MANGALORE UNIVERSITY

Project Report On

"SKIN LESION SEGMENTATION AND CLASSIFICATION USING U-NET + RESNET"

Carried out and submitted

To

DEPARTMENT OF POST-GRADUATE STUDIES AND RESEARCH IN COMPUTER SCIENCE

In Partial Fulfilment for the Award of Degree in Master of Computer Science during the Academic Year 2024-2025

Submitted By:

BHAWYA DEVI

P05AZ23S038006

MSc Computer Science

Under the Guidance of:

Dr. B. H. SHEKAR

Senior Professor

Department of Computer Science

Mangalore University

Department of Post-Graduate Studies and Research in Computer Science

Mangalore University

Mangalagangothri, Mangalore – 574 199

MANGALORE UNIVERSITY

DEPARTMENT OF POST-GRADUATE STUDIES AND RESEARCH IN COMPUTER SCIENCE

Mangalore University
Mangalagangothri, Mangaluru – 574 199

CERTIFICATE

This is to certify that the project work entitled **"SKIN LESION SEGMENTATION AND CLASSIFICATION USING U-NET + RESNET"** has been successfully carried out in the Department of Post- Graduate Studies and Research in Computer Science by **Ms. BHAWYA DEVI (Reg. No: P05AZ23S038006),** student of fourth semester Master of Computer Science, under the supervision and guidance of **Dr. B.H SHEKAR,** Senior Professor, Department of Post-Graduate Studies and Research in Computer Science, Mangalore University. The project is partial fulfilment of the requirements for the award of **Master of Computer Science** by **Mangalore University** during the academic year **2024-2025**

Internal Guide

Internal Examiner **External Examiner**

Submitted for the viva-voce examination held on _____

DECLARATION

This Project work entitled "**SKIN LESION SEGMENTATION AND CLASSIFICATION USING U-NET + RESNET**" has been successfully carried out by me under the supervision and guidance of **Dr. B.H SHEKAR,** Senior Professor**,** Department of Post-Graduate Studies and Research in Computer Science, Mangalore University, Mangalagangothri. This project is submitted in partial fulfilment for the award of **Master of Computer Science** degree by **Mangalore University** during the academic year 2024-25. This work or any part of this work has not been submitted to any other University or Institute/School for the award of any other Degree or Diploma.

Date: Name: Bhawya Devi

Place: Mangalagangothri Reg No: P05AZ23S038006

ACKNOWLEDGEMENT

I take this opportunity to express my sincere thanks to my supervisor **Dr. B.H SHEKAR**, Senior Professor, Department of Post-Graduate Studies and Research in Computer Science, Mangalore University, for providing me this project and other relevant infrastructures work in the Department, for her all-round guidance, timely help at every stage of this project and for the valuable suggestions and unlimited support.

I would like to express my gratitude to **Dr. MANJAIAH D.H**, Senior Professor and Chairperson Department of Post-Graduate Studies and Research in Computer Science, Mangalore University, for according persimmon and for providing all kinds of infrastructure facilities in the department to carry out this project successfully.

I express my sincere thanks to **Dr. H. L. SHASHIREKHA**, Professor and **Mr. PRAKASHA M**., Assistant Professor, Department of Post-Graduate Studies and Research in Computer Science.

I would like to express my heartfelt gratitude to **Ms. Sharal Coelho**, my parents, friends, and well-wishers who have always inspired and blessed me, including those whom I may have inadvertently failed to mention.

Last but not the least; it gives me immense pleasure and a great sense of fulfilment to thank all those who have directly and indirectly helped me in the completion of my project work.

Above all, with all my heart, I thank you God for empowering me with the dedication, focus and patience to carry out this project successfully.

BHAWYA DEVI

ABSTRACT

Skin cancer is one of the most common and potentially deadly forms of cancer worldwide. Early detection and accurate diagnosis of skin lesions play a critical role in improving patient outcomes. This project focuses on developing an automated system for skin lesion segmentation and classification leveraging deep learning techniques, specifically the U-Net architecture for segmentation and ResNet for classification. The segmentation step aims to accurately delineate the lesion area from dermoscopic images, which is essential for extracting meaningful features and improving classification accuracy.

The dataset used includes dermoscopic images categorized into eight distinct classes of skin lesions: Melanoma (MEL), Melanocytic Nevus (NV), Basal Cell Carcinoma (BCC), Actinic Keratosis / Bowen's Disease (AKIEC), Benign Keratosis (BKL), Dermatofibroma (DF), Vascular Lesion (VASC), and Squamous Cell Carcinoma (SCC). These classes cover a wide spectrum of both benign and malignant lesions, making the classification task comprehensive and challenging.

The U-Net model is employed for the segmentation task due to its proven effectiveness in medical image segmentation, enabling precise localization and boundary identification of lesions. Separately, the ResNet model, a deep residual neural network known for its powerful feature extraction capabilities and robustness against vanishing gradients, is used for the classification task, categorizing lesions into one of eight predefined classes. Both tasks are performed independently, allowing for focused optimization of segmentation and classification models. The system incorporates preprocessing steps such as data augmentation to address class imbalance and improve model generalization.

A web-based platform named DermaCare has been developed to deploy the classification model. Users can upload dermoscopic images and receive instant predictions of the lesion type. The intuitive interface ensures ease of use for both medical professionals and non-experts, bridging the gap between research and practical application as a decision-support tool in dermatology.

In summary, this work demonstrates the feasibility and effectiveness of using U-Net for segmentation and ResNet for multi-class skin lesion classification as separate yet complementary components, contributing to the advancement of computer-aided diagnosis systems in dermatology.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1. INTRODUCTION

1.1. Overview

Skin cancer is one of the most prevalent forms of cancer worldwide, with its incidence steadily rising due to factors such as prolonged exposure to ultraviolet (UV) radiation, genetic susceptibility, and environmental influences. Early detection and accurate diagnosis are critical for improving patient survival rates and enabling timely treatment. However, manual diagnosis through visual inspection or dermoscopic examination can be subjective and dependent on the expertise of the dermatologist, leading to potential variations in interpretation. To address these challenges, computer-aided diagnosis (CAD) systems using deep learning have emerged as promising solutions.

This project, titled **"Skin Lesion Segmentation and Classification Using U-Net + ResNet"**, focuses on implementing two distinct but related tasks: **skin lesion segmentation** and **skin lesion classification**. In the segmentation task, the U-Net architecture is employed due to its proven effectiveness in biomedical image segmentation. U-Net enables precise localization of lesion boundaries, producing binary masks that clearly separate lesion regions from surrounding skin. This aids in quantitative analysis and facilitates better visualization for clinical use.

In the classification task, a ResNet50-based model is utilized to categorize dermoscopic images into multiple lesion types, including both malignant and benign classes. The model is trained on benchmark datasets such as ISIC 2018 and ISIC 2019, incorporating preprocessing techniques like hair removal, contrast enhancement, and image resizing to improve input quality. Class imbalance is addressed through data augmentation and balancing strategies to enhance the model's generalization capability.

By performing segmentation and classification as separate processes, the project aims to develop specialized, optimized models for each task, ensuring high accuracy and reliability. The outcomes have the potential to support dermatologists in clinical decision-making, improve diagnostic efficiency, and extend skin lesion screening capabilities to remote and resource-limited healthcare settings.

1.1.2. Objectives

The primary objectives of this project are:

1. **To apply effective image preprocessing techniques** such as hair removal, contrast enhancement, and resizing to improve model input quality.
2. **To address dataset imbalance** through augmentation and balancing strategies to enhance model generalization.
3. **To develop a deep learning-based segmentation model** using the U-Net architecture for precise identification and boundary delineation of skin lesions from dermoscopic images.
4. **To implement a classification model** using the ResNet50 architecture for accurately categorizing skin lesions into multiple predefined classes.
5. **To evaluate model performance** using standard metrics such as accuracy, precision, recall, F1-score, and AUC for classification, and Dice coefficient or IoU for segmentation.

1.1.3. Statement of the Problem

Early detection of skin lesions is vital for preventing skin cancer progression, yet manual diagnosis is time-consuming, subjective, and reliant on dermatologist expertise, often leading to inconsistencies. Limited access to specialists further delays diagnosis, highlighting the need for automated, accurate, and efficient systems for lesion segmentation and classification to support timely and reliable medical decisions.

1.1.4. Motivation

Skin cancer rates are increasing globally, making early detection critical for improving survival rates. Manual diagnosis can be subjective and resource-intensive, particularly in regions with limited access to dermatologists. Advances in deep learning offer an opportunity to develop automated systems that can perform accurate skin lesion segmentation and classification, supporting timely diagnosis and reducing the burden on healthcare professionals.

1.1.5. Challenges

1. **Visual Similarity Between Lesions**

   Different types of skin lesions often share similar visual characteristics such as color, shape, and texture. For instance, early-stage melanoma can closely resemble benign nevi. This visual overlap makes it challenging for even advanced models to differentiate between classes accurately, potentially leading to misclassification**.**

2. **Class Imbalance in Dataset**

   The ISIC datasets have an uneven distribution of lesion types. Some classes, like Melanocytic Nevus (NV), have thousands of samples, while others, such as Dermatofibroma (DF) or Vascular Lesions (VASC), contain only a few hundred. This imbalance can bias the model to favor majority classes during training, resulting in poor performance on minority classes.

3. **Challenges in Lesion Segmentation Due to Irregular Shapes**

   Skin lesions typically do not have well-defined shapes or boundaries. Their irregular and complex contours make segmentation using traditional image processing techniques very difficult. This complexity demands more sophisticated, learning-based segmentation methods.

4. **High Computational Requirements**

   Deep learning models like U-Net and ResNet are computationally intensive. Training them on large image datasets requires high-end GPUs, substantial RAM, and long training times. This may be a limiting factor for researchers and institutions with limited hardware resources.

5. **Need for Advanced Class Balancing Techniques**

   Because of the data imbalance, models may become biased toward majority classes. Therefore, strategies like class reweighted loss functions and synthetic data generation (e.g., via GANs) are essential to ensure fair learning across all lesion types. However, implementing these techniques is time consuming.

6. **Variability in Image Quality**

   Dermoscopic images can vary widely due to differences in devices, lighting conditions, image resolution, and the presence of artifacts such as hair, ink markings, or reflections. These inconsistencies introduce noise and reduce the quality of data, which in turn affects model performance during both segmentation and classification.

1.1.6. Application

- **Early Melanoma Detection :** The system can help dermatologists and general physicians detect melanoma at an early stage by analyzing skin images for suspicious lesions. Early detection increases treatment success and patient survival rates.
- **Monitoring Treatment Response**: Skin lesion detection and classification algorithms can be used to monitor the response of a lesion to treatment over time. This helps dermatologists adjust treatment plans as needed and track improvements or changes in the lesion.
- **Clinical Research:** Skin lesion detection algorithms can analyze large datasets of skin images to identify new lesion subtypes, patterns, or risk factors. This information can contribute to research on better diagnostic tools, treatments, and understanding of skin cancer progression.
- **Educational Tool for Medical Students :** The system can serve as a learning aid, allowing students to explore various skin lesion types, understand characteristics of melanoma vs benign lesions, and practice diagnosis in a controlled environment.

1.1.7. Organization Of Report

- Chapter 1 gives the brief introduction of Skin Lesion classification and segmentation using U-Net + ResNet model, its Application, Problem Statement, Challenges of the system and Motivation.
- Chapter 2 contains literature survey that provide summary of individual paper.
- Chapter 3 provides information about hardware & software requirements.
- Chapter 4 provides overview of existing work for Skin Lesion segmentation and Classification that has done by using Deep Learning methods.
- Chapter 5 presents experiment, its results and technology used to achieve this and dataset details.
- Chapter 6 contains conclusion and future work about Skin Lesion classification using Deep Learning Methods.

