Ain Shams University

Faculty of Computer and Information Sciences
Scientific Computing Department

SKIN DISEASE RECOGNITION USING THE DISEASE IMAGE

June 2024

Ain Shams University

Faculty of Computer and Information Sciences
Scientific Computing Department

This Project was carried out by:

Sondos Mohamed Mahmoud [SC]

Hagar Salem Awdah [SC]

Asmaa Amr Abdel Hamid [SC]

Mohamed Abdullah Abdel Hakim [SC]

Marlen Ehab Rezk [SC]

Nada Yasser Ahmed [SC]

Hossam El Din Mostafa Nabeh [SC]

Under the Supervision of:

Dr. Doaa Ezat

(Supervisor)

Scientific Computing Department,

Faculty of Computer and Information Sciences,

&

Rezk Mohamed

(Teaching Assistant)

Scientific Computing Department,

Faculty of Computer and Information Sciences,

Ain Shams University.

Table of Contents

List of Figures and Tables

1. Acknowledgement

All praise and thanks to ALLAH, who provided us the strength, patience, and perseverance to complete this work. May this project be a stepping stone towards further achievements in the future.

We would like to express our deepest gratitude to our parents and families, whose unwavering support and encouragement have been instrumental throughout our years of study. Their belief in our abilities has been a constant source of motivation.

Our sincere thanks go to our supervisors, Dr. Doaa Ezat and T.A. Rezk Mohamed, whose guidance, knowledge, and patience have been invaluable throughout this project. Their support and constructive feedback have significantly contributed to the successful completion of this thesis.

Finally, we would thank our friends and all people who gave us support and encouragement.

2. Abstract

Skin disease recognition using disease images is crucial in the medical field, where early detection and diagnosis can significantly enhance patient outcomes. This project aims to develop a mobile application that leverages image processing and machine learning techniques to accurately identify various skin diseases from images, providing a practical tool for both dermatologists and patients.

The primary problem addressed by this project is the need for an accessible, reliable, and efficient tool to assist in diagnosing skin conditions. The main objectives include collecting a comprehensive dataset of skin disease images, preprocessing these images to enhance their quality, implementing and training machine learning models, and evaluating their performance in terms of accuracy and reliability. Additionally, the project focuses on creating a user-friendly mobile application to ensure broad accessibility and ease of use.

The project follows a structured approach: initial research and literature review, data collection and annotation, image preprocessing, model selection and training, mobile application development, and system testing and validation. The system architecture is designed to integrate these components seamlessly, ensuring a robust and user-friendly application.

In conclusion, this project demonstrates the potential of machine learning and mobile technology in medical diagnostics by providing a tool that aids in the early and accurate detection of skin diseases. Future work may involve expanding the dataset, improving model accuracy, and continuously enhancing the mobile application's features to better serve its users.

3. List of Abbreviations

The following nomenclatures shall be used frequently in the document, in order to facilitate such matter, we have summed up the extended terminologies and abbreviated them as follows:

AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
CSV	Comma-Separated Values
DL	Deep Learning
GPU	Graphics Processing Unit
IoT	Internet of Things
ML	Machine Learning
PNG	Portable Network Graphics
SVM	Support Vector Machine
UI	User Interface
UX	User Experience
XML	Extensible Markup Language

4. Chapter One – Introduction

4.1 Problem Definition

Skin diseases affect millions of people worldwide, often leading to discomfort, distress, and significant healthcare costs. Early detection and accurate diagnosis are critical in managing these conditions effectively. However, access to dermatological care can be limited, particularly in remote and underserved regions. This project addresses the challenge of skin disease recognition using disease images, leveraging advanced image processing and machine learning techniques to develop a reliable and accessible diagnostic tool.

4.1.1 History

The history of skin disease recognition dates back to ancient times when visual inspection by experienced practitioners was the primary diagnostic method. With the advent of photography and digital imaging, more sophisticated techniques have been developed. Recent advancements in artificial intelligence (AI) and machine learning (ML) have further revolutionized the field, enabling automated and highly accurate diagnosis.

4.1.2 Applications

The primary application of this project is in the medical field, providing a tool for dermatologists and general practitioners to assist in diagnosing skin conditions. Additionally, it can be used for patient self-assessment, telemedicine services, and educational purposes in medical training programs.

4.2 Motivation

The motivation for this project stems from the need for accessible, reliable, and efficient diagnostic tools for skin diseases. Many individuals do not have easy access

to dermatologists, especially in rural or underserved areas. By creating a mobile application capable of recognizing various skin conditions from images, we aim to bridge this gap, ensuring that more people can receive timely and accurate diagnoses. This tool can potentially reduce the burden on healthcare systems and improve patient outcomes through early intervention.

4.3 Objectives

The main objectives of this project are as follows:

- **Data Collection** Gather a comprehensive dataset of skin disease images from reliable sources
- **Image Processing** Enhance the quality of images through preprocessing techniques to ensure accurate model training
- **Model Development** Implement and train machine learning models, particularly convolutional neural networks ("**CNN**"s), to recognize different skin diseases
- **Performance Evaluation** Evaluate the models' performance in terms of accuracy, precision, recall, and other relevant metrics
- **Application Development** Develop a user-friendly mobile application that integrates the trained models for real-time skin disease recognition
- **Accessibility** Ensure the application is accessible to a wide audience, including non-specialists and individuals in remote areas

4.4 Time Plan

The following time plan was implemented in order to successfully deliver and handout the project in a timely manner.

