Istanbul Arel University
Faculty of Engineering and Architecture
Computer Engineering Department

SKIN DISEASE AND MOISTURE CLASSIFICATION USING THE HAM1000 DATASET AND CONVOLUTIONAL NEURAL NETWORKS

A project submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Engineering

by

Muhammad Akbar (210303894)

Supervised by

Tuğberk Kocatekin

ACKNOWLEDGEMENTS

I, Muhammed AKBAR, would like to extend my heartfelt thanks to all those who have played a part in the successful completion of my project.
My deepest gratitude goes to our distinguished mentor, Asst. Prof. Tuğberk Kocatekin. His expert guidance, invaluable advice, and constant encouragement have been instrumental to the progress and success of this project. I am deeply appreciative of their unwavering support, willingness to always make time for my inquiries, and their thoughtful feedback. His dedication to our growth and open-door approach have been a true source of inspiration throughout this journey.I am also grateful to my professors, peers, and colleagues who shared their knowledge and perspectives with us along the way. Their constructive feedback, thoughtful suggestions, and words of encouragement have been essential in shaping the final outcome of this project.

Finally, I wish to acknowledge everyone who contributed to our success, whether directly or indirectly. Your support, both seen and unseen, has been vital to the completion of this work.

Thank you all for being an integral part of this journey. Your contributions have made all the difference.

UNDERTAKING

This is to declare that the project entitled "SKIN DISEASE AND MOISTURE CLASSIFICATION USING THE HAM1000 DATASET AND CONVOLUTIONAL NEURAL NETWORKS" is an original work done by undersigned, in partial fulfillment of the requirements for the degree "Bachelor of Science in Computer Engineering" at Computer Engineering Department, Faculty of Engineering and Architecture, Istanbul Arel University.

All the analysis, design and system development have been accomplished by the undersigned. Moreover, this project has not been submitted to any other college or university.

Muhammad Akbar

ABSTRACT

The skin, a vital organ that acts as a protective barrier for the human body, is prone to various conditions caused by fungi, viruses, environmental factors like dust, and intrinsic properties such as moisture levels. Conditions like acne, eczema, and skin cancer, as well as variations in skin moisture (dry, normal, or oily), can lead to significant physical and emotional distress. Accurate diagnosis of both skin diseases and moisture-related characteristics is essential for effective treatment and personalized skincare, alleviating the suffering of those affected. This study aims to develop a robust system for the automated detection and classification of skin diseases and facial skin types using Convolutional Neural Networks (CNNs), which are well-suited for image analysis due to their ability to learn complex visual patterns.

The proposed system employs advanced image processing techniques, allowing users to upload photos of affected skin areas or facial images for analysis. These images are processed on a central server, where CNN models classify both skin diseases and moisture levels, providing diagnoses and skin type assessments. The application focuses on enhancing diagnostic precision and efficiency, addressing the growing demand for automated solutions to assist medical practitioners and skincare professionals in expediting treatment plans and recommendations. For skin diseases, the system targets seven types of lesions, including melanoma, using the HAM10000 dataset, a benchmark collection of diverse dermatoscopic images. For skin moisture classification, a curated dataset of 1074 images, manually verified for accurate labeling of dry (294 images), normal (319 images), and oily (459 images) skin types, is used to ensure reliable performance.

Skin diseases and improper moisture levels are global health and cosmetic concerns, often posing risks that can lead to severe physical and mental health consequences, including skin cancer or chronic skin irritation. Diagnosing these conditions from clinical or facial images is challenging due to the complexity and variability of skin lesions and the subtle *differences in moisture-related features like texture and sheen. Manual diagnosis, while effective, can be subjective, time-consuming, and dependent on expert availability, underscoring the need for automated systems. For skin moisture classification, the challenge is amplified by the lack of standardized datasets and methods tailored to distinguishing dry, normal, and oily skin, making high-quality data curation critical.*

This study integrates deep learning with collective intelligence to develop an advanced framework for both skin lesion and moisture classification. By leveraging multiple CNNs trained on the HAM10000 dataset for disease classification and a curated 1074-image dataset for skin type classification, the system achieves high accuracy in identifying skin conditions and moisture levels. The hierarchical learning capabilities of CNNs enable precise differentiation of complex visual patterns, contributing to early detection of diseases, efficient treatment planning, and personalized skincare recommendations, ultimately improving outcomes for patients and individuals worldwide.

Keywords: Skin Disease Classification, Skin Moisture Classification, Convolutional Neural Networks (CNNs), Deep Learning, Skin Lesions, Facial Skin Types, HAM10000 Dataset

TABLE OF CONTENTS

1. INTRODUCTION 1
1.1. Purpose of the Study 2
2. RELATED WORKS 4
3. METHODOLOGY 7
3.1. Datasets 7
3.1.1. Dataset Overview for skin disease classification 7
3.1.2. Dataset overview for skin moisture classification 8
3.2. Model Overview 9
3.3. Deployment Platform Overview 9
4. Data Preprocessing 11
4.1. Data Preprocessing for skin disease classification 11
4.1.1. Image Resizing 11
4.1.2. Data Splitting 11
4.1.3. Data Normalization 11
4.1.4. Data Augmentation 11
4.2. Data Preprocessing for Skin Moisture Classification 12
4.2.1. Random Resized Cropping 12
4.2.2. Data Splitting 12
4.2.3. Data Augmentation and Normalization (Training) 12
4.2.4. Conservative Preprocessing (Validation) 12
5. ARCHITECTURE 14
5.1. Standard CNN Model 14
5.1.1. Architecture of the Standard CNN Model 14
5.1.1.1. Input Layer 14
5.1.1.2. Convolutional Layers 14
5.1.1.3. Pooling Layers 15
5.1.1.4. Fully Connected Layers 15
5.1.1.5. Output Layer 16
5.1.1.6. Adam Optimizer 16
5.1.1.7. Other Optimizers vs Adam 17
5.1.2. Advantages of the Standard CNN Model 17
5.2. Pre-trained image classification models 18
5.3. Architecture of Pretrained CNN Models 18
5.3.1. VGG16 18
5.3.2. ResNet (Residual Networks) 19
5.3.3. DenseNet (Dense Convolutional Networks) 20
5.3.4. EfficientNet 20
5.3.4 Training Procedure for Standard CNN Model 22

