GlobalAveragePooling2D()(x)
```

- **Purpose:** This process decreases the spatial dimensions of feature maps by calculating the average value for each feature map, resulting in a single value per map.
- **Benefit:** Reduces the number of parameters and helps prevent overfitting compared to Flattening.

#### 4.2. First Dense Layer

```
x = layers.Dense(256, activation='relu')(x)
```

- **Purpose:** Adds a fully connected layer with 256 neurons and ReLU activation.
- **Function:** Learns complex patterns from the features extracted by the convolutional base.

#### 4.3. First Dropout Layer

```
x = layers.Dropout(0.2)(x)
```

- **Purpose:** Randomly sets 20% of the inputs to zero during training.
- **Benefit:** Helps prevent overfitting by not relying too heavily on any one neuron.

#### 4.4. Second Dense Layer

```
x = layers.Dense(64, activation='relu')(x)
```

- **Purpose:** Adds another fully connected layer with 64 neurons.
- **Function:** Further refines the features for better classification.

#### 4.5. Second Dropout Layer

```
x = layers.Dropout(0.1)(x)
```

- **Purpose:** Randomly sets 10% of the inputs to zero during training.
- **Benefit:** Additional regularization to prevent overfitting.

#### 4.6. Output Layer

`predictions = layers.Dense(3, activation='softmax')(x)`

- **Purpose:** Final classification layer with 3 neurons corresponding to your 3 classes (BCC, Melanoma, Nevus).
- **Activation:** Softmax function to output probabilities for each class.

#### 5. Creating the Final Model

Python code

`model = Model(conv_base.input, predictions)`

- **Purpose:** Defines the model by specifying the inputs and outputs.
- **Structure:** Combines the pre-trained convolutional base with your custom classification layers.

#### 6. Setting Up Callbacks

```
] callbacks = [ModelCheckpoint('.mdl_wts.hdf5', monitor='val_loss', mode='min', verbose=1, save_best_only=True),
               ReduceLROnPlateau(monitor='val_loss', factor=0.3, patience=2, verbose=1, mode='min', min_lr=0.0000000001)]
```

- **ModelCheckpoint:** Saves the model after every epoch if there's an improvement in validation loss.
  - **Parameters:**
    - `monitor='val_loss'`: Monitors the validation loss.
    - `mode='min'`: Looks for a decrease in the monitored value.
    - `save_best_only=True`: Saves only the model with the best performance.
- **ReduceLROnPlateau:** Reduces the learning rate when the validation loss plateaus.
  - **Parameters:**
    - `factor=0.3`: Reduces the learning rate by a factor of 0.3.
    - `patience=2`: Waits for 2 epochs before reducing the learning rate.
    - `min_lr=1e-11`: Sets a lower bound on the learning rate.

#### 7. Compiling the Model

- ```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
BS = 64
```

- **Loss Function:** categorical\_crossentropy is used for multi-class classification.
- **Optimizer:** adam optimizer is chosen for its efficiency and adaptive learning rate capabilities.
- **Metrics:** Tracks the accuracy during training and validation.

## 8. Training the Model

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
BS = 64
print("[INFO] training head...")
H = model.fit(
    trainAug.flow(x_train,y_train, batch_size=BS),
    steps_per_epoch=train_len // BS,
    validation_data=(x_test, y_test),
    validation_steps=test_len // BS,
    epochs=30,callbacks=callbacks)
```

- [INFO] training head...
- 
- **BS (Batch Size):** Number of samples processed before the model is updated.
- **Data Augmentation:** trainAug is an ImageDataGenerator that applies random transformations to the training images to improve model generalization.
- **Parameters:**
  - steps\_per\_epoch: Number of batches per epoch.
  - validation\_data: Tuple (x\_test, y\_test) for validation.
  - epochs=30: Trains the model for 30 epochs.
  - callbacks: Uses the callbacks defined earlier for saving the best model and adjusting the learning rate.

## 9. Training Output

- The training output shows the loss and accuracy for each epoch on both the training and validation sets.
- **Learning Rate Adjustments:** The ReduceLROnPlateau callback reduces the learning rate when the validation loss stops improving.
- **Model Checkpoints:** The ModelCheckpoint callback saves the model whenever there's an improvement in validation loss.

## 10. Saving and Loading the Model

Python code

```
model.save('/content/drive/My Drive/skin_model/model_v1.h5')
```

- **Purpose:** Saves the trained model to a file for later use.

Python code

```
model = load_model('/content/drive/My Drive/skin_model/model_v1.h5')
```

- **Purpose:** Loads the saved model for evaluation or further predictions.

## 11. Evaluating the Model

Python code

```
accuracy = model.evaluate(x_test, y_test, verbose=1)
```

```
print('\n', 'Test_Accuracy:-', accuracy[1])
```

- **Purpose:** Evaluates the model on the test data to determine its performance.
- **Output:** Prints the test accuracy, which in your case is approximately **90.81%**.