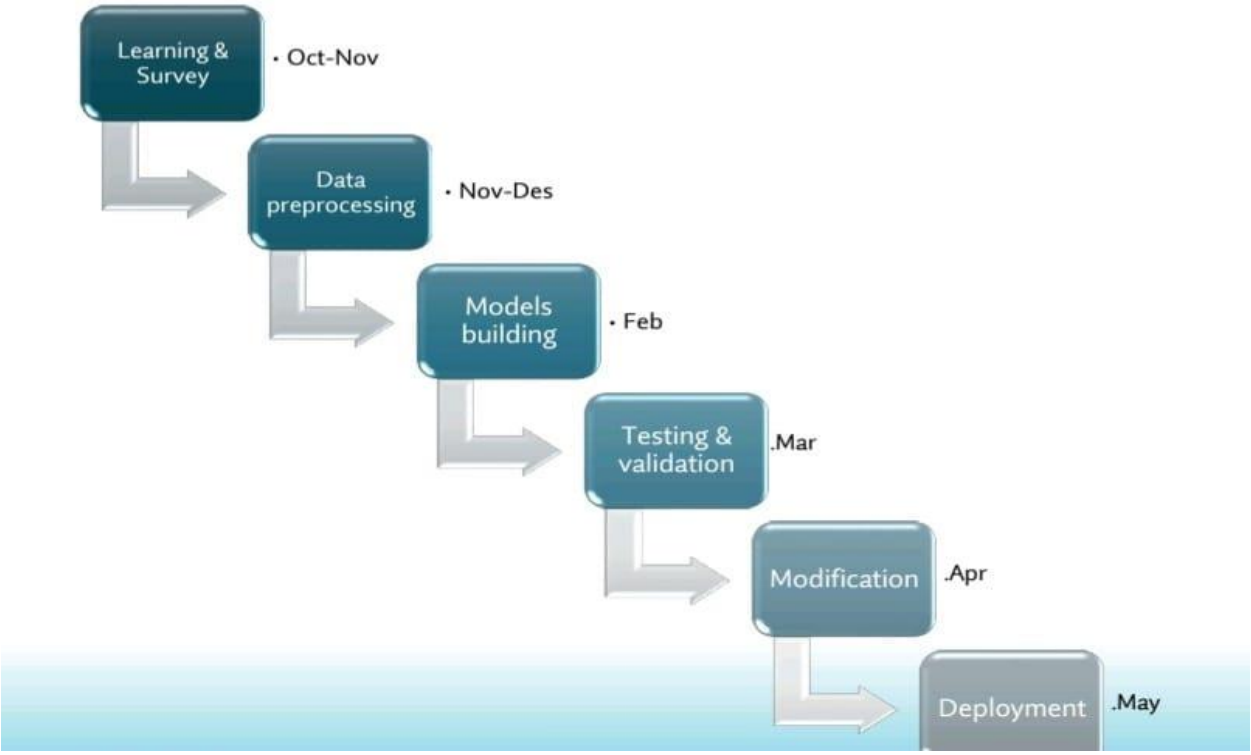


Figure 1 Graduation Project Time Plan

4.5 Documentation Outline

This documentation is organized into the following chapters:

- **Chapter (2) – Background** Provides a literature review, relevant surveys, mathematical background, and necessary algorithms and techniques
- **Chapter (3) – System Architecture** Details the architecture design, showing the main components of the system and their relationships

- **Chapter (4) – System Implementation**	Describes the detailed implementation of the system architecture, including flowcharts and pseudocode
- **Chapter (5) – System Testing**	Explains how to test and run the application, clarified with screenshots and test results
- **Chapter (6) – Conclusion and Future Work**	Conclusion and Future Work - Summarizes the project, discusses its advantages and disadvantages, and outlines potential future work

By following this structured approach, we aim to deliver a comprehensive and functional tool for skin disease recognition that can significantly aid in medical diagnostics.

5. Chapter Two – Background

5.1 System Overview

The application developed in this project aims to revolutionize the diagnosis of skin diseases by leveraging advanced image recognition techniques. It operates as a user-friendly interface accessible via web or mobile platforms, facilitating efficient interaction for both healthcare professionals and individuals seeking self-assessment.

5.1.1 Key Features

1. Image Input Users can upload images of skin lesions or diseases directly from their devices.

2. Preprocessing The system employs preprocessing techniques to enhance image quality, normalize color variations, and remove noise, ensuring accurate analysis.

3. Feature Extraction Advanced algorithms such as Histogram of Oriented Gradients (HOG), texture analysis, and color variation analysis are utilized to extract meaningful features from the input images.

4. Classification The extracted features are fed into machine learning models including Support Vector Machines (SVMs), Convolutional Neural Networks (CNNs) like VGG and ResNet, and other specialized architectures such as Inception Networks. These models classify the skin disease based on learned patterns and provide diagnostic insights.

5. User Interface The application offers an intuitive interface displaying results in a comprehensible manner, including disease classification, probability scores, and recommended actions (e.g., consultation with a dermatologist).

6. Integration It supports integration with existing healthcare systems, enabling seamless data exchange and collaboration between medical professionals and the application's users.

5.1.2 Goals

- Accuracy Achieve high accuracy in disease classification to assist healthcare providers in making informed decisions.

- Accessibility Provide accessible and user-friendly tools for individuals to perform preliminary assessments of skin conditions from anywhere.

- Scalability Ensure scalability to accommodate a growing dataset of images and diverse types of skin diseases, continually improving diagnostic capabilities.

5.1.3 Benefits

- Early Detection Facilitates early detection of skin diseases, potentially leading to targeted treatment and improved patient outcomes.

- Enhanced Self-Diagnosis Allows users experiencing dermatological issues to conduct initial self-assessments of their symptoms and receive general guidance on necessary measures.

- **Healthcare Cost Reduction** By offering an efficient and effective service for diagnosing skin diseases, it can reduce costs associated with frequent visits to dermatologists.

- **Health Awareness** Increases health awareness among individuals by providing accurate and accessible information about dermatological health and diseases.

- **Medical Collaboration** Facilitates efficient data exchange and collaboration between healthcare providers and patients, improving care quality and patient satisfaction.

5.1.4 Conclusion

This system overview sets the stage for understanding the application's design and functionality, emphasizing its role in leveraging technology to enhance the diagnosis and management of skin diseases through innovative image recognition methodologies.

5.2 Definitions

- **Skin Disease Recognition** The automated process of identifying and classifying various dermatological conditions based on visual analysis of skin lesion images.

- **Image Preprocessing** Techniques applied to raw images to enhance quality, reduce noise, and normalize color variations before feature extraction and classification.

- **Feature Extraction** The process of capturing and quantifying relevant information from images, such as texture, color, and shape characteristics, to facilitate disease classification.

- **Histogram of Oriented Gradients (HOG)** A feature descriptor used for object detection and image classification based on gradients' orientation in localized regions.

- **Deep Learning** A subset of machine learning methods based on artificial neural networks with multiple layers (deep architectures) that automatically learn representations of data through the use of hierarchical layers of neurons.

- **Convolutional Neural Network (CNN)**: A type of deep neural network designed to process and classify visual data, widely used in image recognition tasks including dermatological diagnostics.

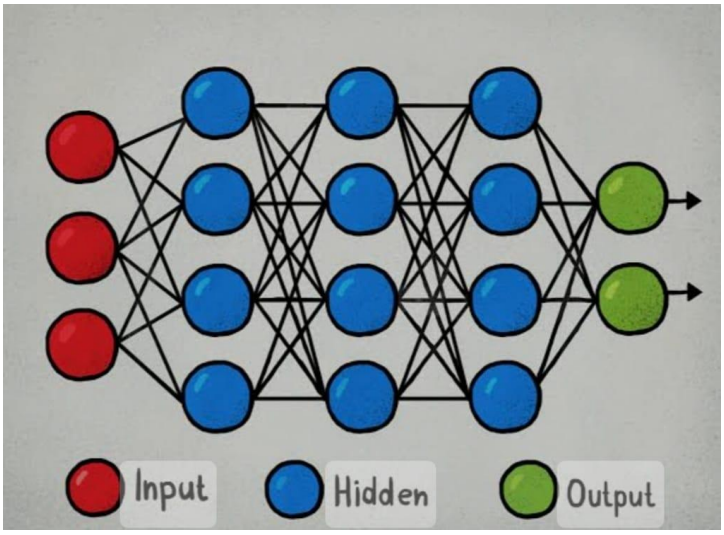


Figure 2 Convolutional Neural Network (CNN)

- **Support Vector Machine (SVM)** A supervised learning model used for classification and regression tasks, effective in separating classes by finding an optimal hyperplane in high-dimensional space.

- **Visual Geometry Group (VGG)** A specific architecture of CNN known for its deep layers and straightforward design, often

	used as a baseline for image classification benchmarks.
- **Inception Network**	Also known as GoogLeNet, it utilizes inception modules to improve computational efficiency and performance in deep learning tasks.
- **Residual Network (ResNet)**:	A deep neural network architecture featuring residual connections, addressing the vanishing gradient problem to enable training of very deep networks effectively.
- **Color Variation Analysis**:	Techniques for analyzing variations in color distribution within images, which can provide insights into different stages and types of skin diseases.