5.4. Training Procedures for Skin Disease Classification Using ResNet, VGG16, and DenseNet 23
5.5. Training Procedures for Skin Moisture Classification Using ResNet and EfficientNet 24
5.6. Mathematical Formulations 25
5.6.1. Adam Optimizer 25
5.7. Evaluation Metrics 25
5.7.1. Precision 25
5.7.2. Recall (Sensitivity) 26
5.7.3. Accuracy 26
6. IMPLEMENTATION 27
6.1. Skin Disease Classification (HAM10000) 27
6.1.1. Standard CNN Model 27
6.1.2. VGG16 27
6.1.3. ResNet 27
6.1.4. DenseNet 27
6.2. Skin Moisture Classification (Facial Skin Type) 28
6.2.1. ResNet50 28
6.2.2. EfficientNetB0 29
7. Analysis of Model Performance 31
7.1. Skin Disease Classification (HAM10000 Dataset) 31
7.2. Skin Moisture Classification 32
7.3. Limitations 32
7.3.1. Skin Disease Classification 32
7.3.1.1. Imbalanced Dataset 32
7.3.2. Skin Moisture Classification 34
8. MODEL DEPLOYMENT on ANDROID APP 35
8.1. Kivy App with Minimal Functionality 35
8.1.1. Kivy 35
8.2. Full-Stack Healthcare Application 35
8.2.1. React Native for Client Side 35
8.2.1.1 Other Optimizers 36
Table 8.1: Comparison of React Native and flutter 36
8.2.1.2. Advantages of React Native 37
8.2.2. Backend Technologies 37
8.2.5. Database 38
8.2.4. App Integration & Full-Stack Flow 38
8.3. Advantages and Challenges 40
9.4. Future Work 40
9. CONCLUSION 42
10. REFERENCES 43

LIST OF FIGURES

Figure 1.1: Example of skin lesions used for classification.
Figure 1.2: Dry vs Oily skin
Figure 3.1: Example of each class from HAM10000 dataset
Figure 3.2: Example of images from Skin moisture classification
Figure 4.1: Vertical or horizontal pixel shift with a maximum of 10
Figure 4.2: Steps of data preprocessing
Figure 5..1: Illustration of convolutional layers extracting features
Figure 5.2: Illustration of a functioning Max Pooling layer
Figure 5.3: Fully Connected Layer
Figure 5.4: All Layers performing to give results that contribute in predicting the class using output layer
Figure 5.5: VGG16 Architecture
Figure 5.6: resnet Architecture
Figure 5.7: densenet Architectur
Figure 5.8: EfficientNet Architecture
Figure 7.1: Mislabelled visually oily skin person's image labelled as dry
Figure 8.2: Normal workflow of our application
Figure 8.3: workflow of our application in terms of RUST Framework

LIST OF TABLES

Table 2.1: Detailed summary of all related works regarding to our study

Table 3.1: Classes in HAM1000 with number of samples

Table 5.1: Comparison of Adam with other widely used optimizers

Table 5.2: comparison of standard CNN and pretrained transform learning model in

Table 5.3 : summarized structure and key features of the transform learning models used

Table 6.1 and 6.2 : the training and validation results of each model respectively

Table 6.3: Results of different models when ran on different sized datasets

Table 8.1: Comparison of React Native and flutter

1. INTRODUCTION

Skin diseases, from mild issues to serious, life-threatening conditions, are a major global health problem, affecting millions of people. Early and accurate diagnosis is key, as quick treatment often makes all the difference in how well patients recover. Traditional methods, while helpful, require a lot of expertise and time, which can be tough in places with limited resources. This shows why we need smart, automated tools to make diagnosis faster and more accurate. The World Health Organization's First Global Meeting on Skin-Related Neglected Tropical Diseases (skin NTDs) in March 2023 in Geneva, Switzerland, highlighted the huge impact of skin conditions, estimating that around 1.1 billion people are affected at any time [1]. Skin infections, whether bacterial, viral, fungal, or parasitic, are some of the most common diseases in tropical and low-resource areas. In many communities, skin NTDs make up about 10% of all skin diseases [2]. These findings push for community-based approaches in high-risk areas to tackle skin NTDs and other skin issues as part of Universal Health Coverage (UHC) and the Sustainable Development Goals (SDGs), ensuring everyone gets care [3]. Convolutional Neural Networks (CNNs) have become a game-changer for automating skin disease diagnosis over the last decade. They can learn complex patterns from medical images, helping spot and classify skin lesions early [4]. Accurate detection and classification of skin lesions are critical, especially for catching melanoma, one of the deadliest skin cancers [5]. Early detection boosts treatment success, while delays can lead to cancer spreading, sometimes with severe outcomes like death.

Skin cancers fall into four main types: basal cell carcinoma, squamous cell carcinoma, Merkel cell carcinoma, and melanoma. Melanoma is the most dangerous, causing the most deaths. Even though its overall death rate is around 10%, treatment can be painful, and late diagnosis can lead to serious issues, like amputation [6]. Cancer rates are expected to rise significantly, with projections of a 24.1% increase in men and 20.6% in women [7]. Automated systems, especially those using neural networks, are becoming essential in dermatology. A review of neural network-based systems for melanoma detection shows how artificial intelligence is improving diagnostic speed and accuracy [8]. The HAM10000 dataset, with 10,000 dermatoscopic images of various skin conditions, is a key tool for training and testing machine learning models for skin lesion classification [9]. Despite progress, building systems to classify multiple skin disease types is still tough. Past studies often focused on one disease or struggled with the subtle differences between skin conditions [10]. This study uses the HAM10000 dataset to develop a CNN model that can accurately classify multiple skin lesion types like those shown in Figure 1.1, aiming to fill these gaps. By improving diagnostic accuracy, this work seeks to advance early detection and treatment of skin diseases worldwide.

We will also shortly mention another skin image classification problem which is oily vs dry skin image classification. Automating facial skin type classification using deep learning enables scalable, non-invasive skin analysis for dermatological diagnostics and skincare recommendations. This study addresses challenges in training a robust classifier by shifting from large, noisy datasets to a smaller, high-quality dataset of 1074 images, manually curated for label accuracy across three classes: Dry, Normal, and Oily. Using a pre-trained ResNet18 model with tailored augmentations and transfer learning, we achieved usable validation accuracy. Our findings underscore the importance of data quality over quantity, demonstrating that careful dataset curation, preprocessing, and model tuning significantly enhance generalization and stability in skin type prediction tasks.

Although both studies are not much related to each other, we applied different deep learning techniques to analyze how well our models perform related to human skin related issues. Thisstudy comprehensively examines performance of CNN models on different problems.