## Model Compilation:

The model is compiled with the specified optimizer (Adam) and loss function (binary\_crossentropy). Metrics such as accuracy are also tracked during training.

**-Training Epochs:** The training proceeds over 30 epochs (Epoch 1 to Epoch 30), with each epoch consisting of 154 batches. Each batch appears to be processed in approximately 99-132 seconds.

### -Training Progress:

- **Loss and Accuracy:** The training starts with a loss of 0.5126 and accuracy of 0.7884 in the first epoch, gradually improving.
- **Validation:** The validation loss and accuracy (val\_loss and val\_accuracy) are also monitored. Early epochs show high val\_loss, indicating potential overfitting or unstable training.
- **Learning Rate:** The learning rate (lr) is initially 0.001 and is reduced dynamically by a factor of 0.3 (ReduceLROnPlateau callback) when the validation loss plateaus.

**-Model Checkpoints:** The best model weights (model.save()) based on val\_loss are saved in the file .mdl\_wts.hdf5.

### -Validation Performance:

- The validation loss (val\_loss) improves significantly over epochs, indicating that the model generalizes better with training.
- Validation accuracy (val\_accuracy) also improves steadily, reaching 0.9081 in the final epoch.



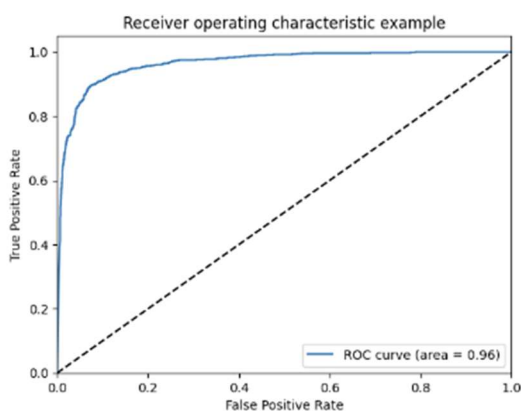
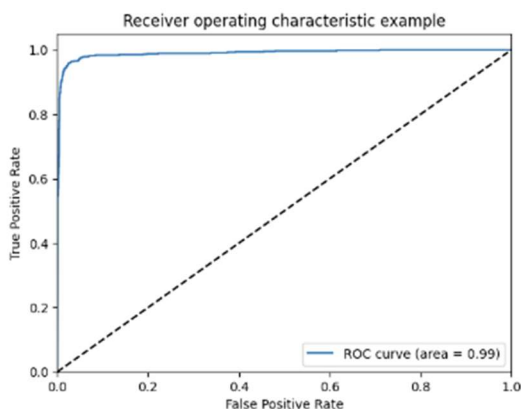
## 12. ROC AUC CURVE:

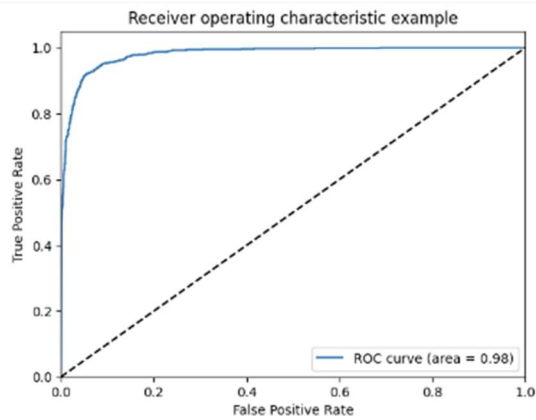
The ROC (Receiver Operating Characteristic) curve is a visual tool used to assess the performance of a classification model at different threshold levels. It displays two key metrics:

1. True Positive Rate (TPR): Also known as sensitivity or recall, TPR indicates the percentage of actual positives that the model correctly identifies.
2. False Positive Rate (FPR): FPR shows the percentage of actual negatives that the model incorrectly classifies as positives.

The ROC curve is useful for understanding the balance between TPR and FPR.

- $AUC = 1$ : The model perfectly separates the classes.
- $AUC = 0.5$ : The model's performance is equivalent to random guessing.
- $AUC < 0.5$ : The model is performing worse than random chance.