CHAPTER 2

LITERATURE SURVEY

2. LITRATURE REVIEW

- **Duan Wang, "Skin Lesion Segmentation of Dermoscopy Images Using U-Net", Beijing University of Posts and Telecommunications, June 2023.**

  In the June 2023 paper titled *"Skin Lesion Segmentation of Dermoscopy Images Using U-Net"* from Beijing University of Posts and Telecommunications, Duan Wang presents a streamlined and effective approach for segmenting skin lesions using the U-Net architecture The proposed system comprises two key components: a **Skin Image Analysis Module**, which handles learning and validation using training data, and a **Skin Image Segmentation Module**, which processes new dermoscopic images to produce segmentation masks Training was conducted using 100 ISIC dataset images over 10 epochs on a GPU, yielding a training accuracy of 0.9085 and validation accuracy of 0.9536 The system allows users to upload lesion images into a designated folder and receive fast, reliable segmentation results—an important feature for enhancing diagnostic workflows and potentially improving patient outcomes Overall, Wang's work demonstrates U-Net's capability to deliver high-precision lesion segmentation with moderate training resources and practical usability.

- **Lina Liu, Lichao Mou, Xiao Xiang Zhu, Mrinal Manda, "Skin Lesion Segmentation Based on Improved U-net" IEEE Access, 11 October 2019.**

  Lina Liu, Lichao Mou, Xiao Xiang Zhu, and Mrinal Mandal (2019) propose an **Improved U-Net** architecture for automatic skin lesion segmentation, presented at the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). Citing the persistent challenge of melanoma segmentation due to high visual complexity and ambiguity in dermoscopic images, the authors enhance the standard U-Net framework to achieve state-of-the-art performance using the ISIC 2017 melanoma detection dataset.

  Their improvements include adopting dilated convolutions within the U-Net architecture to expand receptive fields without increasing model complexity, thereby improving segmentation accuracy.The modified architecture retains U-Net's encoder–decoder structure but enhances its ability to capture broader context and finer details—critical for delineating irregular lesion boundaries. Experimental results indicate that this model outperforms the conventional U-Net baseline, marking a meaningful advancement in robust, deep learning-based skin lesion segmentation.

- **Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", arXiv preprint arXiv:1512.03385, 10 December 2015.**
  In this paper, Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun introduced the Residual Network (ResNet), a breakthrough architecture that enabled the successful training of substantially deeper neural networks by reformulating layers to learn **residual functions** relative to their inputs.This innovative residual learning framework, featuring identity skip connections, addressed the "degradation problem"—where deeper networks exhibited higher training error—and significantly improved both optimization and accuracy. ResNet models with depths up to **152 layers** were shown to outperform shallower architectures like VGG, achieving a remarkable **3.57% error** on the ImageNet test set, thereby securing first place in the ILSVRC 2015 competition. The framework also demonstrated broad generalizability, contributing to top performance in COCO object detection and segmentation challenges, with a reported **28% relative improvement** over previous methods. ResNet's identity mappings facilitate efficient signal propagation across layers, both forward and backward, which enhances convergence and mitigates vanishing gradients.
  This paper fundamentally transformed deep learning by demonstrating that ultra-deep networks—once deemed impractical—could indeed be trained effectively, spawning widespread adoption of residual connections in modern computer vision and beyond.

- **Huthaifa Abuhammad, Mohammad Alhawarat, Duaa Mehiar, "Soft Attention-Enhanced ResNet101 for Robust Skin Lesion Classification in Dermoscopic Images," 2025 International Conference for Artificial Intelligence, Applications, Innovation and Ethics (AI2E), February 2025.**
  Abuhammad, Alhawarat, and Mehiar (2025) propose an improved ResNet101 architecture enhanced with a soft attention mechanism for robust skin lesion classification in dermoscopic images. The integration of soft attention allows the model to focus selectively on the most informative regions of skin lesions, improving feature extraction and discrimination among multiple lesion classes. This approach addresses challenges such as lesion variability, complex backgrounds, and subtle texture differences often found in dermoscopic images.
  The model was trained and evaluated on benchmark datasets, demonstrating improved classification accuracy and robustness compared to baseline ResNet101 models without

attention. The use of soft attention helped reduce misclassification rates, particularly for visually similar lesion types. This work highlights the effectiveness of combining deep residual networks with attention mechanisms to enhance automated skin lesion diagnosis, potentially supporting clinical decision-making with higher reliability.

- **Mehwish Zafar, Muhammad Imran Sharif, Muhammad Irfan Sharif, Seifedine Kadry, Syed Ahmad Chan Bukhari, Hafiz Tayyab Rauf, "Skin Lesion Analysis and Cancer Detection Based on Machine/Deep Learning Techniques: A Comprehensive Survey."**

  Zafar et al. conducted a thorough survey on the application of machine learning and deep learning techniques for skin lesion analysis and cancer detection. The study reviews various methodologies for segmentation, feature extraction, and classification of dermoscopic images, ranging from classical approaches to recent deep learning architectures. Special attention is given to convolutional neural networks (CNNs), which have demonstrated superior performance in feature learning and classification accuracy compared to traditional methods. The authors discuss challenges faced in this domain, including dataset imbalance, presence of artifacts such as hair and bubbles, and variability in lesion appearance. Publicly available datasets like ISIC and HAM10000 are also analyzed in terms of their suitability for training and evaluation. The survey concludes with suggestions for future research focusing on improving model robustness, interpretability, and integration into clinical workflows, highlighting the significant potential of AI-driven tools in aiding early diagnosis and treatment of skin cancer.

CHAPTER 3
HARDWARE & SOFTWARE REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

- Processor : Intel Core I3
- Hard Disk : 500 GB
- Ram : 8 GB

3.2 SOFTWARE REQUIREMENT

- Operating System : Window 7 And Above
- Coding Language : Python
- Framework : Google Colab

CHAPTER 4

METHODOLOGY

4.1. GENERAL FRAMEWORK OF PROPOSED SYSTEM

Our proposed system for automated skin lesion analysis consists of five main modules: dataset acquisition, preprocessing, data splitting, model training, and classification. Dermoscopic images are collected from datasets like ISIC 2018 and 2019 and undergo preprocessing steps such as hair removal, contrast enhancement, and resizing. The data is then split into training and testing sets, usually with an 80:20 ratio. Two separate deep learning models are trained independently—U-Net for lesion segmentation and ResNet50 and ResNet101 for multiclass lesion classification. Both models are trained over several epochs, and their performance is evaluated using metrics like Dice coefficient for segmentation and accuracy for classification.

Architecture Of Proposed System

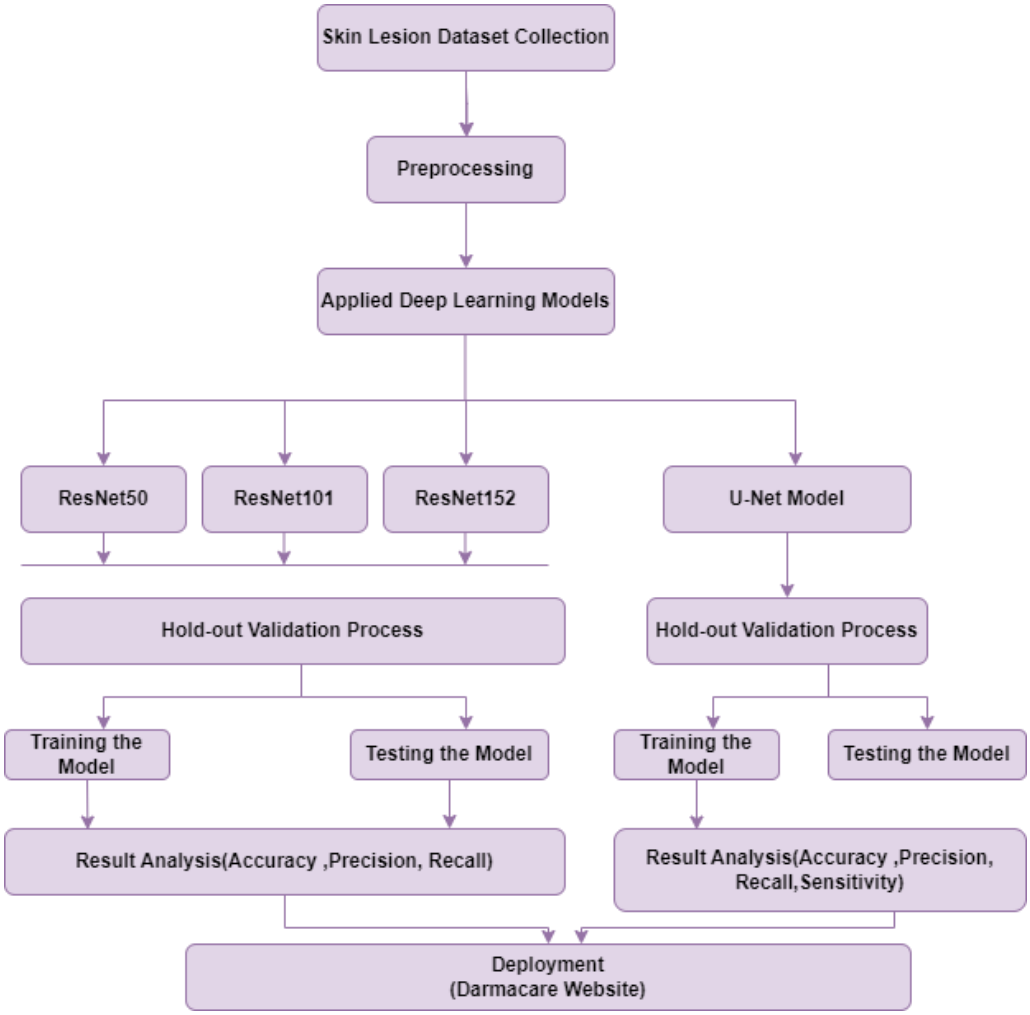


Fig 4.1 Architecture Of Proposed System

4.2.1. Working with a Dataset

The dataset used in this project consists of dermoscopic images sourced from both the ISIC 2018 and ISIC 2019 challenges. The ISIC 2018 dataset, comprising approximately 3693 images, provides expert-annotated **segmentation masks** alongside the images, making it ideal for training and evaluating lesion segmentation models.

On the other hand, the ISIC 2019 dataset contains around 25,331 images categorized into eight classes for **multiclass skin lesion classification**: Melanoma, Basal Cell Carcinoma, Benign Keratosis-like Lesions, Dermatofibroma, Vascular Lesions, Squamous Cell Carcinoma, Actinic Keratosis, and Nevus. Both datasets reflect diverse lesion types, skin tones, and imaging conditions, closely simulating real-world clinical scenarios. Notably, there is significant **class imbalance** in the 2019 dataset, with some lesion classes having many more samples than others. To mitigate this, strategies like class weighting and oversampling are applied during model training. The datasets are carefully split into training, validation, and testing subsets, maintaining representative class distributions to ensure robust model evaluation.