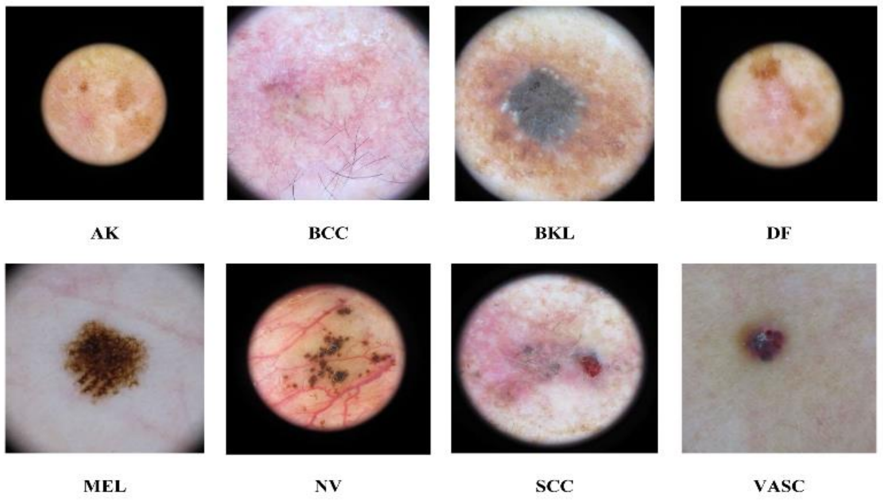

Figure 1.1: Example of skin lesions used for classification.

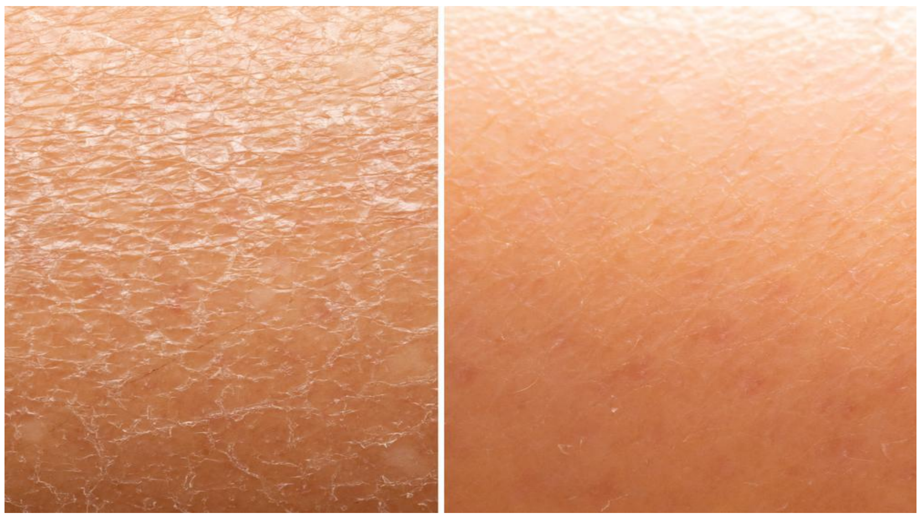

Figure 1.2: Dry vs Oily skin

1.1. Purpose of the Study

The purpose of this study is to develop and evaluate a Convolutional Neural Network (CNN) model capable of accurately classifying multiple skin lesion categories using the HAM10000 dataset. By addressing existing limitations in automated diagnostic systems such as their focus on single diseases or challenges in distinguishing nuanced similarities between skin conditions, this study seeks to advance the field of dermatology through the application of innovative deep learning techniques. Ultimately, this research aims to enhance early detection and diagnostic precision for a broad range of skin diseases, improving treatment outcomes and contributing to global health initiatives like Universal Health Coverage (UHC) and the Sustainable Development Goals (SDGs). The proposed model aspires to provide a robust and scalable solution, particularly for resource-constrained and endemic regions, where early and accurate diagnosis is critical for effective healthcare delivery.

This application also includes an additional model for classifying skin types as oily or dry. This provides users with a more personalized skin care experience, allowing for tailored recommendations that go beyond simple disease prediction. By integrating these two models, the app offers a holistic dermatological solution, helping users understand both potential skin conditions and their skin’s fundamental characteristics.

2. RELATED WORKS

The ISIC Archive, developed by the International Skin Imaging Collaboration, provides a comprehensive repository of thousands of dermoscopic images accompanied by expert annotations. This dataset is a cornerstone for skin lesion analysis, widely utilized in challenges focused on segmentation and classification of skin lesions. Its extensive collection and high-quality annotations make it an invaluable resource for training and validating AI algorithms in dermatology, enabling researchers to address diverse lesion types and improve diagnostic accuracy [24].

The PH2 dataset comprises 200 dermoscopic images, primarily focused on melanocytic lesions, and is designed to support research in lesion border detection and classification. Its curated collection offers detailed annotations, making it a popular choice for developing and testing algorithms aimed at precise lesion segmentation and diagnostic tasks. The dataset's emphasis on melanocytic lesions provides a specialized resource for advancing AI-driven dermatological research [25].

DermNet NZ offers a broad collection of clinical images capturing various skin diseases in real-world clinical settings. Unlike dermoscopy-focused datasets, it includes photographic images, broadening the scope of research to include non-dermoscopic applications. This dataset supports the development of AI models for diagnosing a wide range of skin conditions, providing a practical resource for real-world clinical scenarios [26].

The human skin, as the body's largest organ, is susceptible to a wide range of diseases, making accurate diagnosis both critical and challenging, particularly in resource-scarce regions. Artificial intelligence has emerged as a transformative tool in addressing this challenge, evolving from traditional methods like handcrafted feature extraction with classical classifiers to advanced deep learning techniques. Convolutional Neural Networks (CNNs) have demonstrated exceptional performance in dermatology, as evidenced by numerous studies highlighting their efficacy.

One study combined saliency-based segmentation with a 16-layer CNN and leveraged transfer learning with DenseNet201 to achieve high accuracy in skin lesion classification [11]. This work exemplifies how integrating segmentation, deep learning, and transfer learning can significantly enhance model performance. A comprehensive review further underscores CNNs as powerful tools for image classification, particularly in recognizing multiple skin lesions from dermoscopic images [12], solidifying their role as a cornerstone in dermatological diagnostics.

A landmark study trained a deep learning model on over 129,000 clinical images, demonstrating that CNNs can match the diagnostic accuracy of board-certified dermatologists in identifying skin cancer [13]. This achievement highlights the potential of AI to rival human expertise in medical diagnostics. Beyond dermatology, deep learning has made significant strides in other medical domains. For example, ResNet and VGG architectures have improved brain tumor segmentation [14], while the U-Net architecture has enhanced liver tumor segmentation on the LiTS dataset [15]. Similarly, CNNs with bio-inspired techniques have been applied to detect mitotic nuclei in breast tissue images, aiding breast cancer diagnosis [16].