5.3 Feature Extraction

Feature extraction plays a crucial role in the process of identifying and classifying skin diseases from images. It involves capturing meaningful information from raw image data that can distinguish between different types of skin conditions. In this section, various techniques and methods used for feature extraction in the context of skin disease recognition are discussed.

5.3.1 Key Techniques

1. **Histogram of Oriented Gradients (HOG)**:
    - **Description**: HOG is a feature descriptor technique that computes the distribution of gradient orientations in localized portions of an image.

- **Application**: It is effective in capturing the shape and edge information of skin lesions, providing discriminative features for classification tasks.

2. **Texture Analysis**:
   - **Description**: Texture analysis techniques examine patterns within images, such as fine details, roughness, and regularity.
   - **Application**: By analyzing textural characteristics, the system can differentiate between different textures associated with various skin diseases.

3. **Color Variation Analysis**:
   - **Description**: This technique focuses on analyzing variations in color distribution across an image.

   - **Application**: It helps in identifying specific color patterns or anomalies that indicate different stages or types of skin conditions, aiding in accurate classification.

Importance:

- **Discriminative Power**: Effective feature extraction techniques enhance the system's ability to distinguish between visually similar skin diseases, improving diagnostic accuracy.

- **Preprocessing**: Feature extraction is often preceded by image preprocessing steps, such as noise reduction and normalization, to ensure that extracted features are robust and meaningful.

- **Algorithm Compatibility**: The choice of feature extraction methods impacts the performance of subsequent classification algorithms, such as SVMs or CNNs, by providing them with relevant and informative input data.

Integration with Deep Learning:

In the context of deep learning, feature extraction is often integrated within the architecture of convolutional neural networks (CNNs). CNNs automatically learn hierarchical representations of features directly from raw pixel data, eliminating the need for manual feature engineering in some cases. However, traditional feature extraction methods like HOG and texture analysis still play a significant role in augmenting deep learning models' capabilities, particularly in fine-tuning and optimizing performance.

5.3.2 Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG) is a method used to capture the structure and texture information in an image by analyzing the distribution of gradient orientations. Here's a simplified explanation of how HOG is calculated:

5.3.2.1 Steps to Calculate HOG:

1. **Image Preparation**:
   - Convert the image to grayscale to simplify processing.
   - Normalize the image intensity if needed to ensure consistent results.

2. **Gradient Calculation**:
   - Compute the gradient magnitude and orientation for each pixel in the image using gradient filters like Sobel or Prewitt.

3. **Orientation Binning**:
   - Divide the image into small cells (e.g., 8x8 pixels).
   - For each pixel within a cell, determine its gradient orientation and assign it to a predefined orientation bin.

4. **Histogram Construction**:
   - Create a histogram of gradient orientations within each cell.
   - Accumulate the gradient magnitudes into bins corresponding to their orientation.

5. **Block Normalization**:
   - Combine neighboring cells into blocks (e.g., 2x2 cells).
   - Normalize the histograms within each block to make the descriptor invariant to changes in illumination and contrast.

6. **Feature Vector Formation**:
   - Concatenate the normalized block histograms into a single feature vector.
   - Optionally, apply further normalization across the entire feature vector to enhance robustness.

5.3.3 Image Texture

Image texture analysis is a method used to characterize and quantify patterns within images, focusing on properties such as smoothness, roughness, and regularity. This technique plays a crucial role in extracting features that distinguish different textures associated with various skin diseases or other visual elements in images. By analyzing texture, the system can identify unique visual patterns that aid in accurate classification and diagnosis, making it valuable in applications ranging from medical imaging to object recognition in computer vision. Texture analysis complements other feature extraction methods like HOG by providing additional descriptive information that enhances the system's ability to interpret and classify visual data effectively.

5.3.4 Color Variation

Color variation analysis is a technique used to analyze changes in color distribution within an image. This method focuses on identifying distinctive color patterns or anomalies that may indicate different stages or types of skin diseases or other visual elements. By analyzing color variation, the system can improve classification and diagnosis accuracy, enhancing the precise understanding and interpretation of visual data in applications such as medical imaging and computer vision technologies.

5.4 Algorithms

There are 2 different techniques that are traditional machine learning as SVM and deep learning as CNN.

5.4.1 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm that is primarily used for classification tasks, though it can also be adapted for regression. SVM operates by finding the optimal hyperplane in a high-dimensional space that separates data points into different classes with the maximum margin.

Key Concepts:

1. Hyperplane In SVM, a hyperplane is a decision boundary that separates data points belonging to different classes. For a binary classification problem, SVM finds the hyperplane that maximizes the margin, which is the distance between the hyperplane and the closest data points from each class.

2. Margin The margin is crucial in SVM as it determines the separation between classes and influences the algorithm's ability to generalize well on unseen data. SVM aims to find the hyperplane that not only separates classes but also maximizes this margin.

3. Kernel Trick SVM can handle linearly inseparable data by mapping the original input space into a higher-dimensional feature space through a kernel function. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid kernels. This transformation allows SVM to find a linear separation in the transformed space, even when the data is not separable in the original space.

4. Support Vectors: These are the data points that lie closest to the decision boundary (hyperplane) and determine the maximum margin. Support vectors are critical as they influence the placement and orientation of the hyperplane.

Limitations:

- Computationally Intensive Training SVMs can be time-consuming, especially when dealing with large datasets.

- Selection of Kernel Choosing the right kernel function and tuning its parameters can significantly impact SVM's performance.

- Interpretability SVMs do not provide direct probability estimates, making it challenging to interpret the confidence of predictions compared to probabilistic classifiers like logistic regression.

5.4.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a class of deep neural networks that are particularly well-suited for processing and analyzing visual data. CNNs have revolutionized the field of computer vision and are widely used for image recognition, object detection, and other tasks that involve visual pattern recognition.

Key Concepts:

1. Convolutional Layer:

-Filters/Kernels: The core operation of a CNN is the convolution, which involves sliding a filter (also called a kernel) over the input image to produce a feature map. Each filter detects

specific features, such as edges, textures, or patterns, by performing element-wise multiplications and summing the results.

- Stride and Padding: The stride determines the step size of the filter as it moves across the image. Padding involves adding extra pixels around the border of the image to control the spatial dimensions of the output feature map. Padding can be "same" (output size matches input size) or "valid" (no padding, leading to reduced output size).

2. Activation Function:

After each convolution operation, an activation function (usually ReLU – Rectified Linear Unit) is applied to introduce non-linearity into the model. This allows the network to learn complex patterns and relationships.

3. Pooling Layer:

- Max Pooling: This layer reduces the spatial dimensions of the feature map by taking the maximum value in each patch of the feature map. This operation helps in reducing the computational complexity and overfitting by downsampling the feature maps.

- Average Pooling: Instead of taking the maximum value, average pooling computes the average value of the elements in the patch. Max pooling is more commonly used due to better performance in practice.

4. Fully Connected Layer:

After several convolutional and pooling layers, the feature maps are flattened into a single vector and passed through one or more fully connected (dense) layers. These layers combine the features to make final predictions.

5. Dropout:

Dropout is a regularization technique used to prevent overfitting. During training, dropout randomly sets a fraction of the input units to zero, which forces the network to learn redundant representations and improves generalization.

Architecture:

A typical CNN architecture consists of a sequence of convolutional and pooling layers, followed by one or more fully connected layers. The final layer typically uses a softmax activation function to produce probability scores for classification tasks.