URL : https://challenge.isic-archive.com/data/

The number of images obtained for different lesion types in ISIC 2019 dataset are:

Type	Training Sample	Validation Sample	Testing Sample	Total
Melanoma	3158	664	700	4522
BasalCell Carcinoma	2318	501	504	3323
Benign Keratosis-like Lesions	1835	381	408	2624
Dermatofibroma	171	38	30	209
Actinic Keratoses	603	121	143	867
Melanocytic Nevi	9007	1974	1894	12875
Squamous Cell Carcinoma	462	84	82	628
Vascular Lesions	177	37	39	253

Table 4.1 Dataset Samples

The number of images in 2018 dataset are :

Training Sample : 2593 images

Validation Sample : 100 images

Testing Sample : 1000 images

Sample Images For Each Lable

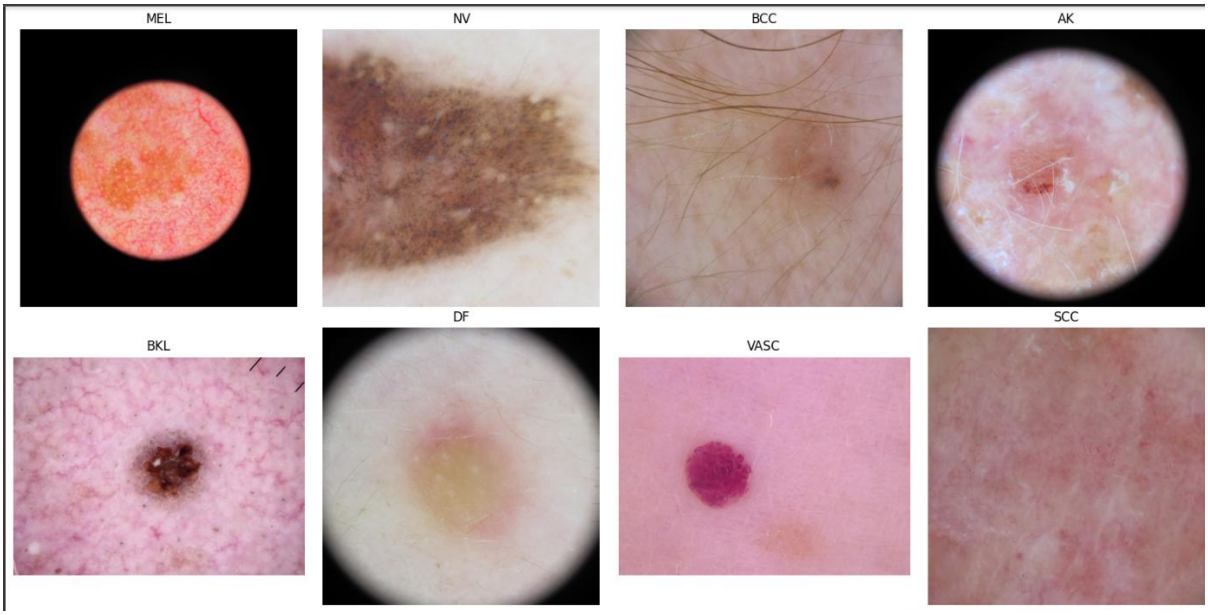
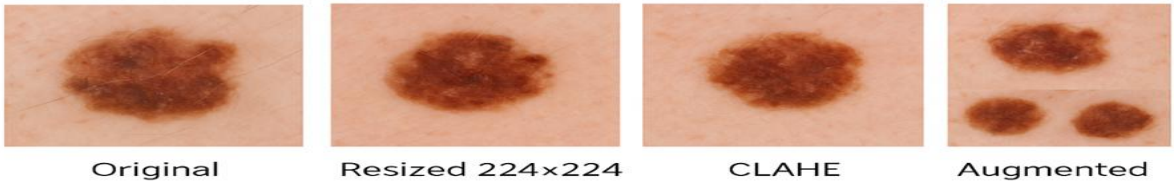


Fig 4.2. ISIC 2019 Dataset

4.2.2 Pre-processing

Preprocessing is a crucial step in developing an effective skin lesion segmentation and classification system. It involves preparing the raw dermoscopic images to enhance important features, reduce noise, and improve the overall performance of deep learning models. Various preprocessing techniques are applied to address challenges such as illumination variations, artifacts, and class imbalance. The different methods used in this project are explained in detail below.



Resizing

All dermoscopic images are resized to 224 × 224 pixels to match the input size required by DenseNet. Resizing ensures uniformity across the dataset, which is important because CNN models expect a fixed input shape. This step also reduces computational complexity by lowering the number of pixels the network must process, without significantly compromising image quality.

Normalization

Pixel intensity values are scaled from the range [0, 255] to [0, 1] by dividing each pixel value by 255. Normalization stabilizes and accelerates the training process by ensuring that the inputs to the neural network are within a small, consistent range. This helps prevent issues such as exploding or vanishing gradients and allows the optimizer to converge more efficiently.

Enhancement (CLAHE)

Contrast Limited Adaptive Histogram Equalization (CLAHE) is applied to improve the visibility of lesion structures. Unlike standard histogram equalization, CLAHE operates on small regions (tiles) and limits the amplification of noise, thus preserving important details while avoiding over-saturation in homogeneous areas. This is especially useful for dermoscopic images, where subtle contrast variations can indicate the difference between lesion types.

Hair Removal

Dermoscopic images often contain hair, which can obstruct lesion boundaries and confuse the model. Hair removal algorithms detect hair pixels using edge detection and morphological operations, then inpaint these regions with surrounding pixel values. This produces cleaner lesion images, ensuring that the model focuses on lesion features rather than irrelevant artifacts.

Augmentation

Data augmentation artificially increases the diversity of the training dataset, improving the model's ability to generalize to unseen data. Transformations applied include:

Rotation (e.g., ±15° to ±45°) – simulates different camera orientations.

Flipping (horizontal/vertical) – mimics changes in viewpoint.

Zoom (in/out) – introduces variations in lesion scale.

Brightness Adjustment – compensates for differences in lighting conditions.

Augmentation not only increases dataset size but also reduces overfitting by preventing the model from memorizing exact image patterns.

4.2.3. Working with Deep Learning model

1. U-Net Model

The U-Net is a convolutional neural network initially developed for biomedical image segmentation at the University of Freiburg, Germany. It is an extension of the fully convolutional network proposed by Long, Shelhamer, and Darrell.

The U-Net architecture enhances the segmentation accuracy by incorporating upsampling operators instead of pooling operations, which increases the output resolution. Additionally, the network employs a symmetric u-shaped structure. This design choice enables the network to capture and propagate context information effectively. By utilizing a large number of feature channels in the upsampling part, the network can extract and utilize rich contextual information, leading to more accurate segmentations.

Unlike traditional networks that rely on fully connected layers, the U-Net does not use them and instead extrapolates missing context in the border region by mirroring the input image. This approach enables the network to handle large images efficiently.

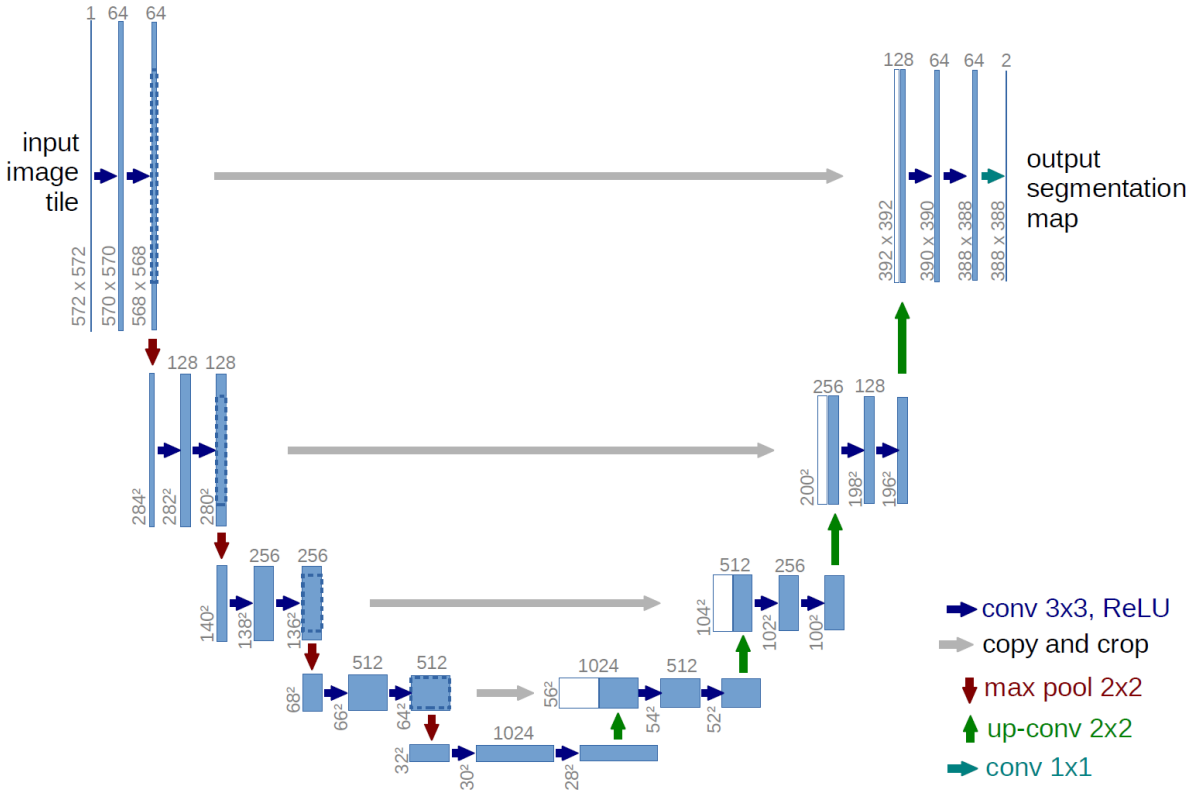


Fig. 4.3 U-Net Architecture

The U-Net architecture consists of an encoder-decoder structure with skip connections, enabling it to capture both high-level and low-level features.

1. **Encoder:** The encoder is responsible for capturing high-level features from the input image.
    - It starts with a series of convolutional layers with a ReLU activation function, followed by padding to maintain spatial dimensions.
    - Convolutional layers apply a set of filters to extract features.
    - After each convolutional layer, another convolutional layer with the same number of filters is applied to capture more complex features.
    - MaxPooling is performed to downsample the feature maps and reduce spatial dimensions.
2. **Bridge:** The bridge connects the encoder and decoder through skip connections.
    - It consists of additional convolutional layers with ReLU activation.
    - The bridge helps in preserving spatial information by concatenating the feature maps from the encoder to the corresponding decoder layers.

3. **Decoder:** The decoder generates the final segmentation map using the concatenated feature maps from the bridge.
    - It starts with upsampling to increase the spatial dimensions of the feature maps.
    - Convolutional layers with ReLU activation are applied to refine the feature maps.
    - The upsampled feature maps are concatenated with the corresponding feature maps from the encoder.
    - More convolutional layers are applied to further refine the features.
    - The decoder ends with a convolutional layer with a sigmoid activation function to produce the final segmentation map.

In this project, **U-Net** was used for the **segmentation of skin lesions** in dermoscopic images. The model predicted pixel-wise masks highlighting the lesion areas, providing a clear visualization of lesion boundaries. The segmentation task was performed separately from classification and helped in analyzing lesion shapes and sizes without directly influencing the classification model.

2. Overview of ResNet model

The **Residual Network (ResNet)**, introduced by Microsoft researchers in 2015, has become a widely used deep learning architecture for image classification tasks. Its success lies in the concept of **residual learning**, which addresses the **vanishing gradient problem** during backpropagation. By incorporating shortcut connections that skip one or more layers, ResNet allows for the effective training of deeper networks, which is essential for capturing complex features in medical images.

The structure of the ResNet model can be summarized as follows:

- **Convolutional Layers:** The input image passes through multiple convolutional layers, each followed by batch normalization and a ReLU activation function. These layers extract low- and high-level features, with downsampling performed via max pooling.
- **Shortcut Connections:** Residual (shortcut) connections skip one or more layers, allowing the network to learn identity mappings. This design addresses the vanishing gradient problem and enables deeper architectures.
- **Output Layer:** A global average pooling layer reduces the spatial dimensions, followed by a fully connected layer with softmax activation to output class probabilities. The predicted class is the one with the highest probability.

Different types of ResNet models used in this project are explained below.