In early Alzheimer's detection, complex pipelines combining image features with optimization algorithms have enabled earlier diagnosis, improving treatment outcomes [17]. Custom deep learning models have also outperformed pre-trained architectures like VGG16 and ResNet50 in brain tumor classification, achieving superior accuracy [18]. Ensemble learning approaches, such as bagging with K-Nearest Neighbor and stacking, have improved disease prediction by addressing data imbalances [19].

Another significant study using over 129,000 clinical images further confirmed that CNNs can achieve dermatologist-level performance in skin cancer identification [20]. A review of deep learning in dermatology emphasized the necessity of large, diverse datasets to maximize CNN performance [21], highlighting the importance of datasets like ISIC, PH2, and DermNet NZ. In melanoma classification, CNNs have surpassed human performance, with one model outperforming 58 dermatologists in diagnostic accuracy using dermoscopic images [22].

Researchers have also enhanced skin lesion classification by integrating traditional ABCDE rules (Asymmetry, Border, Color, Diameter, Evolution) with machine learning techniques like Support Vector Machines [23], effectively combining clinical expertise with AI capabilities. Transfer learning, where pre-trained deep neural networks are fine-tuned for skin lesion images, has further improved accuracy and reliability [24]. Ensemble models that combine segmentation and classification networks have gained popularity for their ability to capture complex features like color, shape, and texture, outperforming single-network approaches [25].

The Inception-v3 model from GoogLeNet has achieved dermatologist-level accuracy in skin cancer classification [26], demonstrating the enduring effectiveness of established CNN architectures. Even older models like AlexNet have delivered impressive results, achieving over 93% accuracy in mole detection systems [27]. Residual networks such as ResNet-50 and ResNet-101, along with efficient models like Xception and MobileNet-V2, have reached approximately 90.7% accuracy in melanoma classification when combined with handcrafted features [28], offering a balance of accuracy and computational efficiency.

DenseNet-201, with its densely connected layers, facilitates feature reuse and has achieved over 81% accuracy on datasets like ISIC 2017 when paired with fully convolutional networks for segmentation [29]. Hybrid models combining architectures like ResNet and AlexNet in ensemble voting systems have improved predictions for various skin lesion types, including melanoma [30]. Finally, sophisticated decision fusion methods that integrate outputs from multiple CNNs, using techniques like majority voting or confidence-weighted averages, have enhanced classification reliability for distinguishing melanoma from non-melanoma lesions [31].

Table 2.1: Detailed summary of all related works regarding to our study

Application Area	**References**
Skin Lesion Classification	[11, 12, 23, 24, 25, 28, 29, 30, 31]
Skin Cancer Diagnosis	[13, 20, 26]
Dermatology Review	[21]
Melanoma Classification	[22]
Mole Detection	[27]
Brain Tumor Segmentation	[14]
Brain Tumor Classification	[18]
Liver Tumor Segmentation	[15]
Breast Histopathology	[16]
Alzheimer’s Disease Detection	[17]
Disease Prediction	[19]

3. METHODOLOGY

Lets dive into examination of our structured methodology that we will use till the end of project, we had to decide the right model, dataset and platform to deploy our models on, the details of those are given as follows

3.1. Datasets

Finding appropriate datasets was a tiring task, to get a get accuracy on both validation and training we must ensure we have enough training images so that the model can learn features more precisely, keeping these things in mind we round with estimated to be best datasets for our study which are discussed in the following sections 3.1.1 and 3.1.2

3.1.1. Dataset Overview for skin disease classification

The HAM10000 was created by researchers primarily from the Medical University of Vienna and the University of Vienna, along with collaborators. HAM1000 dataset is a publicly available collection of 10,000 dermatoscopic images curated to facilitate research in skin lesion classification.. It's a widely used public dataset released to support research in melanoma and other skin lesion classification. It contains 10,015 dermatoscopic images covering seven types of pigmented skin lesions, including melanoma, benign nevi, and other important categories as seen in Figure 2.1. The dataset is accompanied by metadata and is divided into training, validation, and test sets, enabling robust evaluation of machine learning models. It also provides high-quality labeled images, enabling the development and benchmarking of machine learning models for automated skin cancer classification [23]. HAM10000's diverse collection helps address issues of class imbalance and variability in lesion appearance, making it a valuable resource for training robust deep learning models.

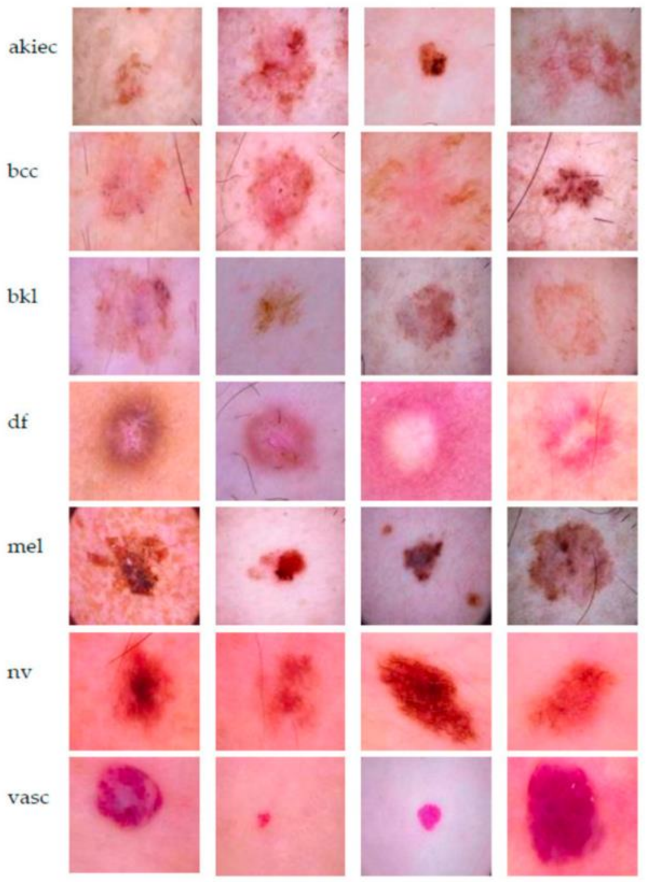

Figure 3.1: Example of each class from HAM10000 dataset

Table 3.1: Classes in HAM1000 with number of samples

Label Code	**Full Name**	**Number of Samples**
akiec	actinic	327
bcc	basal cell carcinoma	514
bkl	benign keratosis lesions	1099
df	Dermatofibroma	115
mel	Melanoma	1113
nv	Melanocytic nevi	6705
vasc	vascular lesions	142

3.1.2. Dataset overview for skin moisture classification

Initial experiments used a 3153-image augmented dataset, which overfit due to synthetic distortions, causing poor generalization. A 1373-image noisy dataset with classes oily, dry and normal as seen in Figure 3.2, followed, but mislabeled instances and class imbalance led to unstable convergence. The final dataset, with 1074 real images (294 Dry, 319 Normal, 459 Oily), was manually reviewed for label accuracy and organized for PyTorch's ImageFolder utility, offering the best data integrity despite minor noise.