Training:

- Forward Propagation: During training, the input image is passed through the network, and intermediate feature maps are computed by the convolutional and pooling layers. The output of the network is compared to the ground truth labels using a loss function.
- Backpropagation: The loss is minimized using backpropagation, where gradients of the loss with respect to each weight are computed and the weights are updated using an optimization algorithm like Stochastic Gradient Descent (SGD) or Adam.

Challenges:

- Data Requirements: CNNs require large amounts of labeled data for training to achieve good performance.
- Computationally Intensive: Training CNNs is resource-intensive, often requiring powerful GPUs and large amounts of memory.
- Hyperparameter Tuning: Selecting the right architecture, including the number of layers, filter sizes, and other hyperparameters, is crucial and can be challenging.

5.4.3 Visual Geometry Group

The Visual Geometry Group (VGG) models are a series of convolutional neural networks (CNNs) developed by the Visual Geometry Group at the University of Oxford. VGG models gained significant attention and acclaim due to their performance in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014, where they achieved top results in image classification and localization tasks.

Key Features of VGG Models:

1. Simple and Uniform Architecture:

- VGG models are characterized by their simplicity and uniformity. They use small convolutional filters of size 3x3 throughout the network, which is the smallest size that captures spatial patterns while maintaining spatial resolution.

- The depth of the network is increased by adding more convolutional layers, which allows the network to learn more complex features.

2. Convolutional Layers:

- The network uses a series of convolutional layers with 3x3 filters. These layers are stacked together with increasing depth. The use of small filters ensures that the number of parameters is kept manageable, even for deeper networks.

- The convolutional layers are followed by ReLU activation functions, which introduce non-linearity and help the network learn complex patterns.

3. Pooling Layers:

Max pooling layers with a 2x2 filter size and a stride of 2 are used to reduce the spatial dimensions of the feature maps. Pooling helps in reducing the computational load and the number of parameters, as well as controlling overfitting.

4. Fully Connected Layers:

- The final layers of the VGG network consist of fully connected layers. Typically, there are two or three fully connected layers, each followed by a ReLU activation function.

- The last fully connected layer has a softmax activation function, which produces the probability distribution over the classes for classification tasks.

5. Depth and Configurations:

- VGG networks come in different configurations, such as VGG-16 and VGG-19, where the numbers denote the total number of layers in the network (including convolutional, fully connected, and pooling layers).

- VGG-16 consists of 13 convolutional layers, 3 fully connected layers, and 5 max pooling layers, totaling 16 weight layers.

- VGG-19 has 16 convolutional layers, 3 fully connected layers, and 5 max pooling layers, totaling 19 weight layers.

Training and Performance:

- Training: VGG models are trained on large-scale datasets like ImageNet. Training deep networks like VGG requires substantial computational resources, including powerful GPUs, and large amounts of labeled data. The models are trained using backpropagation and optimization algorithms such as Stochastic Gradient Descent (SGD) with momentum.

- Performance: VGG models have demonstrated excellent performance in image classification tasks. They are known for their high accuracy and ability to generalize well to unseen data. However, the increased depth and use of small filters come at the cost of higher computational complexity and memory usage.

Applications:

- Image Classification: VGG networks are extensively used for image classification tasks, achieving high accuracy on datasets like ImageNet.

- Feature Extraction: The deep convolutional layers of VGG networks make them excellent feature extractors for various computer vision tasks, such as object detection and image segmentation.

- Transfer Learning: Due to their strong performance and generalization capabilities, pretrained VGG models are often used for transfer learning. They serve as a base model,

and their learned features are fine-tuned for specific tasks with smaller datasets.

Advantages:

- Consistent Architecture: The uniform architecture with small 3x3 filters simplifies the network design and makes it easier to implement and experiment with different depths.

- High Performance: VGG models have set benchmarks for image classification accuracy, making them reliable choices for many computer vision tasks.

Challenges:

- Computational Resources: Training and deploying VGG models require significant computational resources, including high-end GPUs and large amounts of memory.

- Model Size: VGG models have a large number of parameters, making them memory-intensive and slower to deploy, especially on resource-constrained devices.

5.4.4 Inception Network

The Inception Network, also known as GoogLeNet, is a type of convolutional neural network (CNN) that was introduced by Google in 2014. It gained widespread recognition for its innovative architecture, which allowed it to achieve state-of-the-art performance on image classification tasks while being computationally efficient. The original Inception Network was followed by several improvements, leading to versions such as Inception-v2, Inception-v3, Inception-v4, and Inception-ResNet.

Key Features of Inception Networks:

1. Inception Module:

- The core idea behind the Inception Network is the Inception module, which aims to capture multi-scale features by performing convolutions with different filter sizes (1x1, 3x3, 5x5) within the same module.

- Each Inception module includes parallel convolutional layers with different filter sizes, allowing the network to capture both fine and coarse features.

- A 1x1 convolution is used before the larger convolutions to reduce the dimensionality, thereby reducing the computational cost and number of parameters.

2. Dimensionality Reduction:

The 1x1 convolutions within the Inception modules serve as bottleneck layers that reduce the number of input channels before performing the more computationally expensive 3x3 and 5x5 convolutions. This significantly reduces the computational burden and the number of parameters.

3. Pooling Layers:

- Inception modules also include pooling layers (usually max pooling) alongside the convolutional layers. The outputs of these different layers are concatenated to form the final output of the Inception module.

- Pooling layers help in reducing the spatial dimensions and capturing translation invariance.

4. Stacking of Inception Modules:

Multiple Inception modules are stacked together to form the deep network. This allows the network to learn increasingly complex and abstract features as the depth increases.

5. Auxiliary Classifiers:

- Inception Networks employ auxiliary classifiers during training. These are additional classification heads that branch off from intermediate layers of the network.

- The auxiliary classifiers provide additional gradient signals, which help in mitigating the vanishing gradient problem and improve the convergence of the network during training.

Improvements in Later Versions:

1. Inception-v2 and Inception-v3:

- Introduced batch normalization to improve training stability and speed.

- Replaced the 5x5 convolutions with two successive 3x3 convolutions to reduce computational cost.

- Added factorization of convolutions (e.g., a 3x3 convolution split into 1x3 and 3x1 convolutions) to further reduce computational complexity.

2. Inception-v4:

Combined ideas from Inception-v3 with residual connections (similar to ResNet) to improve gradient flow and training of very deep networks.

3. Inception-ResNet:

Integrated residual connections into the Inception architecture, which helped in training deeper networks by alleviating the vanishing gradient problem.

Training and Performance:

- Training: Training an Inception Network involves standard procedures such as forward propagation, loss computation, backpropagation, and optimization using algorithms like Stochastic Gradient Descent (SGD) with momentum or Adam. Batch normalization and auxiliary classifiers aid in training deep Inception Networks by improving gradient flow and convergence.
- Performance: Inception Networks have achieved top performance on several benchmark datasets, including ImageNet. They are known for their high accuracy, efficient use of computational resources, and ability to handle large-scale image data.

Applications:

- Image Classification: Inception Networks are widely used for image classification tasks due to their superior performance and ability to capture multi-scale features effectively.
- Object Detection: Inception Networks serve as backbone architectures for object detection models, such as Faster R-CNN and SSD (Single Shot MultiBox Detector).
- Image Segmentation: Inception Networks are employed in image segmentation tasks where accurate delineation of objects within an image is required.