1. Working of ResNet 50 Model

ResNet-50 consists of 50 layers that are divided into 5 blocks, each containing a set of residual blocks. The residual blocks allow for the preservation of information from earlier layers, which helps the network to learn better representations of the input data. ResNet-50 incorporates 50 bottleneck residual blocks, arranged in a stacked manner. The early layers of the network feature conventional convolutional and pooling layers to preprocess the image before it undergoes further processing by the residual blocks. Ultimately, fully connected layers positioned at the pinnacle of the structure utilize the refined data to categorize the image with precision. Through the strategic integration of bottleneck residual blocks and shortcut connections, ResNet-50 adeptly mitigates the vanishing gradient issue, enabling the creation of more profound and potent models for image classification. This innovative architectural approach has opened the door to notable strides in the field of computer vision.

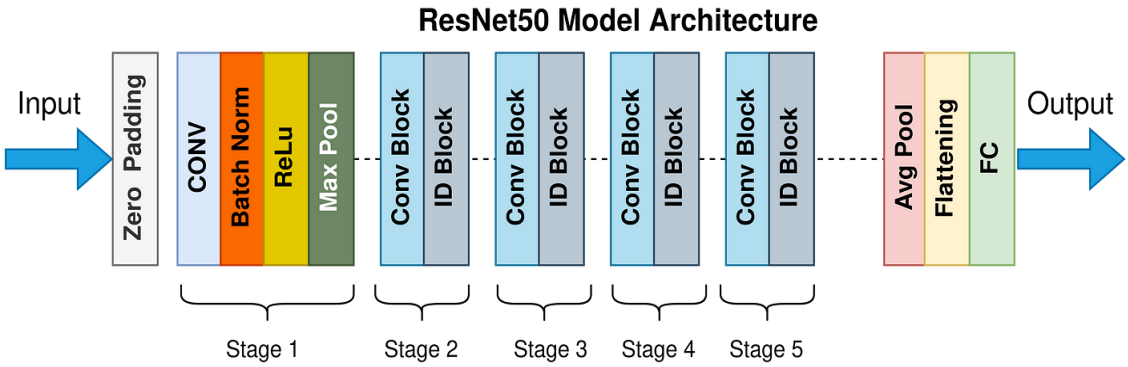


Fig 4.4 ResNet50 layered architecture

The fine-tuning of the ResNet50 model is a common approach for skin lesion classification in dermoscopic images. ResNet50, originally trained on the large-scale ImageNet dataset for object recognition, consists of convolutional, pooling, and fully connected layers arranged in a residual learning framework. In this project, ResNet50 is used as a baseline feature extractor for skin lesion analysis. The lower layers capture general image features such as edges, textures, and patterns, while the final fully connected layers are replaced with new layers tailored for multiclass skin lesion classification.

After modifying the fully connected layers, the model is fine-tuned on the ISIC 2019 dataset, allowing the weights of selected layers to adapt to the characteristics of dermoscopic images. The input data consist of preprocessed dermoscopic images resized to a fixed resolution for compatibility with the network. The model outputs a probability distribution across the eight lesion classes, and the class with the highest probability is selected as the final prediction.

During training, the fine-tuned ResNet50 learns to extract features relevant to distinguishing between different lesion types. However, in this study, ResNet50 achieved lower classification accuracy compared to deeper architectures such as ResNet101 and ResNet152. This performance gap is attributed to its relatively shallower depth and reduced capacity to capture highly complex patterns present in dermoscopic images. Nevertheless, ResNet50 provides a computationally less expensive alternative for rapid experimentation and benchmarking.

2. Working of ResNet 101 Model

ResNet-101 is a deep convolutional neural network (CNN) proposed by He et al. in the paper *"Deep Residual Learning for Image Recognition"* (2015). It is an extended version of the ResNet architecture with **101 layers**, designed to address the degradation problem that arises when increasing network depth. ResNet-101 achieves this by introducing **residual connections** (also called skip connections) that allow gradients to flow directly through the network, making training of very deep models feasible.

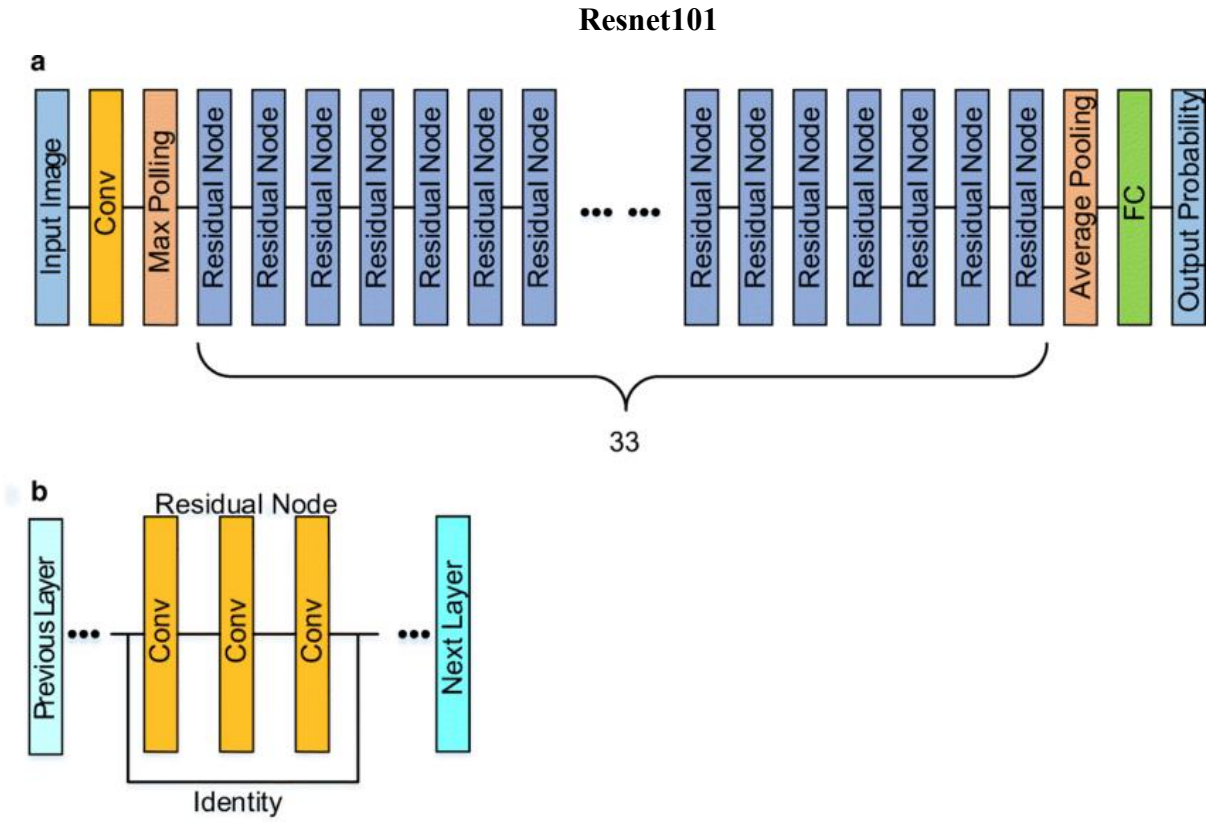


Fig 4.5 Architecture of ResNet101

Core Components of ResNet-101

- Input Layer: The network starts by receiving an input image, typically of size 224×224×3 (for RGB images).
- Convolutional Layers: The initial layers perform convolution to extract low-level features. The first convolution is typically a 7×7 filter with stride 2, followed by batch normalization and ReLU activation, then a 3×3 max pooling with stride 2 to reduce spatial dimensions.
- Residual Blocks: The main body of ResNet-101 consists of stacked residual blocks. Each block is designed to learn a residual mapping, expressed

as F(x)=H(x)−x$F(x)=H(x)-x$, where H(x)$H(x)$ is the desired underlying mapping. The output of the block is then the sum: y=F(x)+x$y=F(x)+x$. This is made possible by the shortcut connection that "skips" over the intermediate layers, directly adding the input to the block's output.

- In ResNet-101, the residual block uses a bottleneck design: each block contains three convolutional layers—1×1, 3×3, and 1×1 filters, in that order. The 1×1 convolutions are used to reduce and then restore dimensions, so computation is more efficient.
- The typical block structure is:
  1×1 conv, 3×3 conv, 1×1 conv, with a skip connection from input to output.

- Block Stacking:
  - ResNet-101 includes a total of 33 bottleneck residual blocks, each with 3 layers, to reach 99 layers, plus the initial conv layer and the final classification fully connected layers (making it total 101 layers).
  - As feature map size decreases (by stride 2), the number of filters doubles.
- Pooling and Classification: After the residual blocks, a global average pooling is performed. Finally, a fully connected (dense) layer with softmax activation outputs the classification predictions.

In this project, ResNet101 was employed through **transfer learning** for multi-class classification of dermoscopic skin lesions. The pre-trained model was loaded, and its top layers were replaced with a custom classification head corresponding to the number of lesion classes in the dataset. The network was fine-tuned on our preprocessed images, which included resizing, hair removal, and contrast enhancement, while data augmentation and class weighting were applied to handle class imbalance. During training, checkpoints and learning rate scheduling ensured stable optimization, allowing the model to effectively learn discriminative features for accurate lesion classification. Finally, the trained model was used to predict the class of new dermoscopic images with high reliability.

3. Working with ResNet152 Model

ResNet-152 is one of the deepest versions in the original ResNet (Residual Network) family. consisting of **152 layers**, designed to address the vanishing gradient problem in very deep networks. Like other ResNet architectures, it uses **residual connections (shortcuts)** that allow the network to learn identity mappings, enabling effective training of extremely deep models without degradation in performance. The network is composed of stacks of convolutional layers, batch normalization, and ReLU activations, with occasional downsampling through convolutional strides or pooling layers. Its depth allows it to capture highly complex and fine-grained features from images, which is particularly useful for subtle patterns like those found in dermoscopic skin lesion.

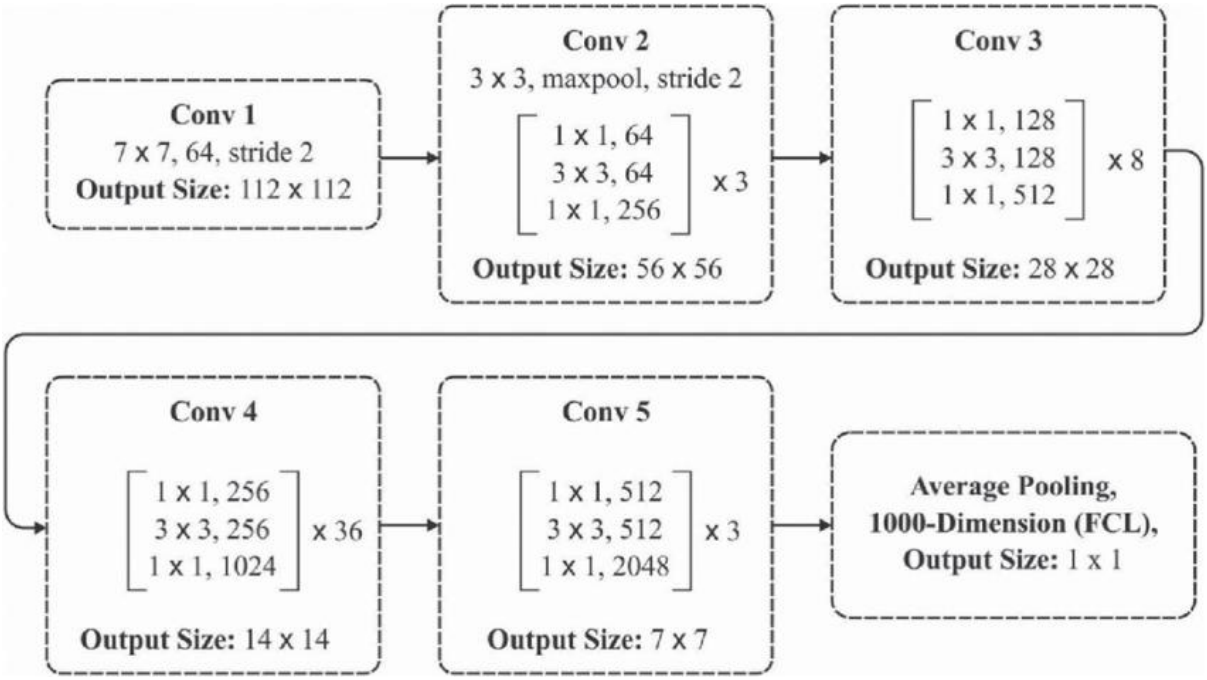


Fig 4.6 Architecture of ResNet152

Core Principles :

- Residual Learning: Instead of each layer learning a direct function, it learns a *residual* function: F(x)=H(x)−x$F(x)=H(x)-x$. This is implemented through *skip connections* (also known as shortcut connections), allowing gradients to flow directly through identity paths, which facilitates deep network training.
- Bottleneck Blocks: ResNet-152, like ResNet-50 and ResNet-101, uses a *bottleneck design* in its residual blocks. Each block has three layers: 1×11×1 (for compression), 3×33×3, 1×11×1 (for decompression) convolutions, with a skip connection linking the input to the block's output.