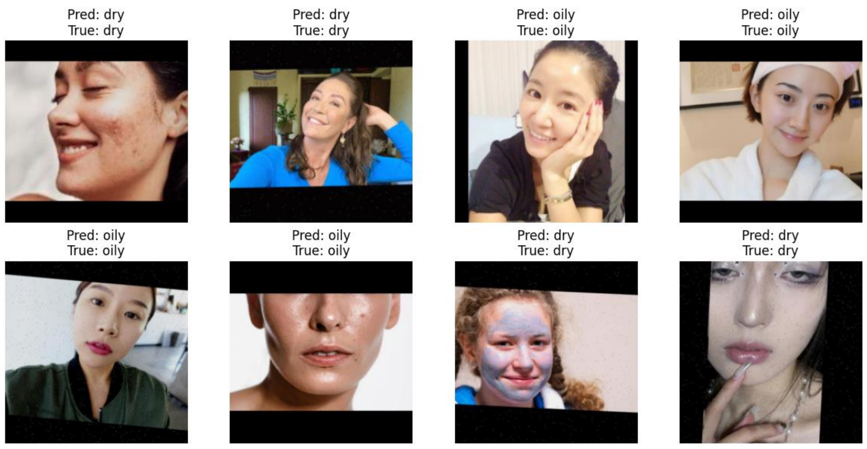

Figure 3.2: Example of images from Skin moisture classification

3.2. Model Overview

To tackle skin lesion classification using the HAM10000 dataset, we explored several convolutional neural network (CNN) architectures, including both standard and transfer learning-based models. The standard CNN serves as a baseline, designed from scratch to learn features specific to the dataset. For transfer learning, we leveraged pre-trained models like VGG16, ResNet, and DenseNet, which are well-known for their strong performance in image classification tasks. VGG16, with its deep, sequential layers, excels at capturing detailed visual features but can struggle with computational complexity [41].

ResNet introduces residual connections, allowing deeper networks without the issue of vanishing gradients, making it efficient for complex pattern recognition [42]. DenseNet, with its dense connectivity, promotes feature reuse and reduces the number of parameters, leading to robust and efficient learning [43]. These transfer learning models were fine-tuned on the HAM10000 dataset to adapt their learned features to skin lesion classification. Additionally, we also experimented with an ensemble model, combining predictions from these architectures to investigate whether their complementary strengths can improve overall performance, particularly in handling the dataset's class imbalance and nuanced differences between lesion types.

Each model was trained and evaluated on accuracy, precision, recall, and loss to assess their effectiveness in detecting and classifying skin lesions. EfficientNet and ResNet will only be used for the related task of classifying skin moisture levels, specifically distinguishing between oily and dry skin.

3.3. Deployment Platform Overview

To bring our skin lesion classification models to end users, we developed two distinct deployment platforms tailored for Android devices, each addressing different user needs. The first is a lightweight mobile app built using Kivy, a Python framework that enables rapid development and offline functionality.

This app allows users to capture a skin lesion image and run the model locally, ensuring accessibility in areas with limited internet and prioritizing user privacy by keeping data on the device [44]. The app was packaged into an APK using Buildozer, a tool that simplifies the compilation process for Android deployment [45]. The second platform is a full-featured healthcare app developed with React Native, which supports cross-platform deployment for both Android and iOS from a single codebase. This app not only performs skin disease classification but also includes features like user authentication, appointment scheduling, blog reading, e-commerce, and profile customization, creating a comprehensive health management tool [46].

The backend for this app is powered by Django, paired with MySQL for robust data storage and management, ensuring scalability and security [47]. We explored different deployment options to balance performance, accessibility, and user experience, testing both lightweight, offline solutions and feature-rich, cloud-connected platforms to determine the most effective way to deliver AI-driven diagnostics to users.

4. Data Preprocessing

Data preprocessing is one of the most integral prerequisites of model training, We shall preprocess the data, in our case images prior to feeding it to the model, the data preprocessing methods for both disease and moisture are unrelated and chosen according to model needs, let us study them briefly in the following sections

4.1. Data Preprocessing for skin disease classification

The dataset underwent a series of preprocessing steps to ensure compatibility with the deep learning model and improve generalization performance. The main steps involved are summarized below:

4.1.1. Image Resizing

All images were resized to a fixed dimension of 56×56×3 to ensure uniformity and compatibility with the convolutional neural network (CNN) input layer.

4.1.2. Data Splitting

The dataset was divided into training and testing sets using an 80/20 split ratio via the train_test_split function from scikit-learn. This ensures a balanced distribution of samples for evaluation and training.

4.1.3. Data Normalization

Pixel values of all images were rescaled to the [0, 1] range by dividing by 255. This normalization step is essential for stable and faster convergence during model training.

4.1.4. Data Augmentation

To enhance model robustness and mitigate overfitting, the training data was augmented using techniques such as random rotations, width and height shifts, shearing, and both horizontal and vertical flipping as seen in Figure 4.1. The augmentation was implemented using TensorFlow's ImageDataGenerator.

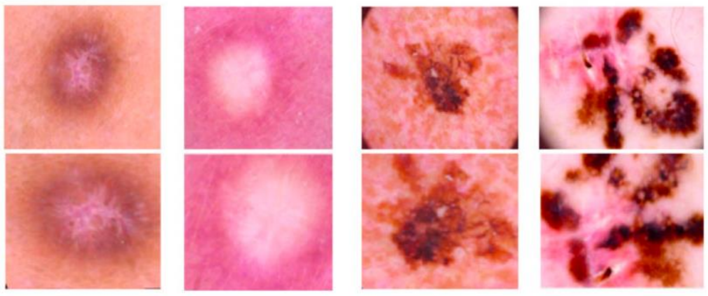

Figure 4.1: Vertical or horizontal pixel shift with a maximum of 10

4.2. Data Preprocessing for Skin Moisture Classification

The dataset underwent a tailored preprocessing pipeline to ensure compatibility with deep learning models and to promote generalization. The primary steps involved are outlined below:

4.2.1. Random Resized Cropping

To improve generalization, all training images were randomly cropped and resized to a fixed dimension of 224×224×3. This helps introduce scale variability and prevents the model from overfitting to specific spatial arrangements.