Advantages:

- Multi-Scale Feature Extraction: The use of different filter sizes within the Inception modules allows the network to capture features at multiple scales, improving its ability to recognize objects of varying sizes.
- Computational Efficiency: The dimensionality reduction through 1x1 convolutions and factorized convolutions reduces the number of parameters and computational cost, making the network efficient.
- Flexibility: The modular structure of Inception Networks makes them flexible and easy to modify or extend for various tasks and applications.

Challenges:

- Complexity in Design: The design of Inception modules and the integration of various filter sizes and pooling operations can be complex and require careful tuning.
- Training Complexity: Despite the auxiliary classifiers and batch normalization, training very deep Inception Networks can still be computationally intensive and require significant resources.

5.4.5 Residual Network

Residual Networks (ResNets) are a type of deep neural network architecture introduced by Kaiming He and his colleagues in 2015. ResNets addressed the

problem of training very deep networks, which often suffer from the vanishing gradient problem, making it difficult for networks to learn and converge effectively. ResNets introduced a novel architecture that allowed them to train extremely deep networks with hundreds or even thousands of layers while maintaining high performance.

Key Features of ResNet:

1. Residual Learning:

- The fundamental idea behind ResNet is residual learning. Instead of trying to learn a direct mapping from the input to the output, ResNet learns a residual mapping. This is achieved by introducing shortcut connections (or skip connections) that bypass one or more layers.

2. Shortcut Connections:

- Shortcut connections are identity mappings that add the input of a layer to its output. This helps in preserving the flow of information and gradients throughout the network, effectively mitigating the vanishing gradient problem.

- Mathematically, if the input to a layer is \(x\) and the desired underlying mapping to be learned is \(H(x)\), ResNet reformulates the learning problem to \(F(x) = H(x) - x\). The original function then becomes \(H(x) = F(x) + x\).

3. Residual Blocks:

- A ResNet is composed of multiple residual blocks. Each block consists of a few convolutional layers and a shortcut connection. The typical structure of a residual block involves two or three convolutional layers with a ReLU activation function, followed by a batch normalization layer.

- Residual blocks can be expressed as:

\[
y = F(x, \{W_i\}) + x
\]

where \(x\) is the input, \(y\) is the output, \(F\) represents the residual mapping to be learned, and \(\{W_i\}\) are the weights of the layers.

4. Deep Networks:

- ResNet architectures vary in depth, with popular variants being ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152. The numbers indicate the total number of layers in the network.

- ResNet-50, for instance, uses 50 layers with bottleneck designs to balance computational efficiency and accuracy.

Training and Performance:

- **Training**: ResNets are trained using standard backpropagation and optimization techniques such as Stochastic Gradient Descent (SGD) with momentum or Adam. The presence of shortcut connections facilitates the training of very deep networks by improving gradient flow and convergence.

- Performance: ResNets have set new benchmarks in various computer vision tasks, including image classification, object detection, and image segmentation. Their ability to train extremely deep networks has led to significant improvements in accuracy on challenging datasets like ImageNet.

Applications:

- Image Classification: ResNets have achieved top performance on image classification tasks, making them a preferred choice for many applications.
- Object Detection: ResNets are used as backbone networks in object detection models like Faster R-CNN and YOLO (You Only Look Once), providing robust feature extraction.
- Image Segmentation: ResNets are employed in image segmentation tasks, where precise delineation of objects within images is required.
- Transfer Learning: Pretrained ResNet models are widely used for transfer learning. Their robust feature extraction capabilities make them suitable for fine-tuning on various tasks with limited data.

Advantages:

- Mitigation of Vanishing Gradients: The use of residual learning and shortcut connections effectively addresses the vanishing gradient problem, allowing for the training of very deep networks.
- Improved Accuracy: ResNets have demonstrated superior performance on many benchmarks, achieving high accuracy in image recognition tasks.
- Scalability: ResNets can be scaled to very deep architectures without significant degradation in performance, making them suitable for complex tasks.

Challenges:

- Complexity in Design: The design and implementation of very deep ResNets can be complex and computationally intensive.

- Resource-Intensive: Training and deploying very deep ResNets require substantial computational resources, including high-end GPUs and large memory.

6. Chapter Three – System Architecture

6.1 Overview of the System Architecture

The skin disease recognition system is designed with a layered architecture to facilitate modular development and integration. The primary layers include the following:

1. User Interface (UI) Layer
2. Image Processing Layer
3. Feature Extraction Layer
4. Machine Learning Model Layer
5. Database Layer
6. Integration Layer

6.2 Components of the System Architecture

6.2.1 User Interface (UI) Layer

The UI layer is the front-end of the application where users interact with the system. This layer includes:

- **Mobile Application:** A user-friendly mobile interface that allows users to capture or upload images of skin lesions.
- **Web Interface:** A complementary web platform for users who prefer accessing the system via a browser.
- **User Interaction:** Facilitates easy navigation, image submission, and displays results in an understandable format.

6.2.2 Image Processing Layer

The image processing layer handles the initial preparation of the uploaded images, ensuring they are suitable for feature extraction. This includes:

- **Image Preprocessing:** Techniques such as noise reduction, contrast enhancement, and normalization.

- **Segmentation:** Identifying and isolating the area of interest (skin lesion) from the background.

6.2.3 Feature Extraction Layer

This layer focuses on extracting meaningful features from the preprocessed images. Key techniques used are:

- **Histogram of Oriented Gradients (HOG):** Captures edge and shape information.

- **Texture Analysis:** Analyzes patterns and surface characteristics.

- **Color Variation Analysis:** Examines color distribution and anomalies.

6.2.4 Machine Learning Model Layer

The core of the system, this layer consists of trained machine learning models that classify skin diseases based on extracted features. It includes:

- **Support Vector Machine (SVM):** For initial classification tasks.

- **Convolutional Neural Network (CNN):** Advanced models like VGG, Inception, and ResNet for high accuracy.

- Model Training and Validation: Continuous learning from new data to improve model performance.

6.2.5 Database Layer

The database layer stores all relevant data, including:

- Image Database: Stores raw and preprocessed images.

- Feature Database: Maintains extracted features from images.

- Results Database: Keeps records of classification results and user interactions.

6.2.6 Integration Layer

This layer ensures smooth interaction between all components:

- API Management: Facilitates communication between the mobile app, web interface, and backend services.

- Data Flow Management: Manages the flow of data from image acquisition to result delivery.

6.3 Relationships Between Components

The components interact in a sequential manner:

- **Image Acquisition:** Users upload images via the mobile or web interface.
- **Image Processing:** The system preprocesses the images to enhance quality.
- **Feature Extraction:** Meaningful features are extracted from the images.
- **Classification:** Extracted features are fed into machine learning models for classification.
- **Result Display:** Classification results are presented to the user through the interface.
- **Data Storage:** Images, features, and results are stored in the database for future reference and continuous learning.

7. Chapter Four – System Implementation

7.1 Dataset

- Dataset size:25,352
- It contains 10 different diseases (classes)
- Data is collected from different resources like:previous datasets and medical sites as there is no dataset contains all the diseases
- This Dataset is considered as custom dataset
- This dataset is not used by previous projects but achieved high and acceptable accuracy
- This Dataset include the following Diseases:

- Acne
- Actinic Keratosis
- Eczema
- Herpes simplex
- Impetigo
- Molluscum Contagiosum
- Pityriasis Rosea
- Psoriasis
- Rosacea
- Tinea corporis

Figure 3 Sample images of Diseases included in the Dataset

7.2 Pre-Processing:

7.2.1 Data Splitting

The preprocessing phase begins with splitting the dataset into training and testing sets. This process is handled by the split_train_test function. Initially, it gathers a list of all image files in the specified directory and shuffles them randomly to ensure a balanced distribution of data. The function calculates a split index based on the given training ratio, dividing the list of files into training and testing subsets. It then creates separate 'train' and 'test' directories, if they do not already exist, to store the respective files. The files are moved accordingly, ensuring that the data is organized for subsequent steps in the workflow.