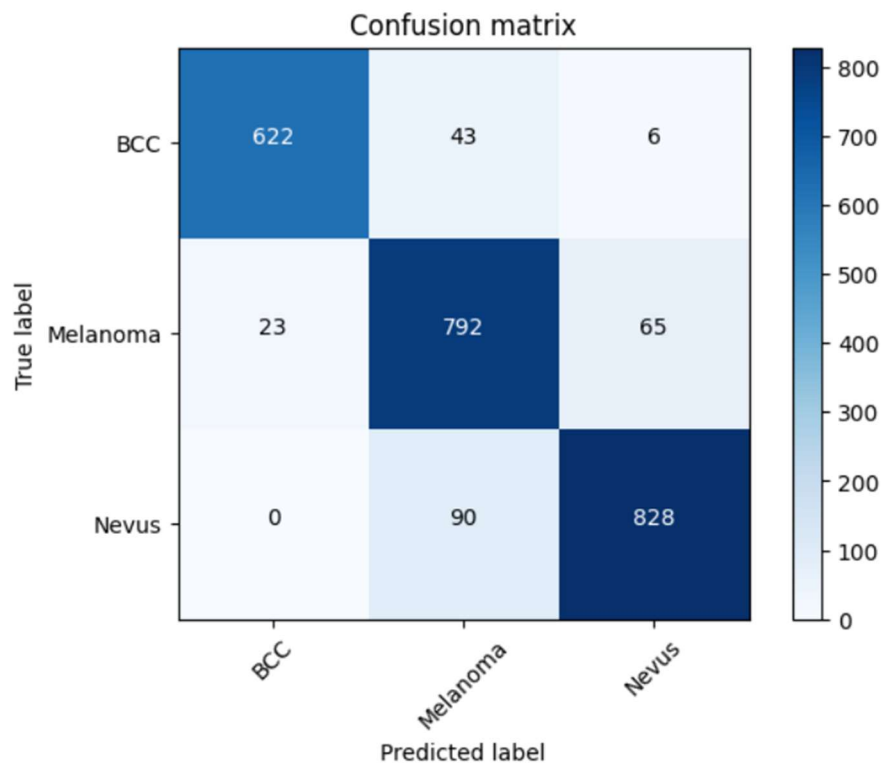




### Prediction Results:

- **Image 1 (Nevus):**
  - **Predicted Class:** Nevus (Non-Cancerous)
  - **Confidence:** 100% Nevus, 0% BCC, 0% Melanoma
- **Image 2 (Melanoma):**
  - **Predicted Class:** Melanoma (Cancer)
  - **Confidence:** 69.14% Melanoma, 30.85% Nevus, 0.01% BCC
- **Image 3 (Basal Cell Carcinoma):**
  - **Predicted Class:** Basal Cell Carcinoma (Cancer)
  - **Confidence:** 98.95% BCC, 0.94% Melanoma, 0.11% Nevus

### 13. CONFUSION MATRIX:



#### **BCC:**

- Correctly classified as BCC: 622 instances.
- Misclassified as MELANOMA: 43 instances.
- Misclassified as NEVUS: 6 instances.

#### **MELANOMA:**

- Correctly classified as MELANOMA: 792 instances.
- Misclassified as BCC: 23 instances.
- Misclassified as NEVUS: 65 instances.

#### **NEVUS:**

- Correctly classified as NEVUS: 828 instances.
- Misclassified as MELANOMA: 90 instances.

## 14.CLASSIFICATION REPORT

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| BCC          | 0.96      | 0.93   | 0.95     | 671     |
| Melanoma     | 0.86      | 0.90   | 0.88     | 880     |
| Nevus        | 0.92      | 0.90   | 0.91     | 918     |
| accuracy     |           |        | 0.91     | 2469    |
| macro avg    | 0.91      | 0.91   | 0.91     | 2469    |
| weighted avg | 0.91      | 0.91   | 0.91     | 2469    |

### 1. Precision

- Precision refers to the proportion of true positive predictions out of all positive predictions. It measures how many of the predicted positive cases are actually positive.
- Precision for BCC: 96% (96% of instances predicted as BCC were truly BCC)
- Precision for Melanoma: 86% (86% of instances predicted as Melanoma were truly Melanoma)
- Precision for Nevus: 92% (92% of instances predicted as Nevus were truly Nevus)

### 2. Recall (Sensitivity)

- Recall represents the proportion of true positive cases out of the total actual positive cases. It shows how well the model identifies positive instances.
- Recall for BCC: 93% (93% of actual BCC cases were predicted correctly)
- Recall for Melanoma: 90% (90% of actual Melanoma cases were predicted correctly)
- Recall for Nevus: 90% (90% of actual Nevus cases were predicted correctly)

### 3. F1-score

- The F1-score is the harmonic mean of precision and recall, offering a balance between the two metrics.
- F1-score for BCC: 95%
- F1-score for Melanoma: 88%
- F1-score for Nevus: 91%

### 4. Accuracy

- Accuracy measures the overall proportion of correctly predicted instances, including both true positives and true negatives.

- Overall accuracy: 91% (91% of all instances were classified correctly)

## 5. **Macro average**

Macro average calculates the average of recall, precision, and F1-score across all classes, giving each class equal weight regardless of its size."