4.2.2. Data Splitting

The dataset was divided into training and validation sets using an 80/20 split ratio, resulting in 859 training and 215 validation images. A fixed random seed (42) was used with the train_test_split function from scikit-learn to ensure reproducibility.

4.2.3. Data Augmentation and Normalization (Training)

Training images were augmented using random horizontal flipping and color jittering to simulate real-world lighting variations and orientations. Additionally, all images were normalized using ImageNet statistics: mean = [0.485, 0.456, 0.406] and standard deviation = [0.229, 0.224, 0.225].

4.2.4. Conservative Preprocessing (Validation)

Validation images underwent conservative preprocessing, including resizing to 256×256 followed by center cropping to 224×224. The same ImageNet normalization was applied to maintain consistency in feature scaling.

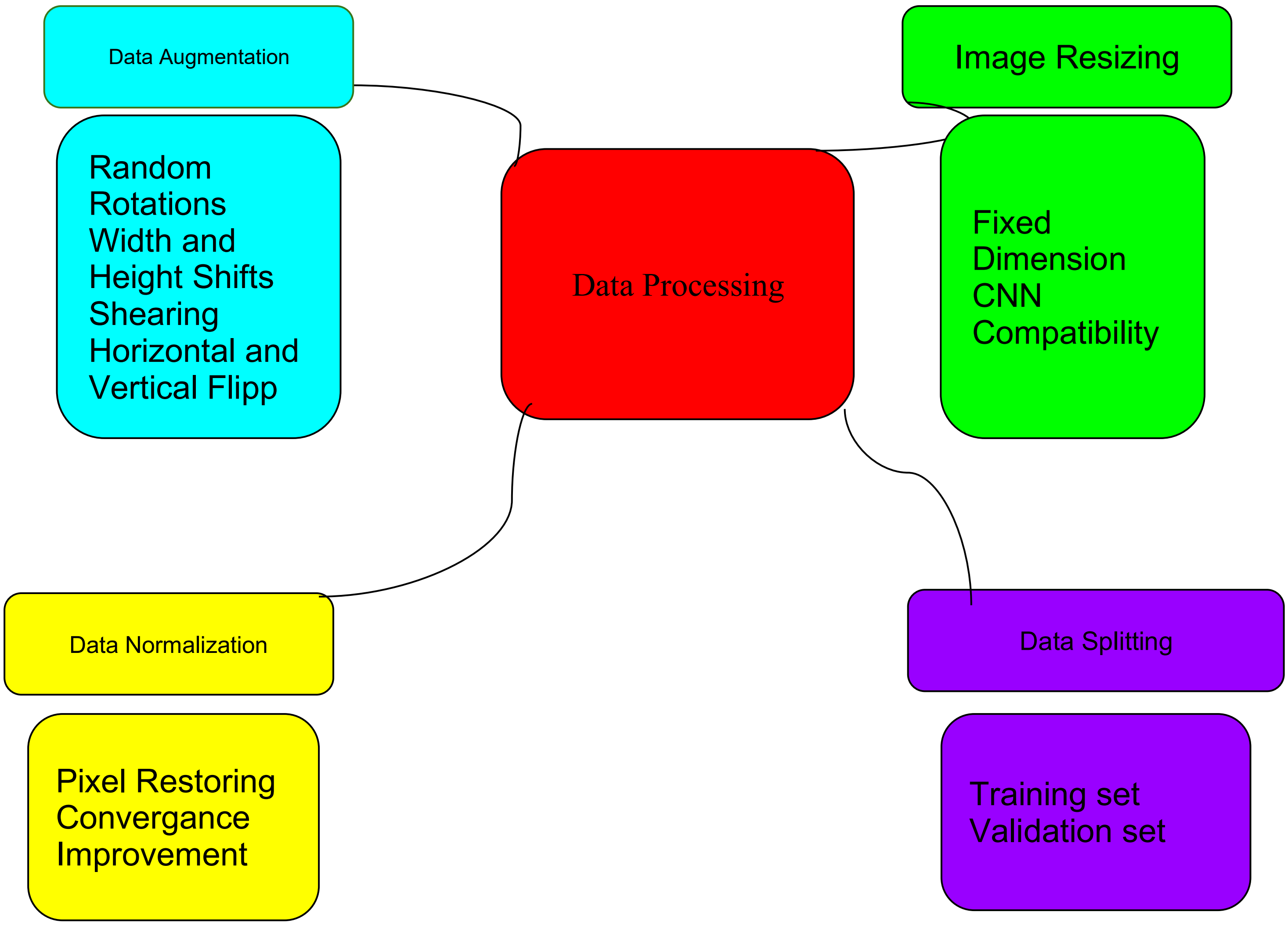

Figure 4.2: Data Processing steps

5. ARCHITECTURE

Convolutional Neural Networks (CNNs) are especially powerful when working with structured grid-like data such as images, audio, and speech. In fact, they often outperform traditional neural networks in these areas [27]. CNNs are built using three key types of layers: convolutional layers, pooling layers, and fully connected (FC) layers. These layers work together with core components, like the input data, convolutional filters, and feature maps to learn and extract meaningful patterns at different levels of detail.

The convolutional layer is the first to act on the input, applying a set of learnable filters to generate feature maps. This helps the model detect local patterns such as edges, textures, and shapes, building up a spatial understanding of the data [28].

Next come the pooling layers, also known as downsampling layers. These layers reduce the size of the feature maps, which helps decrease computational complexity and reduce the risk of overfitting. Common pooling methods include max pooling, which keeps only the maximum value in a region, and average pooling, which calculates the average [29].

Finally, the fully connected (FC) layers take over. Unlike the earlier layers, each neuron in an FC layer connects to all the activations in the previous layer. This dense structure allows the model to integrate the high-level features it has learned and produce the final output such as a classification result [30].