7.2.2 Data Loading

Following the data split, the images are loaded and prepared for model training using the load_data function. This function employs Keras' ImageDataGenerator to generate batches of tensor image data with real-time data augmentation. Specifically, the images are rescaled to have pixel values in the range [0,1], standardizing the data and facilitating more efficient training. The flow_from_directory method is then used to create Python generators that load images directly from the specified directory paths. These generators are set up for both training and testing data, with defined image sizes and batch sizes, ensuring a streamlined and consistent input pipeline for the model.

7.3 Feature Extraction

Preprocessing the dataset involves several critical steps to enhance the quality and consistency of the input images for the deep learning models. First, data augmentation is applied to the training dataset only, which includes techniques like rotation, flipping, and scaling. This process artificially increases the diversity of the training data, helping the model generalize better to new, unseen images. Next, noise removal is performed using a Gaussian filter, which helps to remove unwanted

artifacts such as hair that may obscure important features of the skin lesions. This step smooths the image by averaging pixel values with their neighbors, reducing noise while preserving edges. Sharpening is then applied to enhance the clarity of the image, making the features more distinct and easier for the model to learn. Image normalization follows, which involves scaling pixel values to a standard range, typically between 0 and 1, or normalizing to a mean of zero and a standard deviation of one. This standardization ensures that the model converges faster and performs better. Finally, the images are resized to a fixed size of 224x224 pixels, ensuring a uniform input size for the neural network. This resizing step is crucial for maintaining consistent input dimensions, which simplifies the processing and improves the efficiency of the training process.

7.4 Classification

7.4.1 Models' Parameters

- The pretrained ImageNet weights are used
- The Dataset is splitted to 80% train and 20% test and the validation data is 20% from the training data
- Learning rate=0.001 and number of epochs=10

7.4.2 Models Compilation and Training

- **Model Compilation:**

  The model compilation is handled by the compile_and_train function, which prepares the model for training by specifying the optimizer, loss function, and evaluation metrics. The Adam optimizer is selected for its efficiency and adaptive learning rate capabilities, which aid in faster convergence. The loss function used is categorical cross entropy, suitable for multi-class classification tasks. Accuracy is chosen as the primary evaluation metric to track the model's performance during training.

- **Model Training:**
  Training the model involves fitting it to the training data using the fit method. The model iterates over the training data for a specified number of epochs, learning to minimize the loss function. Validation is conducted on the test data after each epoch, allowing for continuous monitoring of the model's performance and making adjustments as necessary. This iterative process ensures that the model effectively learns to distinguish between the different classes based on the provided training data.

7.4.3 Tried Models and their Results

- Many deep learning models were tried such as Inception, Xception, VGG19, ResNet, custom CNN and MobileNet
- After many attempts and experiments to improve the models' performance the following results were obtained (best results):

  ➔ MobileNet test and validation accuracy: 92% (best accuracy)
  ➔ Xception model: 90% (second best accuracy)
  ➔ Inception model: 88% (third best accuracy)

- Due to the previous results a combination of the best three models (inception, Xception, MobileNet) was used and considering the average of accuracies as the final accuracy which is 91%.

Figure 4 Accuracies of the Tried Models

7.4.4 Used Models' Architecture

7.4.4.1 MobileNet:

MobileNet is a family of efficient convolutional neural networks (CNNs) designed specifically for mobile and embedded vision applications where computational resources and power consumption are critical constraints. Introduced by Google, MobileNet models utilize depthwise separable convolutions, which significantly reduce the number of parameters and computational complexity compared to standard convolutional layers. This approach decomposes a standard convolution into a depthwise convolution, which filters the input channels, and a pointwise convolution, which combines these filtered channels. As a result, MobileNet achieves high accuracy with a reduced model size, making it ideal for real-time applications on devices with limited processing power, such as smartphones and IoT devices.

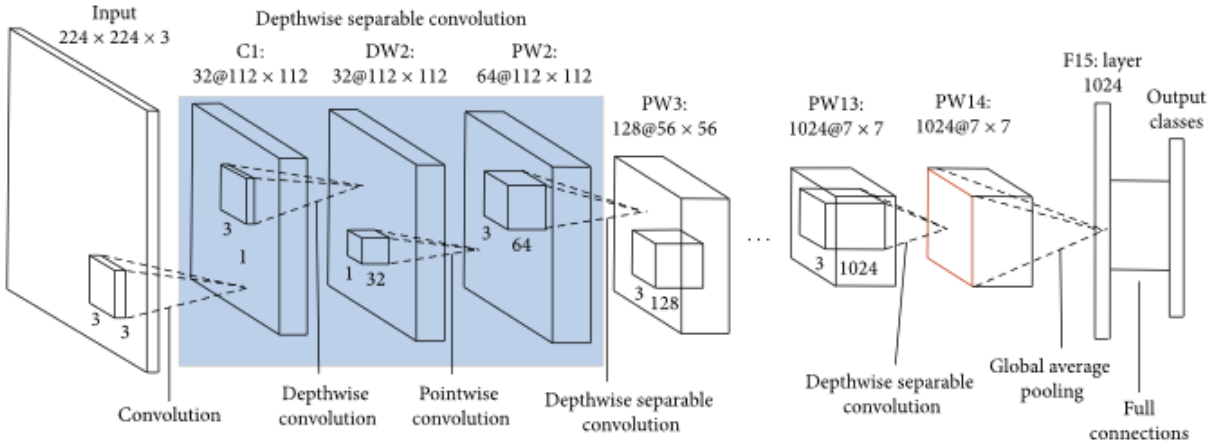


Figure 5 MobileNet Architecture

7.4.4.2 Inception:

Inception, also known as GoogLeNet, is a deep convolutional neural network that introduces the concept of an Inception module to improve computational efficiency and accuracy. The key innovation of the Inception architecture is its ability to capture multi-scale features by performing convolutions with different kernel sizes (1x1, 3x3, and 5x5) in parallel, along with max-pooling operations. These operations are then concatenated to form the output of the Inception module. This design allows the network to learn more complex and varied features without a significant increase in computational cost. Inception has multiple versions, each building on the previous one by incorporating various improvements such as batch normalization, label smoothing, and more complex modules, making it one of the most influential architectures in the field of deep learning.