In this project, **ResNet152** was employed through transfer learning to classify dermoscopic skin lesions into multiple categories. The pre-trained ResNet152 model was loaded, and its top layers were replaced with a custom classification head suited to the number of lesion classes. Images were preprocessed with resizing, hair removal, and contrast enhancement, and data augmentation and class weighting were applied to address class imbalance.

Although ResNet152 has a much deeper architecture than ResNet101 and is capable of learning very complex and fine-grained features, its high depth also makes it **more data-hungry**. With the limited size of the ISIC dataset available for training, the model struggled to generalize effectively, leading to lower validation accuracy compared to ResNet101. Despite this, it successfully extracted hierarchical and discriminative features from dermoscopic images, and the experiment highlighted that **network depth must be matched to dataset size** for optimal performance.

This insight informed the decision to prefer ResNet101 for the final deployment, as it provided a better balance between model capacity and available training data, achieving higher and more reliable classification accuracy.

4.2.4 Performance Matrix

In this project, several performance metrics were used to evaluate the effectiveness of the classification model on dermoscopic skin lesion images. Since the task involves multiple classes with some class imbalance, using multiple complementary metrics provides a comprehensive understanding of model performance.

- **Accuracy:**
  - Measures the proportion of correctly classified images out of the total images.
  - While intuitive, accuracy alone can be misleading for imbalanced datasets, as a model could predict the dominant class most of the time and still appear accurate.
- **Precision, Recall, and F1-Score (per class):**
  - **Precision** indicates how many of the images predicted as a certain class actually belong to that class.
  - **Recall (Sensitivity)** measures how many of the true images of a class were correctly identified.
  - **F1-Score** is the harmonic mean of precision and recall, providing a balanced measure of the model's performance for each class.
  - These metrics are particularly important in medical imaging, where false negatives (missing a melanoma, for example) are more critical than false positives.
- **Confusion Matrix:**
  - A visual representation showing the number of correct and incorrect predictions for each class.
  - Helps identify which classes are being confused with each other, highlighting potential areas for data augmentation or model improvement.
- **ROC-AUC (Receiver Operating Characteristic – Area Under Curve):**
  - Evaluates the model's ability to discriminate between classes.
  - A higher AUC indicates better separability between classes, which is crucial in multi-class medical classification tasks.

- **Macro-Averaging vs Weighted-Averaging:**
  - **Macro-Averaged metrics** treat all classes equally, useful for understanding overall performance irrespective of class size.
  - **Weighted-Averaged metrics** account for class imbalance by giving more weight to classes with more samples, providing a realistic picture of the model's real-world performance.

4.2.5. Web Application Implementation

To provide an interactive and user-friendly interface for skin lesion analysis, a web application named **DermaCare** was developed. The website allows users to upload dermoscopic images and receive automated predictions from the trained skin lesion classification model. It demonstrates the practical deployment of the project, making the research accessible to non-technical users and providing a platform for visualizing and understanding model predictions. While the segmentation and classification tasks were implemented separately, the website showcases the overall functionality of the system in an end-to-end manner.

Technologies Used

- **Frontend:** Built using **ReactJS**, providing a responsive and dynamic user interface. Components are structured to allow smooth image upload, display of predictions, and interactive navigation.
- **Backend:** Implemented using **Flask/FastAPI** to handle API requests. The backend loads the trained models (ResNet101 for classification and U-Net for segmentation) and processes incoming images.
- **Preprocessing:** Images uploaded by users are automatically resized and normalized to match the input requirements of the models.
- **Communication:** The frontend communicates with the backend via **HTTP requests**, sending the uploaded images and receiving predictions or segmentation results in JSON format.
- **Visualization:** Segmentation masks and classification results are displayed on the frontend for easy interpretation.

CHAPTER 5

EXPERIMENTS AND RESULTS

5. Experiments

The system was implemented using two publicly available datasets from the ISIC archive. For segmentation, the ISIC 2018 dataset containing dermoscopic images and their ground-truth masks was used, while the ISIC 2019 dataset with over 25,000 annotated images across seven diagnostic categories was employed for classification. The datasets were split into training, validation, and test sets.

Before training, images were preprocessed to improve quality and model performance. Hair artifacts were removed, and lesion visibility was enhanced through CLAHE (Contrast Limited Adaptive Histogram Equalization). All images were resized to a fixed resolution (224×224 for classification, 256×256 for segmentation) and normalized to the range [0,1]. To address data scarcity and improve generalization, augmentation techniques such as rotation, flipping, zooming, and brightness adjustments were applied. Since the classification dataset was highly imbalanced, class weights and oversampling strategies were used to ensure minority classes were properly represented during training.

For segmentation, a U-Net model with a ResNet encoder was implemented to generate binary lesion masks. The model was trained with a combined Binary Cross-Entropy and Dice Loss function and evaluated using Dice Coefficient and Jaccard Index (IoU).

For classification, ResNet50, ResNet101,and ResNet152 architectures were fine-tuned. The fully connected layers were replaced with dense layers and dropout, and the final output layer used softmax activation for multi-class predictions. Training was performed in phases, beginning with frozen backbones and progressing to fine-tuning with reduced learning rates. The Adam optimizer, ReduceLROnPlateau, and EarlyStopping callbacks were used for stable training. Performance was measured using accuracy, precision, recall, F1-score, and ROC-AUC.

Finally, the trained models were deployed in a web-based application named *DermaCare*. The backend, built with Flask, handled preprocessing, segmentation, and classification, while the React-based frontend allowed users to upload images and view predictions in an interactive interface.

• Implementation: ResNet50 Model summary

Layer (type)	Output Shape	Param #	Connected to
input_layer_4 (InputLayer)	(None, 75, 100, 3)	0	-
conv1_pad (ZeroPadding2D)	(None, 81, 106, 3)	0	input_layer_4[0]…
conv1_conv (Conv2D)	(None, 38, 50, 64)	9,472	conv1_pad[0][0]
conv1_bn (BatchNormalizatio…	(None, 38, 50, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 38, 50, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 40, 52, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 19, 25, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 19, 25, 64)	4,160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormalizatio…	(None, 19, 25, 64)	256	conv2_block1_1_c…
conv2_block1_1_relu (Activation)	(None, 19, 25, 64)	0	conv2_block1_1_b…
conv2_block1_2_conv (Conv2D)	(None, 19, 25, 64)	36,928	conv2_block1_1_r…
conv2_block1_2_bn (BatchNormalizatio…	(None, 19, 25, 64)	256	conv2_block1_2_c…
conv2_block1_2_relu (Activation)	(None, 19, 25, 64)	0	conv2_block1_2_b…
conv5_block3_1_bn (BatchNormalizatio…	(None, 3, 4, 512)	2,048	conv5_block3_1_c…
conv5_block3_1_relu (Activation)	(None, 3, 4, 512)	0	conv5_block3_1_b…
conv5_block3_2_conv (Conv2D)	(None, 3, 4, 512)	2,359,808	conv5_block3_1_r…
conv5_block3_2_bn (BatchNormalizatio…	(None, 3, 4, 512)	2,048	conv5_block3_2_c…
conv5_block3_2_relu (Activation)	(None, 3, 4, 512)	0	conv5_block3_2_b…
conv5_block3_3_conv (Conv2D)	(None, 3, 4, 2048)	1,050,624	conv5_block3_2_r…
conv5_block3_3_bn (BatchNormalizatio…	(None, 3, 4, 2048)	8,192	conv5_block3_3_c…
conv5_block3_add (Add)	(None, 3, 4, 2048)	0	conv5_block2_out… conv5_block3_3_b…
conv5_block3_out (Activation)	(None, 3, 4, 2048)	0	conv5_block3_add…
global_average_poo… (GlobalAveragePool…	(None, 2048)	0	conv5_block3_out…
dense_9 (Dense)	(None, 512)	1,049,088	global_average_p…
dropout_3 (Dropout)	(None, 512)	0	dense_9[0][0]
dense_10 (Dense)	(None, 8)	4,104	dropout_3[0][0]

Total params: 24,640,904 (94.00 MB)
Trainable params: 1,053,192 (4.02 MB)
Non-trainable params: 23,587,712 (89.98 MB)

Fig 5.1 ResNet50 Model Summary

• Implementation: ResNet101 Model summary

Layer (type)	Output Shape	Param #	Connected to
input_layer_3 (InputLayer)	(None, 224, 224, 3)	0	-
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_layer_3[0]…
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,472	conv1_pad[0][0]
conv1_bn (BatchNormalizatio…	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4,160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormalizatio…	(None, 56, 56, 64)	256	conv2_block1_1_c…
conv2_block1_1_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_1_b…
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36,928	conv2_block1_1_r…
conv2_block1_2_bn (BatchNormalizatio…	(None, 56, 56, 64)	256	conv2_block1_2_c…
conv2_block1_2_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_2_b…
conv5_block3_2_relu (Activation)	(None, 7, 7, 512)	0	conv5_block3_2_b…
conv5_block3_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,624	conv5_block3_2_r…
conv5_block3_3_bn (BatchNormalizatio…	(None, 7, 7, 2048)	8,192	conv5_block3_3_c…
conv5_block3_add (Add)	(None, 7, 7, 2048)	0	conv5_block2_out… conv5_block3_3_b…
conv5_block3_out (Activation)	(None, 7, 7, 2048)	0	conv5_block3_add…
global_average_poo… (GlobalAveragePool…	(None, 2048)	0	conv5_block3_out…
dense_6 (Dense)	(None, 512)	1,049,088	global_average_p…
dropout_3 (Dropout)	(None, 512)	0	dense_6[0][0]
dense_7 (Dense)	(None, 8)	4,104	dropout_3[0][0]

Total params: 43,711,368 (166.75 MB)
Trainable params: 43,606,024 (166.34 MB)
Non-trainable params: 105,344 (411.50 KB)