In this study, we focus on using CNNs for skin disease prediction with the HAM10000 dataset. We compare two different approaches to designing the model architecture: building a custom CNN from scratch, and using pre-trained image classification models available in Keras. Both methods have their strengths, and we'll explore the details of each in the following sections.

5.1. Standard CNN Model

The standard CNN model is a single deep learning network that is typically trained from scratch or fine-tuned using a pre-trained model on a large dataset like ImageNet. The goal of the CNN model is to automatically extract features from input images through layers of convolutions, pooling, and fully connected layers to classify the data.

5.1.1. Architecture of the Standard CNN Model

Now we shall examine architecture of our own custom made CNN model, we shall discuss all parts briefly.

5.1.1.1. Input Layer

The input to the model is an image with a fixed size, typically 224x224 or 256x256 pixels, and 3 color channels (RGB) but due to limited resources I have used 58x58 pixel images in my CNN model.

5.1.1.2. Convolutional Layers

The convolutional layers apply various filters to the input images, extract- ing hierarchical features like edges, textures, and shapes. These layers are followed by activation functions (e.g., ReLU) as shown in figure 5, to introduce non-linearity.

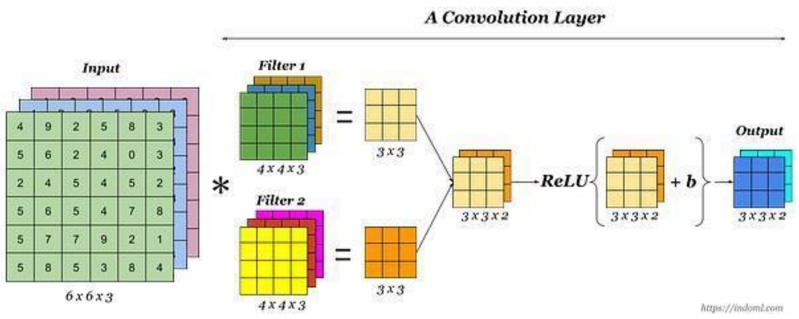

Figure 5.1: Illustration of convolutional layers extracting features

5.1.1.3. Pooling Layers

Max pooling is a technique used in convolutional neural networks to reduce the spatial dimensions of feature maps while preserving key information. It works by sliding a small window, typically 2x2, over the feature map and keeping only the maximum value from each region as seen in figure 6. This reduces computational load, prevents overfitting, and makes the model more robust to slight shifts or distortions in the input.

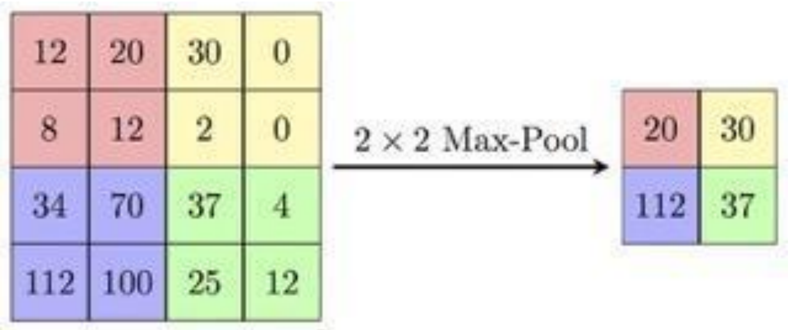

Figure 5.2: Illustration of a functioning Max Pooling layer

5.1.1.4. Fully Connected Layers

After feature extraction, the multidimensional output is flattened into a one-dimensional vector and passed to fully connected (dense) layers. These layers integrate the high-level features captured by the convolutional and pooling layers and learn complex patterns by adjusting weights through backpropagation as seen in figure 7. Fully connected layers act as the decision-making part of the network, combining all extracted information to produce the final output, such as class scores in classification tasks.

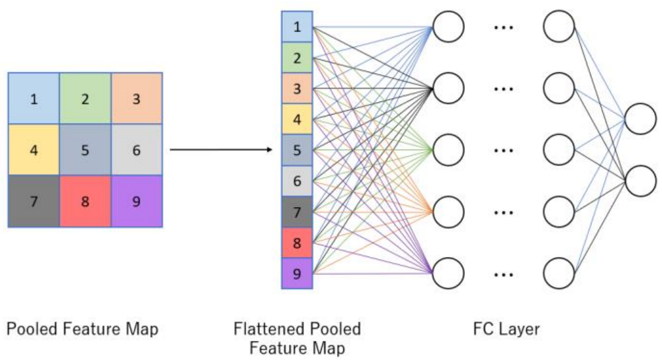

Figure 5.3: Fully Connected Layer

5.1.1.5. Output Layer

The output layer uses a softmax activation function for multi-class classification. Each class corresponds to a specific skin disease type, and the network outputs a probability distribution for each class.

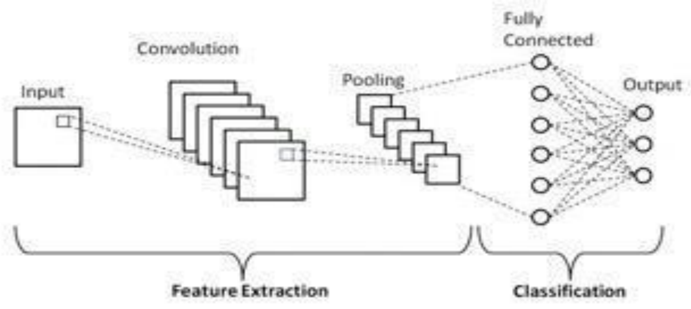

Figure 5.4: All Layers performing to give results that contribute in predicting the class using output layer

5.1.1.6. Adam Optimizer

Adam (Adaptive Moment Estimation) is a popular optimization algorithm in deep learning due to its efficiency, speed, and robustness. It combines the strengths of two other methods: Momentum , which helps accelerate gradients in the right direction by using the exponentially weighted average of past gradients.

RMSProp, which adapts the learning rate for each parameter by dividing by a moving average of recent gradient magnitudes. Adam adjusts the learning rate dynamically for each weight in the network, which helps models converge faster and more reliably, especially on noisy or sparse data like medical images. it also requires less tuning of hyperparameters (learning rate, etc.). It works well with large datasets and high-dimensional spaces and Combines the advantages of both momentum and adaptive learning rate methods, another advantage is that it is efficient in terms of computation and memory, we also have a lot of other optimizers, but for image classification adam is widely used, other optimizers have different pros and cons compared to Adam as shown in table 5.1.

5.1.1.7. Other Optimizers vs Adam

We have a few other options for our optimizer, although Adam is widely used for image classification because of its efficiency and adaptiveness, we also discuss other optimizers in table 5.1 and compare them with Adam to demonstrate the reason for our choice.