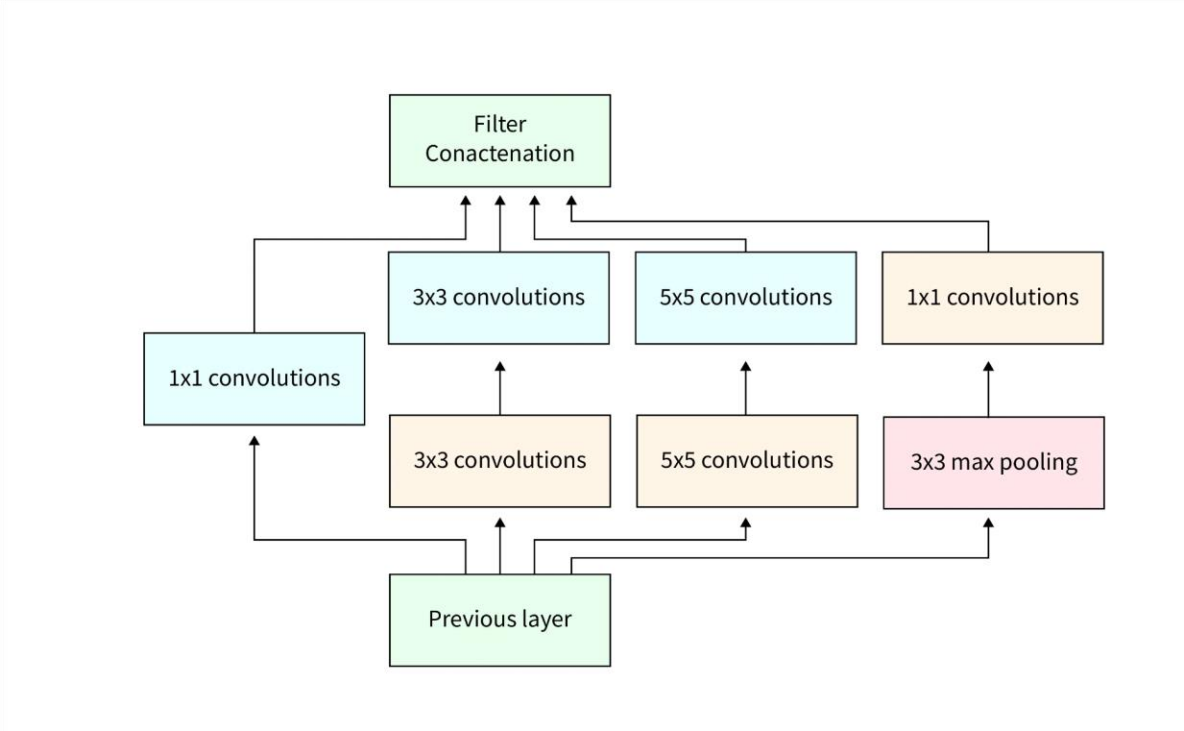


Figure 6 Inception Model Architecture

7.4.4.3 Xception:

Xception, short for "Extreme Inception," is an advanced convolutional neural network architecture developed by Francois Chollet. It builds upon the Inception architecture by replacing the standard Inception modules with depthwise separable convolutions, similar to those used in MobileNet. This modification leads to a simpler yet more efficient design that maintains the model's ability to learn complex representations while reducing the computational load. Xception treats the convolution process as a two-step operation: first, a depthwise convolution filters each input channel independently, and then a pointwise convolution mixes these channels to produce the output. This decoupling of spatial and cross-channel correlations allows Xception to achieve superior performance on image classification tasks while being computationally efficient, making it a powerful alternative to traditional Inception-based networks.

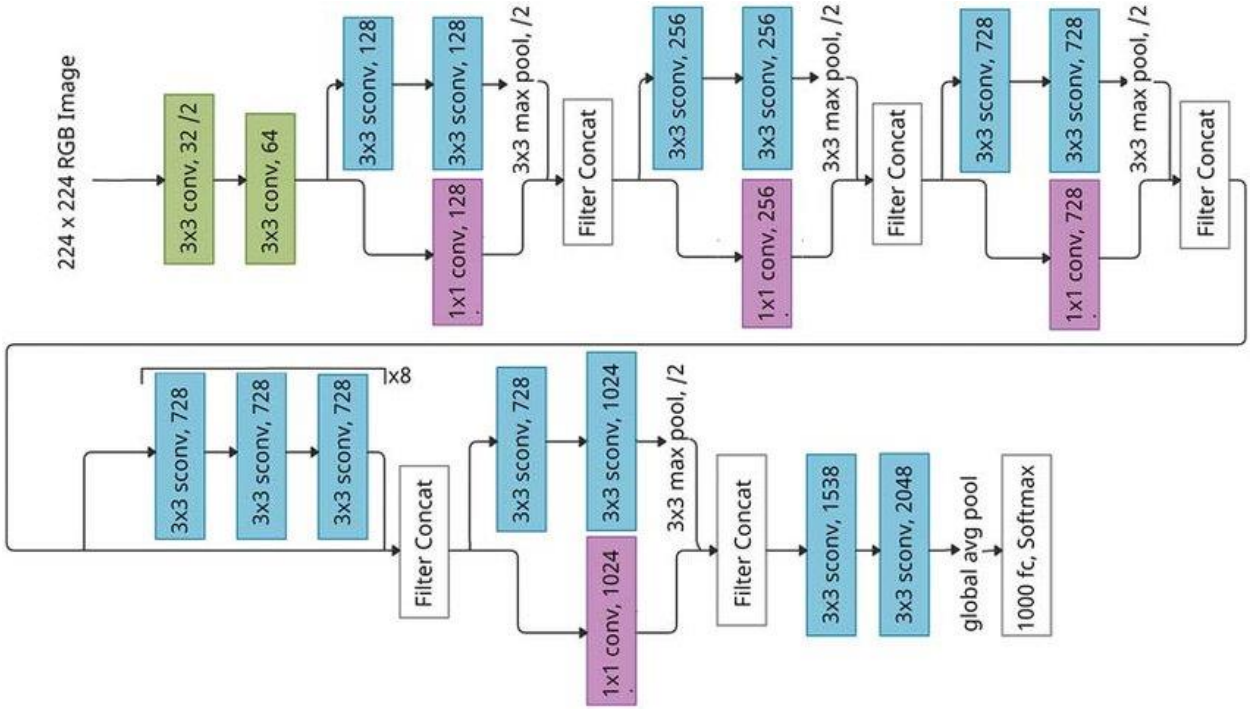


Figure 7 Xception Model Architecture

8. Chapter Five – System Testing

8.1 Running the Application

8.1.1 Starting the Server

- Navigate to the application directory.
- Run the server

8.1.2 Accessing the Application

- Open a web browser and navigate to http://localhost:5000.
- The home screen of the application should be displayed.

8.2 Testing the Application

8.2.1 Uploading an Image

- Click on the "Upload Image" button on the home screen.
- Select an image of a skin lesion from your computer and click "Open."
- The selected image will be displayed on the screen.

8.2.2 Running the Diagnosis

- After the image is uploaded, click on the "Diagnose" button.
- The application will process the image and display the diagnosis results, including the identified disease and confidence level.

8.2.3 Reviewing Results

- Review the diagnosis results displayed on the screen.
- The application should show the name of the disease, probability scores, and any relevant suggestions or next steps.

8.3 Test Cases

8.3.1 Functional Tests

- **Test Case 1:** Upload a clear image of a known skin disease and verify if the application correctly identifies it.
- **Test Case 2:** Upload an image with noise or poor lighting to test the robustness of the preprocessing step.

8.3.2 Performance Tests

- Measure the time taken for the application to process and diagnose multiple images.
- Ensure the application can handle simultaneous requests without crashing.

8.3.3 User Interface Tests

- Verify that all buttons and links are working as expected.
- Check for responsiveness of the application on different devices (e.g., desktop, tablet, mobile, … etc.).

8.3.4 Error Handling

- Upload an unsupported file type and ensure the application provides an appropriate error message.
- Test the application with no image uploaded and ensure it prompts the user to upload an image.