Fig 5.2 ResNet101 Model Summary

Implementation: ResNet152 Model summary

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_layer[0][0]
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,472	conv1_pad[0][0]
conv1_bn (BatchNormalizatio…	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4,160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormalizatio…	(None, 56, 56, 64)	256	conv2_block1_1_c…
conv2_block1_1_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_1_b…
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36,928	conv2_block1_1_r…
conv2_block1_2_bn (BatchNormalizatio…	(None, 56, 56, 64)	256	conv2_block1_2_c…
conv2_block1_2_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_2_b…
conv5_block3_1_bn (BatchNormalizatio…	(None, 7, 7, 512)	2,048	conv5_block3_1_c…
conv5_block3_1_relu (Activation)	(None, 7, 7, 512)	0	conv5_block3_1_b…
conv5_block3_2_conv (Conv2D)	(None, 7, 7, 512)	2,359,808	conv5_block3_1_r…
conv5_block3_2_bn (BatchNormalizatio…	(None, 7, 7, 512)	2,048	conv5_block3_2_c…
conv5_block3_2_relu (Activation)	(None, 7, 7, 512)	0	conv5_block3_2_b…
conv5_block3_3_conv (Conv2D)	(None, 7, 7, 2048)	1,050,624	conv5_block3_2_r…
conv5_block3_3_bn (BatchNormalizatio…	(None, 7, 7, 2048)	8,192	conv5_block3_3_c…
conv5_block3_add (Add)	(None, 7, 7, 2048)	0	conv5_block2_out… conv5_block3_3_b…
conv5_block3_out (Activation)	(None, 7, 7, 2048)	0	conv5_block3_add…
global_average_poo… (GlobalAveragePool…	(None, 2048)	0	conv5_block3_out…
dense (Dense)	(None, 512)	1,049,088	global_average_p…
dropout (Dropout)	(None, 512)	0	dense[0][0]
dense_1 (Dense)	(None, 8)	4,104	dropout[0][0]

Total params: 59,424,136 (226.69 MB)
Trainable params: 1,053,192 (4.02 MB)
Non-trainable params: 58,370,944 (222.67 MB)

Fig 5.3 ResNet152 Model Summary

Implementation: U-Net Model summary

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 256, 384, 3)	0	-
conv2d (Conv2D)	(None, 256, 384, 16)	448	input_layer_1[0]…
batch_normalization (BatchNormalizatio…	(None, 256, 384, 16)	64	conv2d[0][0]
activation (Activation)	(None, 256, 384, 16)	0	batch_normalizat…
conv2d_1 (Conv2D)	(None, 256, 384, 16)	2,320	activation[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 256, 384, 16)	64	conv2d_1[0][0]
activation_1 (Activation)	(None, 256, 384, 16)	0	batch_normalizat…
max_pooling2d (MaxPooling2D)	(None, 128, 192, 16)	0	activation_1[0][…
dropout (Dropout)	(None, 128, 192, 16)	0	max_pooling2d[0]…
conv2d_2 (Conv2D)	(None, 128, 192, 32)	4,640	dropout[0][0]
concatenate_3 (Concatenate)	(None, 256, 384, 32)	0	conv2d_transpose… activation_1[0][…
dropout_7 (Dropout)	(None, 256, 384, 32)	0	concatenate_3[0]…
conv2d_16 (Conv2D)	(None, 256, 384, 16)	4,624	dropout_7[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 256, 384, 16)	64	conv2d_16[0][0]
activation_16 (Activation)	(None, 256, 384, 16)	0	batch_normalizat…
conv2d_17 (Conv2D)	(None, 256, 384, 16)	2,320	activation_16[0]…
batch_normalizatio… (BatchNormalizatio…	(None, 256, 384, 16)	64	conv2d_17[0][0]
activation_17 (Activation)	(None, 256, 384, 16)	0	batch_normalizat…
conv2d_18 (Conv2D)	(None, 256, 384, 1)	17	activation_17[0]…

Total params: 2,164,593 (8.26 MB)
Trainable params: 2,161,649 (8.25 MB)
Non-trainable params: 2,944 (11.50 KB)

Fig 5.4 U-Net Model Summary

5.1 Classification Report

The Precision, Recall and F1-scope is calculated for each model. Precision is the ratio of correctly predicted positive observations. Recall or sensitivity is the ratio of correctly predicted positive observation to the all observations in actual class.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
MEL	0.42	0.40	0.41	362
NV	0.83	0.68	0.75	1030
BCC	0.53	0.38	0.45	266
AK	0.52	0.16	0.24	70
BKL	0.41	0.34	0.37	210
DF	0.55	0.32	0.40	19
VASC	0.83	0.25	0.38	20
SCC	0.29	0.10	0.15	50
Micro Avg	0.65	0.52	0.57	2027
Macro Avg	0.55	0.33	0.39	2027
Weighted Avg	0.65	0.52	0.57	2027
Samples Avg	0.51	0.52	0.51	2027

Table 5.1 Classification report table of ResNet50

Class	Precision	Recall	F1-Score	Support
MEL	0.70	0.66	0.68	664
NV	0.88	0.89	0.89	1974
BCC	0.75	0.86	0.80	501
AK	0.62	0.61	0.62	121
BKL	0.70	0.65	0.68	381
DF	0.77	0.71	0.74	38
VASC	0.81	0.81	0.81	37
SCC	0.68	0.62	0.65	84
Accuracy	-	-	0.80	3800
Macro Avg	0.74	0.73	0.73	3800
Weighted Avg	0.80	0.80	0.80	3800

Table 5.2 Classification Result of ResNet101

Class	Precision	Recall	F1-Score	Support
MEL	0.62	0.68	0.65	664
NV	0.91	0.82	0.86	1974
BCC	0.74	0.83	0.78	501
AK	0.60	0.62	0.61	121
BKL	0.62	0.69	0.65	381

Class	Precision	Recall	F1-Score	Support
DF	0.69	0.76	0.72	38
VASC	0.79	0.84	0.82	37
SCC	0.59	0.63	0.61	84
Accuracy	-	-	0.78	3800
Macro Avg	0.70	0.73	0.71	3800
Weighted Avg	0.79	0.78	0.78	3800

Table 5.3 Classification Result of ResNet152

Metric	Value
Sensitivity (Recall)	**0.8983**
Specificity	**0.9754**
Precision	**0.8025**
F1 Score	**0.8178**
Global Accuracy	**0.9010**

Table 5.4 Segmentation Result of U-Net Model

5.2 Results

The following figure show the output of the U-Net model:

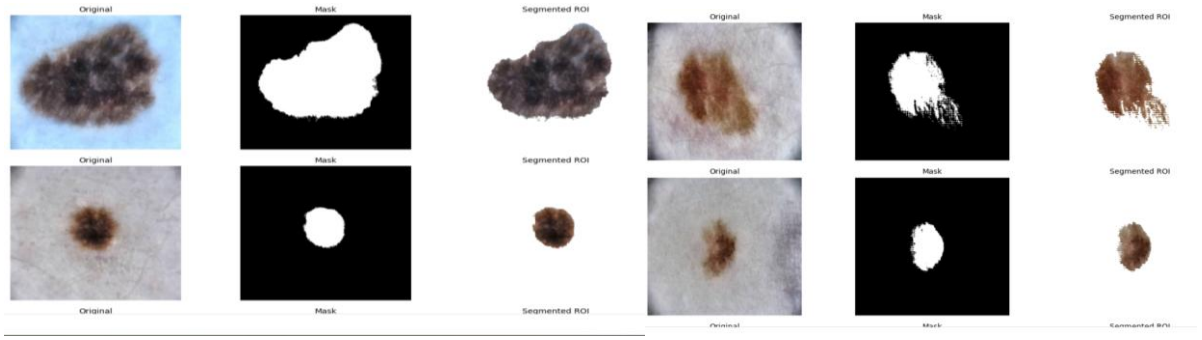


Fig 5.5 U-Net Model Output

The following figure show the output of the ResNet101 model:

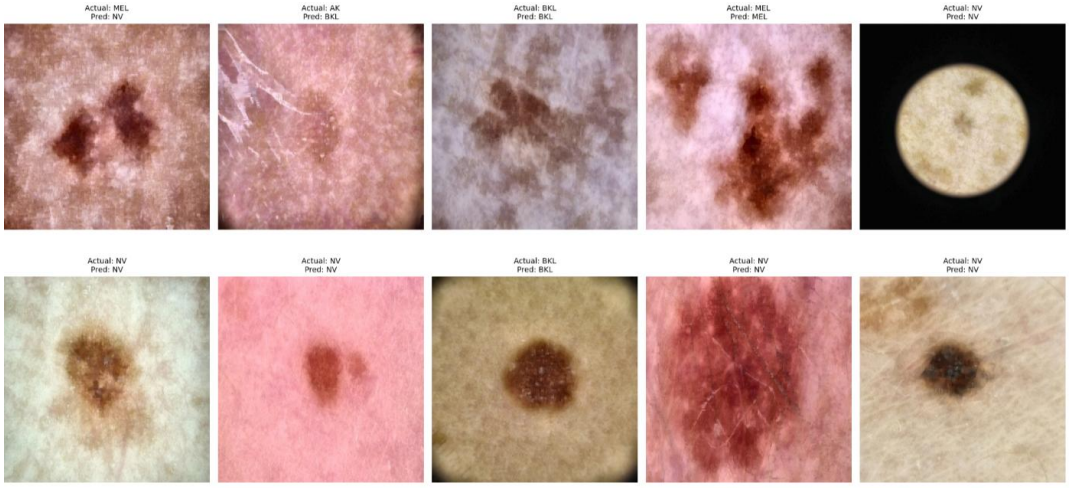


Fig 5.6 ResNet101 Model Output

The following figure show the output of the ResNet152 model:

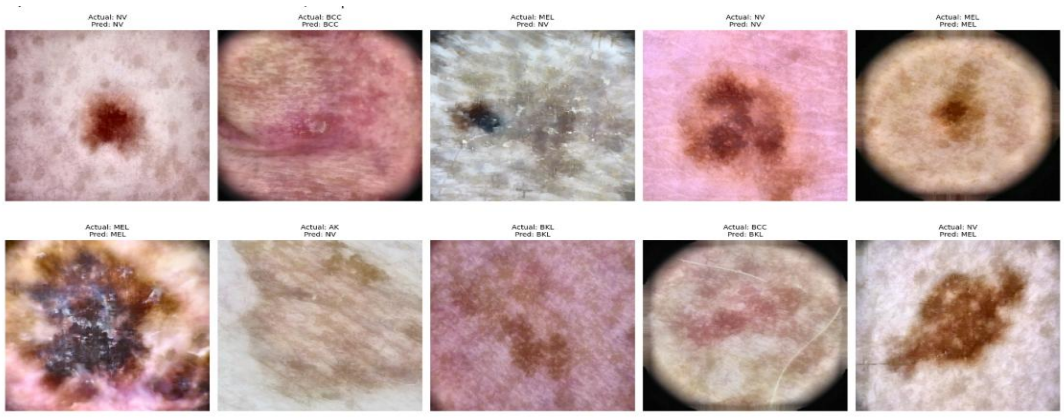


Fig 5.7 ResNet152 Model Output

5.3. User Interface

The DermaCare website features a clean and intuitive **user interface** designed to provide a seamless experience for both medical professionals and general users.

It consists of four main pages:

- **Home Page:** Introduces the application and its purpose, offering easy navigation to other sections.