## 6. **Weighted average**

- Weighted average calculates the precision, and F1-score and recall, taking into account the number of instances in each class, giving more importance to classes with more data.

## **Interpretation:**

- **High Scores:** The model demonstrates strong performance across all metrics (precision, recall, and F1-score) for all categories (BCC, Melanoma, Nevus), with most scores above 86%.
- **Accuracy:** The model achieves an overall accuracy of 91%, meaning it correctly classifies 91% of all instances.

## 15. Conclusion:

The model appears to be training well, with decreasing loss and increasing accuracy over epochs.

Accuracy we get- 90.8%

The use of learning rate reduction helps stabilize training and improve validation performance.

The classification report indicates that the model is performing well for the classification task across multiple classes (BCC, Melanoma, Nevus), with high precision, recall, F1-scores, and accuracy.

It provides a comprehensive overview of how well the model distinguishes between different types of skin lesions based on the given evaluation metrics.

## 16. Literature References

The following are referred journals from the preliminary literature review.

- <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8705277/#:~:text=AI%20can%20be%20of%20use,better%20outcomes%20of%20skin%20cancer>
- <https://ieeexplore.ieee.org/document/10051378>
- <https://ieeexplore.ieee.org/document/8641762>
- <https://jmai.amegroups.org/article/view/8241/html>

### Check list of items for the Final report

|      |                                                                      |   |
|------|----------------------------------------------------------------------|---|
| a)   | Is the Cover page in proper format?                                  | Y |
| b)   | Is the Title page in proper format?                                  | Y |
| c)   | Is Certificate from the Supervisor in proper format? it been signed? | Y |
| d)   | Is Abstract included in the Report? Is it properly written?          | Y |
| e)   | Does the Table of Contents page include chapter page numbers?        | Y |
| f)   | Does the Report contain a summary of the literature survey?          | Y |
| i.   | Are the Pages numbered properly?                                     | Y |
| ii.  | Are the Figures numbered properly?                                   | Y |
| iii. | Are the Tables numbered properly?                                    | Y |
| iv.  | Are the Captions for the Figures and Tables proper?                  | Y |
| v.   | Are the Appendices numbered?                                         | Y |
| g)   | Does the Report have Conclusion / Recommendations of the work?       | Y |
| h)   | Are References/Bibliography given in the Report?                     | Y |
| i)   | Have the References been cited in the Report?                        | Y |
| j)   | Is the citation of References / Bibliography in proper format?       | Y |



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**  
**Work Integrated Learning Programmes Division**  
**II SEMESTER 23-24**

**DSECLZG628T / AIMLCZG628T DISSERTATION**

**(Final Evaluation Sheet)**

*Scheduled Month : June*

**NAME OF THE STUDENT: SHREYA KSHIRSAGAR**

**ID NO. : 2022AA05455**

**Email Address : 2022aa05455@wilp.bits-pilani.ac.in**


**NAME OF SUPERVISOR : Vinaya Sathyanarayana**

**PROJECT TITLE : Skin cancer detection**

| S.No. | Criteria                          | Excellent | Good | Fair | Poor |
|-------|-----------------------------------|-----------|------|------|------|
| 1     | Work Progress and Achievements    | Yes       |      |      |      |
| 2     | Technical/Professional Competence | Yes       |      |      |      |
| 3     | Documentation and expression      | Yes       |      |      |      |
| 4     | Initiative and originality        | Yes       |      |      |      |
| 5     | Punctuality                       | Yes       |      |      |      |
| 6     | Reliability                       | Yes       |      |      |      |
|       | Recommended Final Grade           | A         |      |      |      |

**EVALUATION DETAILS**

| EC No. | Component            | Weightage | Marks Awarded |
|--------|----------------------|-----------|---------------|
| 1      | Dissertation Outline | 10%       | 10%           |
| 2      | Mid-Sem Progress     |           | 30%           |
|        | Seminar              | 10%       |               |
|        | Viva                 | 5%        |               |
|        | Work                 | 15%       |               |
|        | Progress             |           |               |
| 3      | Final Seminar/Viva   | 20%       | 20%           |
| 4      | Final Report         | 40%       | 40%           |
|        | Total out of         | 100%      | 100%          |

|                                  |                                                                                   |  |
|----------------------------------|-----------------------------------------------------------------------------------|--|
|                                  | <b>Organizational Mentor</b>                                                      |  |
| <b>Name</b>                      | Sarang Ashtankar                                                                  |  |
| <b>Qualification</b>             | BE                                                                                |  |
| <b>Designation &amp; Address</b> | Associate Manager,<br>Delhi India                                                 |  |
| <b>Email Address</b>             | ashtankar.sarang5@gmail.com                                                       |  |
| <b>Signature</b>                 |  |  |
| <b>Date</b>                      | 9 <sup>th</sup> September 2024                                                    |  |