8.4 Test Results

- Document the results of each test case.
- Note any bugs or issues encountered during testing.
- Include screenshots of successful tests and error messages for failed tests.

9. Chapter Six – Conclusion and Future Work

9.1 Conclusion

Addressing the problem of skin diseases and the associated challenges has been a rigorous yet fulfilling journey. The core issue was identified as the significant impact of skin diseases on individuals' confidence and social interactions, often leading to emotional distress and limited daily activities. The solution focused on providing an accessible, reliable, and cost-effective method for diagnosing and understanding skin diseases through a mobile application. This system empowers patients with accurate information and treatment options, aiming to reduce the need for expensive and time-consuming medical consultations.

The development of the solution faced several challenges, notably the lack of a comprehensive and readily available dataset. Substantial time and effort were invested in gathering and curating a dataset that encompasses a wide range of skin diseases, ensuring the robustness of the model. Additionally, the similarity in the appearance of different skin diseases posed a significant hurdle, requiring experimentation with multiple deep learning models to achieve optimal accuracy. The choice of MobileNet, Xception, and Inception architectures, combined with model averaging, resulted in a commendable accuracy rate, demonstrating the efficacy of this approach.

Table 1 Advantages and Disadvantages of the Discussed Solution

Advantages	Disadvantages
Provides a cost-effective and easy-to-use solution for diagnosing skin diseases.	Potential for inconsistencies and biases in the custom-built dataset.
Utilizes advanced deep learning models (MobileNet, Xception, Inception) to achieve high accuracy.	High computational demands for training and inference, limiting accessibility on lower-end devices.
Offers patients knowledge about the disease and potential treatments.	Difficulty distinguishing between skin diseases with similar appearances, necessitating continuous updates and improvements.
Includes a wide range of skin diseases by aggregating data from various sources.	Need for ongoing medical validation to ensure alignment with clinical standards.
Designed for use on mobile devices, making it convenient and widely accessible.	

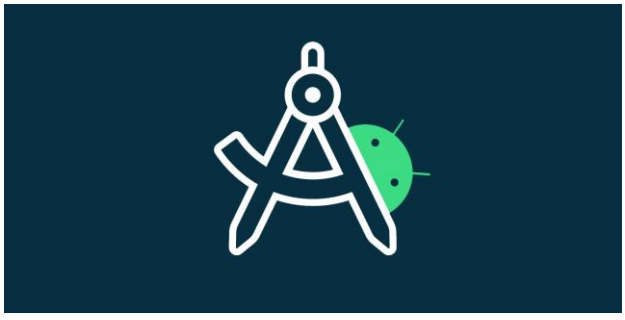
9.2 Future Work

- More Diseases can be added
- Skin care section can be added
- Provide a search option which make the user able to search for a certain disease instead of uploading an image
- Explore integration with wearable health devices to provide real-time monitoring and diagnosis of skin conditions.
- Implement support for multiple languages to make the application accessible to a global audience.
- Partner with dermatologists to validate the model's predictions and ensure clinical accuracy and reliability.
- Expand the system to include personalized treatment recommendations based on user-specific factors such as age, skin type, and medical history.
- Enable users to track the progress of their skin conditions over time, providing insights into the effectiveness of treatments and any changes in the condition.

10. Tools and References

10.1 Tools

- Python
- Convolutional Neural Network
- OpenCv
- TensorFlow
- Flutter
- TensorFlow Lite
- Android Studio
- Kaggle
- Mobile phone
- Colab



10.2 References

- [1]Sadik et al.,"An in-depth analysis of Convolutional Neural Network architectures with transfer learning for skin disease diagnosis",Department of Computer Science and Engineering, Jahangirnagar University, Savar, Dhaka, Bangladesh,Department of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh,November 2023,vol. 3
- [2] Abdul Rafay,Waqar Hussain, "EfficientSkinDis: An EfficientNet-based classification model for a large manually curated dataset of 31 skin diseases",Department of Computer Science, School of Systems and Technology, University of Management and Technology, Lahore, Pakistan,Department of Artificial Intelligence, School of Systems and Technology, University of Management and Technology, Lahore, Pakistan,August 2023,Vol.85
- [3]Inthiyaz et al.,"Skin disease detection using deep learning",Advances in Engineering Software,January 2023,Vol.175
- [4] Li et al.,"Image Analysis and Diagnosis of Skin Diseases",May 2022, Vol.19,pp 199-242
- [5] Sara Medhat et al., "Skin cancer diagnosis using convolutional neural networks for smartphone images: A comparative study",Radiation Engineering Department, National Center for Radiation Research and Technology, Egyptian Atomic Energy Authority, Cairo, Egypt,Computer Science Department, Faculty of Computers and Artificial Intelligence, Helwan University, Cairo, Egypt,March 2022,Vol.15(1),pp 262-267
- [6] Pooja Pathak et al., "Identification of Skin Diseases Using Convolutional Neural Network",GLA University(Springer, Singapore), 2021 ,pp 171-180
- [7] Anika et al., "Skin Disease Recognition: A Machine Vision Based Approach.",7th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India,2021,pp. 1029-1034, doi: 10.1109/ICACCS51430.2021.9441980.
- [8] Elngar et al., "Intelligent System for Skin Disease Prediction using Machine Learning ",Journal of Physics Conference Series,August 2021,DOI:10.1088/1742-6596/1998/1/012037

- [9] Srinivasu et al., "Classification of Skin Disease Using Deep Learning Neural Networks with MobileNet V2 and LSTM",Department of Computer Science and Engineering, Gitam Institute of Technology, GITAM Deemed to be University, Rushikonda, Visakhapatnam 530045, India,Tata Consultancy Services, Gachibowli, Hyderabad 500019, India,Department of Intelligent Mechatronics Engineering, Sejong University, Seoul 05006, Korea,April 2021,doi: 10.3390/s21082852
- [10]Priyanti Paul Tumpa, Md Ahasan Kabir,"An artificial neural network based detection and classification of melanoma skin cancer using hybrid texture features",Sensors International,October 21,Vol.2
- [11] Wang Sidi, Zou Qiang ,"Skin disease recognition and detection box",2019
- [12] Joshi et al.,"Skin Disease Detection And Classification",International Journal of Advanced Engineering Research and Science,May 2019,Vol. 6(5),pp 396-400
- [13] Li-sheng Wei, Quan Gan, Tao Ji,"Skin Disease Recognition Method Based on Image Color and Texture Features",Computational and Mathematical methods in medicine,August 2018
- [14] S.kolkur et al.," Human Skin Detection Using RGB, HSV and YCbCr Color Models",2017
- [15]R.Sumithra et al., "Segmentation and Classification of Skin Lesions for Disease Diagnosis",International Conference on Advanced Computing Technologies and Applications (ICACTA),2015,Vol.45,pp 76-85
- [16]liansheng et al.,"Diagnostic applicability of confocal laser scanning microscopy in tinea corporis",National library of Medicine, October 2013,doi: 10.1111/j.1365-4632.2011.05144
- [17] S.Arivazhagan et al.,"Skin Disease Classification by Extracting Independent Components",Journal of Emerging Trends in Computing and Information Sciences,October 2012,Vol.3
- [18] Ranjan Parekh,AK Mittra,"Automated Detection of Skin Diseases using Texture Features",International Journal of Engineering Science and Technology,June2011