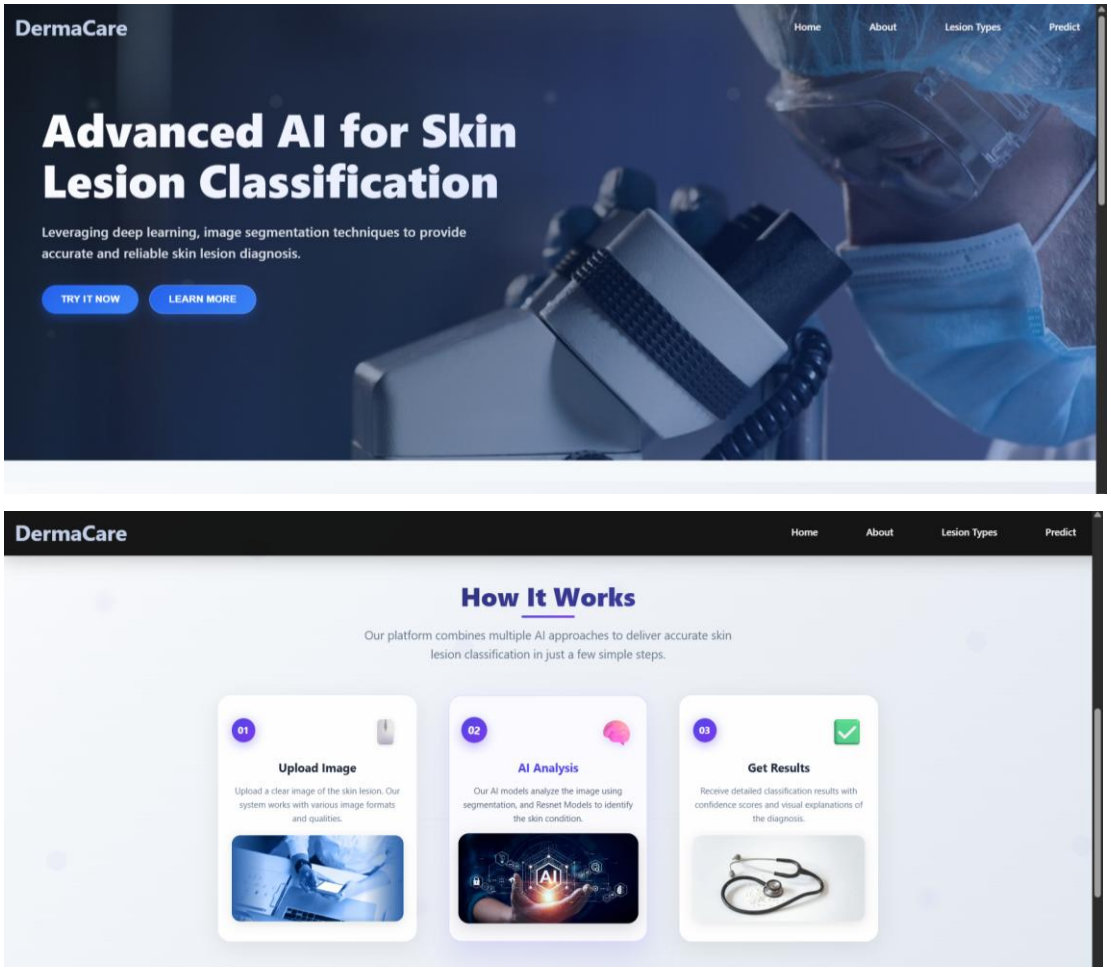


Fig 5.8 Home Page

- **About Page:** Provides information about the project, its objectives, and underlying methodology.

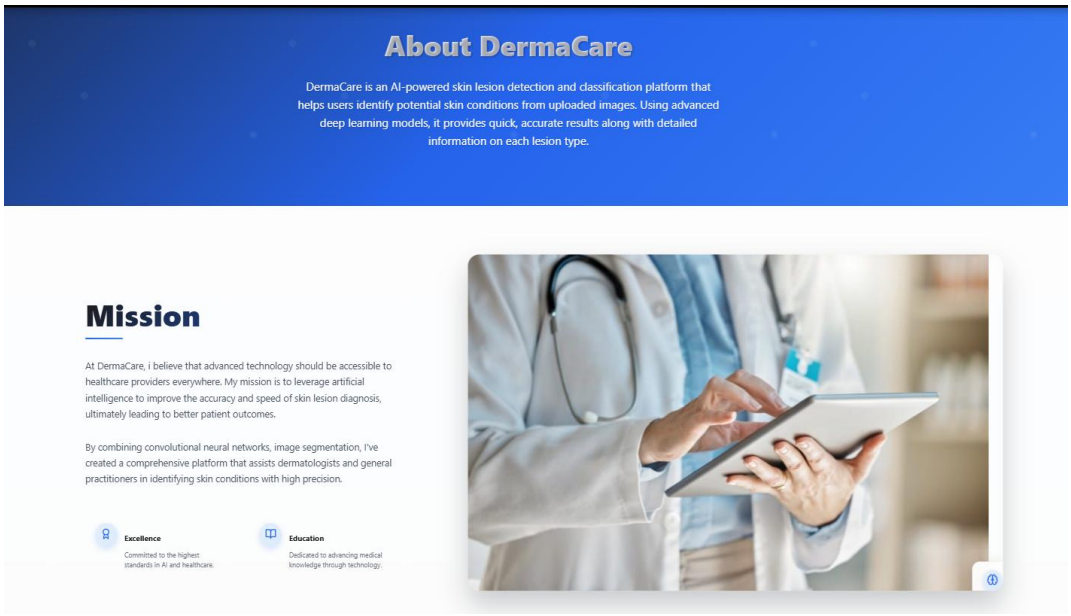


Fig 5.9 About Page

- **Lesion Types Page:** Describes different types of skin lesions with brief explanations and example images, helping users understand the classification categories.

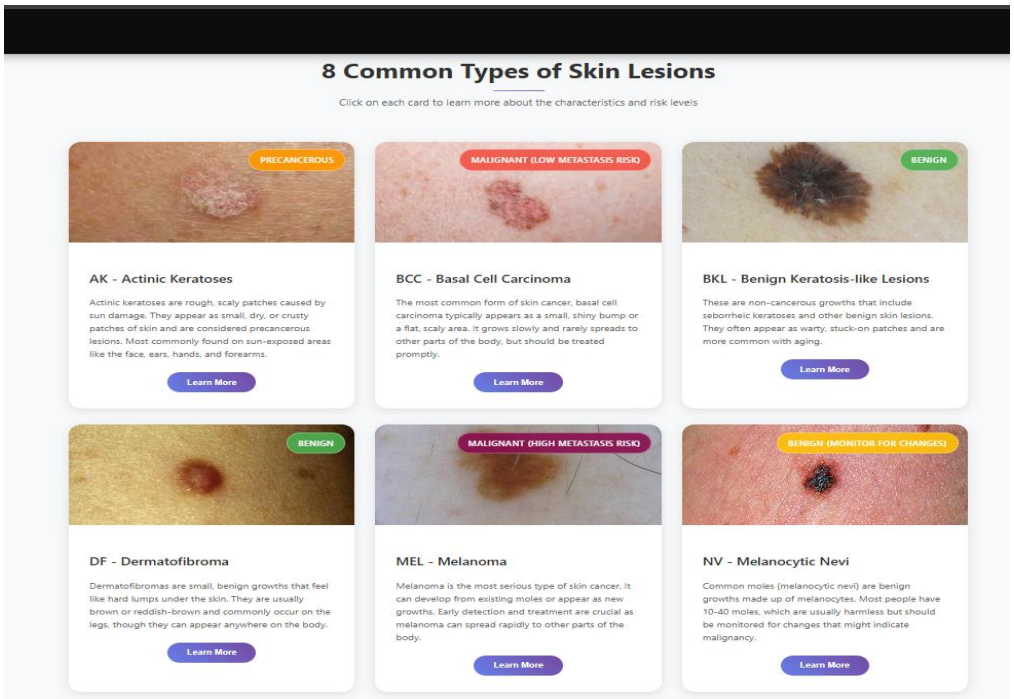


Fig 5.10 LesionTypes Page

- **Predict Page:** Allows users to upload dermoscopic images and view the predicted lesion type quickly and clearly.

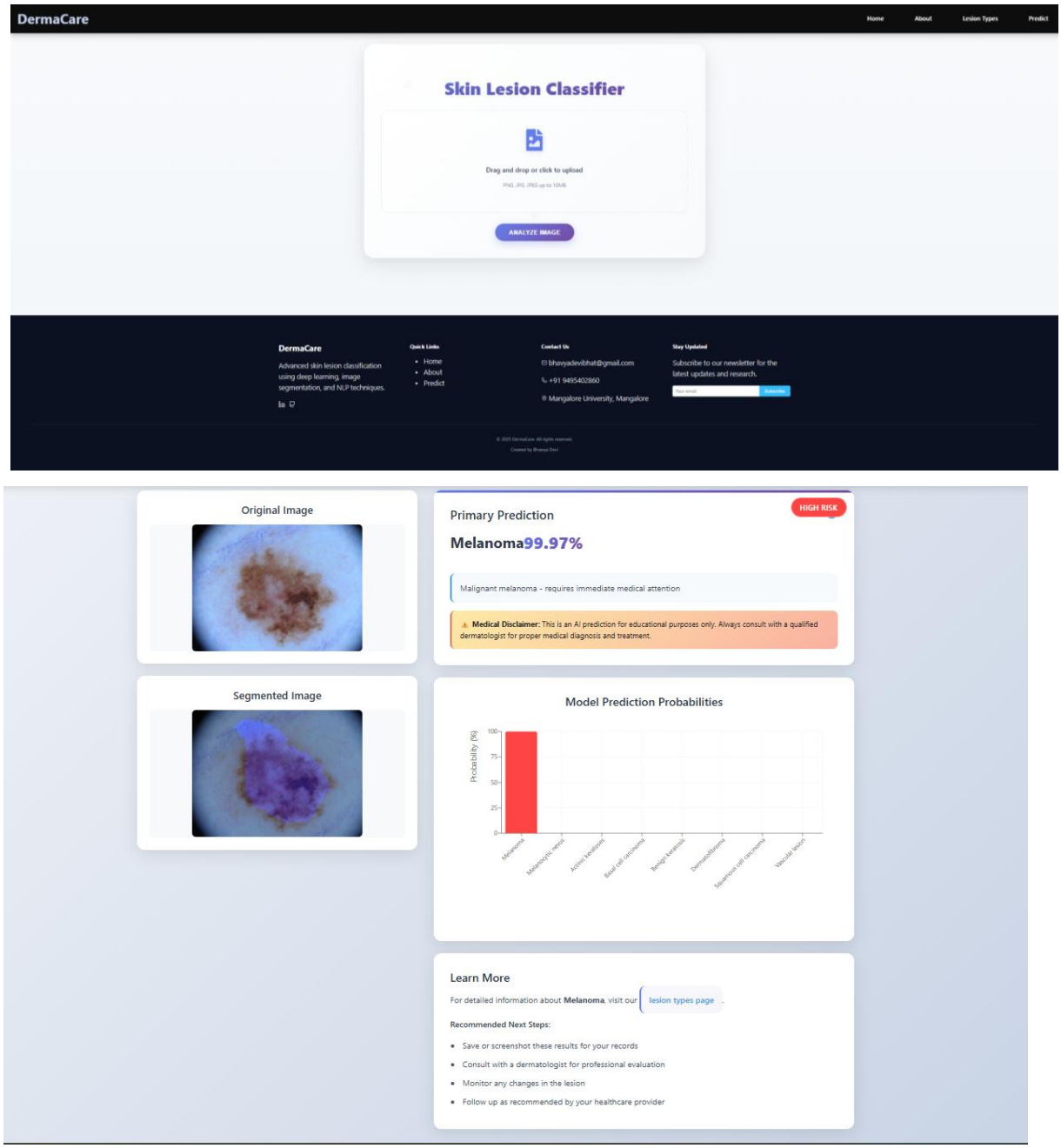


Fig 5.11 Predict Page

CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In this work, an automated system is presented for the classification of skin lesions using dermoscopic images, employing three deep learning architectures from the ResNet family: ResNet50, ResNet101, and ResNet152. All models were fine-tuned on the ISIC 2019 dataset to identify multiple categories of skin lesions. The ResNet50 architecture, with its comparatively shallow depth, achieved a prediction accuracy of 50%, while the deeper ResNet101 model demonstrated significantly better performance, achieving 80% accuracy. ResNet152, despite its increased depth, yielded a slightly lower accuracy of 75%, suggesting that excessive depth may not always lead to improved results in this domain. Among the three models, ResNet101 proved to be the most effective, striking a balance between depth and generalization capability, making it the best-performing architecture for the skin lesion classification task in this study.

6.2 FUTURE WORK

While the current system effectively performs skin lesion classification using the ResNet101 architecture, there is significant scope for further enhancement. In the future, the model can be extended to incorporate more advanced deep learning architectures, such as Vision Transformers (ViT) or hybrid CNN–Transformer models, to improve feature extraction and classification accuracy.

The dataset can be expanded by including additional publicly available dermoscopic image repositories or by applying advanced data augmentation techniques to handle class imbalance and improve model generalization.

Furthermore, integration of lesion segmentation before classification can enhance the focus on the region of interest, potentially improving diagnostic precision. On the application side, the system can be deployed as a cross-platform mobile application, enabling real-time predictions using device cameras.

Finally, the system could evolve into a multi-task framework capable of providing not only lesion type classification but also severity assessment and treatment recommendations, making it a more comprehensive dermatology support tool.

REFERENCES

[1] Duan Wang, "Skin Lesion Segmentation of Dermoscopy Images Using U-Net", Beijing University of Posts and Telecommunications, June 2023.
https://www.researchgate.net/publication/372214599_Skin_lesion_segmentation_of_dermoscopy_images_using_U-Net

[2] Lina Liu, Lichao Mou, Xiao Xiang Zhu, Mrinal Manda, "Skin Lesion Segmentation Based on Improved U-net" IEEE Access, 11 October 2019.
https://ieeexplore.ieee.org/document/8861848

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", arXiv preprint arXiv:1512.03385, 10 December 2015.
https://arxiv.org/abs/1512.03385

[4] Mehwish Zafar, Muhammad Imran Sharif, Muhammad Irfan Sharif, Seifedine Kadry, Syed Ahmad Chan Bukhari, Hafiz Tayyab Rauf, "Skin Lesion Analysis and Cancer Detection Based on Machine/Deep Learning Techniques: A Comprehensive Survey."
https://pubmed.ncbi.nlm.nih.gov/36676093/

[5] Huthaifa Abuhammad, Mohammad Alhawarat, Duaa Mehiar, "Soft Attention-Enhanced ResNet101 for Robust Skin Lesion Classification in Dermoscopic Images," 2025 International Conference for Artificial Intelligence, Applications, Innovation and Ethics (AI2E), February 2025.
https://www.researchgate.net/publication/391684646_Soft_Attention-Enhanced_ResNet101_for_Robust_Skin_Lesion_Classification_in_Dermoscopic_Images

[6] B. Shetty, R. Fernandes, A. P. Rodrigues, R. Chengoden, S. Bhattacharya, and K. Lakshmanna, "Skin lesion classification of dermoscopic images using machine learning and convolutional neural network," *Scientific Reports*, vol. 12, no. 1, p. 20044, Dec. 2022. doi: 10.1038/s41598-022-23942-1.

[7] A. Afroz, R. Zia, A. O. Garcia, M. U. Khan, U. Jilani, and K. M. Ahmed, "Skin lesion classification using machine learning approach: A survey," *IEEE Access*, 2023.
https://ieeexplore.ieee.org/document/9772915

[8] T. G. Debelee, "Skin Lesion Classification and Detection Using Machine Learning Techniques: A Systematic Review," in *Advances in Data Science and Intelligent Data Communication Technologies*, edited by W. A. Mustafa and H. Alquran, Springer, 2022. https://pmc.ncbi.nlm.nih.gov/articles/PMC10572538/

[9]DeepLearning - Mitesh Khapra, SKS Iyengar IIT Ropar and Madras - NPTEL https://www.youtube.com/playlist?list=PLyqSpQzTE6M9gCgajvQbc68Hk_JKGBAYT/

APPENDICES

Code Snapshots

1. Folder Creation and Dataset Loading

```
import os
import pandas as pd
import shutil
from sklearn.model_selection import train_test_split

# === Paths ===

images_path = '/content/drive/MyDrive/Final_Project/Classification/ISIC_2019_Training_Data'
gt_csv_path = '/content/drive/MyDrive/Final_Project/Classification/ISIC_2019_Training_GroundTruth.csv'
output_path = os.path.join(root_path, 'classification_Dataset')

# === Create folders ===
folders = [
    os.path.join(output_path, 'train'),
    os.path.join(output_path, 'validate'),
    os.path.join(output_path, 'test')
]

for folder in folders:
    os.makedirs(folder, exist_ok=True)
    print(f" Folder ready: {folder}")
```

```
Folder ready: /content/drive/MyDrive/skin_classification/classification_Dataset/train
Folder ready: /content/drive/MyDrive/skin_classification/classification_Dataset/validate
Folder ready: /content/drive/MyDrive/skin_classification/classification_Dataset/test
```

2. Preprocessing

```
def remove_hair(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    kernel = cv2.getStructuringElement(1, (17, 17))
    blackhat = cv2.morphologyEx(gray, cv2.MORPH_BLACKHAT, kernel)
    _, mask = cv2.threshold(blackhat, 10, 255, cv2.THRESH_BINARY)
    result = cv2.inpaint(img, mask, 1, cv2.INPAINT_TELEA)
    return result



def apply_CLAHE(img):
    lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    cl = clahe.apply(l)
    merged = cv2.merge((cl, a, b))
    enhanced_img = cv2.cvtColor(merged, cv2.COLOR_LAB2BGR)
    return enhanced_img



def preprocess_image(img):
    resized = cv2.resize(img, (target_width, target_height), interpolation=cv2.INTER_LINEAR)
    no_hair = remove_hair(resized)
    enhanced = apply_CLAHE(no_hair)
    return enhanced
```

3. Handling Class imbalance

```
import pandas as pd
import numpy as np
from sklearn.utils.class_weight import compute_class_weight

# Load your CSV (same as used for training)
df = pd.read_csv(train_csv)

# List of class names
class_names = ['MEL', 'NV', 'BCC', 'AK', 'BKL', 'DF', 'VASC', 'SCC']

# Convert one-hot labels to class indices
y_int = df[class_names].values.argmax(axis=1)

# Compute class weights
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.arange(len(class_names)),
    y=y_int
)

# Convert to dictionary format (required by Keras)
class_weight_dict = dict(enumerate(class_weights))
print(class_weight_dict)
```

```
3: 0.7018286890436986, 1: 0.2460724991673143, 2: 0.9561583261432269, 3: 3.6755804311774463, 4: 1.20783378746594, 5: 12.961257309941521, 6: 12.521892655367232, 7:
.797348484848484}
```

4. Model Loading

```
# === Model Loading ===
initial_epoch = get_last_epoch()
latest_checkpoint = get_latest_checkpoint()

if latest_checkpoint:
    print(f"🔁 Resuming Phase 2 from checkpoint: {latest_checkpoint} at epoch {initial_epoch}")
    model = tf.keras.models.load_model(latest_checkpoint)
else:
    print("🆕 Starting Phase 2 from scratch (loading Phase 1 final model)")
    model = tf.keras.models.load_model(initial_model_path)
```

```
🆕 Starting Phase 2 from scratch (loading Phase 1 final model)
2025-08-13 08:21:00.055999: E external/local_xla/xla/stream_executor/cuda/cuda_driver.cc:152] failed call to cuI
KNOWN ERROR (303)
```

```
# Unfreeze all layers for Phase 2
for layer in model.layers:
    layer.trainable = True

    # Recompile after changing layer.trainable
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

5. Training the model

```
history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=30,         # e.g., 30
    initial_epoch=18,
    class_weight=class_weight_dict,# resume from correct epoch
    callbacks=callbacks
)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `sup
_(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as th
ignored.
  self._warn_if_super_not_called()
Epoch 19/30
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1754149025.524249      94 service.cc:148] XLA service 0x7d4dac003050 initialized for platform CUDA (this does not guarantee that XLA will be
ces:
I0000 00:00:1754149025.525094      94 service.cc:156]   StreamExecutor device (0): Tesla P100-PCIE-16GB, Compute Capability 6.0
I0000 00:00:1754149034.157036      94 cuda_dnn.cc:529] Loaded cuDNN version 90300
I0000 00:00:1754149060.486777      94 device_compiler.h:188] Compiled cluster using XLA!  This line is logged at most once for the lifetime of the proces
555/555 ━━━━━━━━━━━━━━━━━━━━ 427s 544ms/step - accuracy: 0.9255 - loss: 0.1168 - val_accuracy: 0.8034 - val_loss: 0.7674 - learning_rate: 1.0000e-05
Epoch 20/30
555/555 ━━━━━━━━━━━━━━━━━━━━ 167s 300ms/step - accuracy: 0.9258 - loss: 0.1036 - val_accuracy: 0.7945 - val_loss: 0.7769 - learning_rate: 1.0000e-05
Epoch 21/30
555/555 ━━━━━━━━━━━━━━━━━━━━ 162s 292ms/step - accuracy: 0.9258 - loss: 0.1062 - val_accuracy: 0.7992 - val_loss: 0.7710 - learning_rate: 1.0000e-05
Epoch 22/30
555/555 ━━━━━━━━━━━━━━━━━━━━ 0s 271ms/step - accuracy: 0.9266 - loss: 0.1067
Epoch 22: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-06.
555/555 ━━━━━━━━━━━━━━━━━━━━ 166s 298ms/step - accuracy: 0.9266 - loss: 0.1067 - val_accuracy: 0.8011 - val_loss: 0.7808 - learning_rate: 1.0000e-05
Epoch 23/30
555/555 ━━━━━━━━━━━━━━━━━━━━ 163s 294ms/step - accuracy: 0.9384 - loss: 0.0914 - val_accuracy: 0.7953 - val_loss: 0.7869 - learning_rate: 5.0000e-06
Epoch 24/30
555/555 ━━━━━━━━━━━━━━━━━━━━ 164s 296ms/step - accuracy: 0.9342 - loss: 0.0961 - val_accuracy: 0.8021 - val_loss: 0.7938 - learning_rate: 5.0000e-06
Epoch 25/30
555/555 ━━━━━━━━━━━━━━━━━━━━ 0s 269ms/step - accuracy: 0.9430 - loss: 0.0855
Epoch 25: ReduceLROnPlateau reducing learning rate to 2.499999936844688e-06.
555/555 ━━━━━━━━━━━━━━━━━━━━ 165s 297ms/step - accuracy: 0.9430 - loss: 0.0855 - val_accuracy: 0.7895 - val_loss: 0.8057 - learning_rate: 5.0000e-06
```

6. Predicting the output

```
import random
import matplotlib.pyplot as plt

# Pick 10 random indices
random_indices = random.sample(range(len(val_generator.image_filenames)), 10)

plt.figure(figsize=(20, 10))

for i, idx in enumerate(random_indices):
    # Get image info
    image_name = val_generator.image_filenames[idx]
    true_label_index = val_generator.labels[idx]
    true_label_name = val_generator.class_names[true_label_index]
    image_path = os.path.join(val_image_dir, image_name)

    # Load and preprocess the image
    image = cv2.imread(image_path)
    image = cv2.resize(image, (224, 224))
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image_normalized = image_rgb / 255.0
    input_image = np.expand_dims(image_normalized, axis=0)

    # Predict
    pred_probs = model.predict(input_image)
    pred_class_index = np.argmax(pred_probs, axis=1)[0]
    pred_class_name = val_generator.class_names[pred_class_index]

    # Plot
    plt.subplot(2, 5, i + 1)
    plt.imshow(image_rgb)
    plt.axis('off')
    plt.title(f"Actual: {true_label_name}\nPred: {pred_class_name}", fontsize=10)

plt.tight_layout()
plt.show()
```