Table 5.1: Comparison of Adam with other widely used optimizers

Name	**Description**	**Pros**	**Cons**
SGD	Vanilla Stochastic Gradient Descent	Simple, widely used	Slow convergence, needs tuning
Momentum	Adds a fraction of the previous update to current one	Faster convergence than SGD	Needs momentum parameter
RMSProp	Adapts learning rate per parameter	Good for non-stationary problems	May not generalize well
Adagrad	Adapts learning rate using all past gradients	Good for sparse data	Learning rate can become too small
Adam	Combines Momentum + RMSProp	Fast, efficient, adaptive	May sometimes generalize poorly

5.1.2. Advantages of the Standard CNN Model

The standard CNN model comes with several benefits that make it a solid choice for medical image classification. One of its biggest advantages is its simplicity, it’s easy to implement and train on a specific dataset. It also performs automatic feature extraction, meaning the model can learn to identify patterns in images on its own, without the need for manual intervention. CNNs are generally known for their strong performance in image recognition tasks, and while our model may not always achieve perfect accuracy, that’s not due to any weakness in the CNN structure itself. Instead, performance limitations usually come from factors like dataset quality, training conditions, or overfitting. Another strength of the standard CNN is that it’s relatively lightweight compared to more complex models like ensembles or large pre-trained networks, making it a good option when computational resources are limited. That said, using a single CNN architecture can sometimes be limiting, it might not capture all the subtle or complex patterns in the data, which can lead to reduced accuracy when the model is tested on new, unseen images.

Table 5.2: comparison of standard CNN and pretrained transform learning model in

Feature	**Standard CNN Model**	**PreTrained Models**
Implementation Complexity	Simple to build and train	Requires integration and fine tuning of complex architectures
Feature Extraction	Learns features from scratch	Leverages robust features learnt from large scale datasets
Accuracy Potential	High(but depends on dataset and tuning)	Often higher due to deep optimization architecture
Training Time and Resources	Less computationally demanding	More computationally intensive
Risk of Overfitting	Higher if dataset is small	Lower(due to transfer learning and pre trained weights)
Customization	Fully customizable architecture	Limited unless modified extensively
Suitability for small datasets	May struggle in generalization	Performs well with small dataset using transfer learning
Use in Medical Imaging	Good for basic classification tasks	Prefered for complex high accuracy medical imaging applications

5.2. Pre-trained image classification models

Apart from this CNN model we built from scratch, we made use of and analyzed different ready made CNN models on our HAM1000 dataset to discover abilities and weaknesses of different algorithms

5.3. Architecture of Pretrained CNN Models

5.3.1. VGG16

VGG16 is a deep convolutional neural network comprising 16 layers with trainable weights. It adopts a simple and uniform architecture by exclusively using 3 × 3 convolutional filters and 2 × 2 max-pooling layers. The input layer accepts images of size 224 × 224 × 3 (RGB format). The network consists of 13 convolutional layers, all using 3 × 3 filters with a stride of 1 and appropriate padding to preserve spatial resolution. It includes five max-pooling layers with 2 × 2 filters and a stride of 2, which progressively downsample the spatial dimensions. The architecture features three fully connected layers: the first two have 4096 neurons each, and the final one serves as the output layer with 1000 neurons for ImageNet classification as seen in figure 9. ReLU activation is applied after each convolutional and fully connected layer, while the output layer uses a softmax activation function to generate class probabilities.

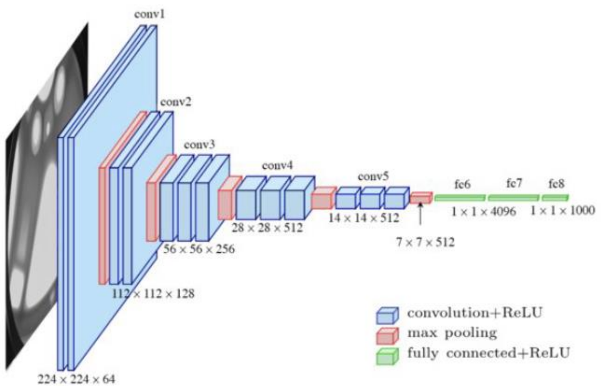

Figure 5.5: VGG16 Architecture

5.3.2. ResNet (Residual Networks)

ResNet (Residual Networks) introduces the concept of residual connections to address the vanishing gradient problem in deep neural networks. The architecture comes in several variants, including ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152. Its input layer accepts images of size 224 × 224 × 3. The network begins with an initial 7 × 7 convolutional layer with a stride of 2, followed by convolutional layers that use 3 × 3 or 1 × 1 filters within residual blocks. These residual blocks incorporate shortcut (skip) connections that bypass one or more layers, facilitating the training of deeper networks. Each block contains a sequence of convolutional operations, batch normalization, and ReLU activations. ResNet also includes both max-pooling and global average pooling layers to reduce spatial dimensions as seen in figure 10. A fully connected (dense) layer at the end outputs predictions for 1000 classes, using a softmax activation function in the output layer, with ReLU activations applied after each convolution throughout the network.

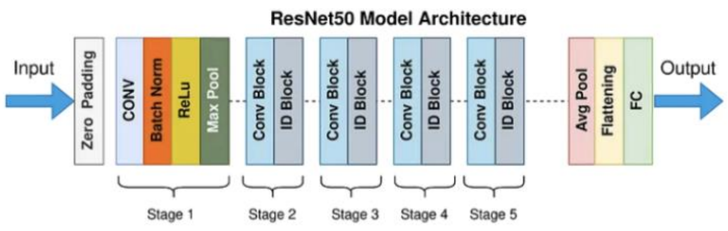

Figure 5.6: Resnet Architecture

5.3.3. DenseNet (Dense Convolutional Networks)

DenseNet, short for Densely Connected Convolutional Networks, is a deep learning architecture designed to improve information flow between layers. As shown in the figure, it starts with a 7x7 convolution followed by average pooling. The network is built using dense blocks (DB), where each layer receives input from all previous layers, encouraging feature reuse and reducing the number of parameters. Between dense blocks, transition layers are used, consisting of 1x1 convolutions and average pooling to compress and downsample feature maps. In this version, the network includes four dense blocks with 6, 12, 24, and 16 layers respectively. After the final dense block, features are passed through a classification layer to produce the prediction as shown in Figure 11. This design results in efficient training, reduced overfitting, and strong performance on image classification tasks.

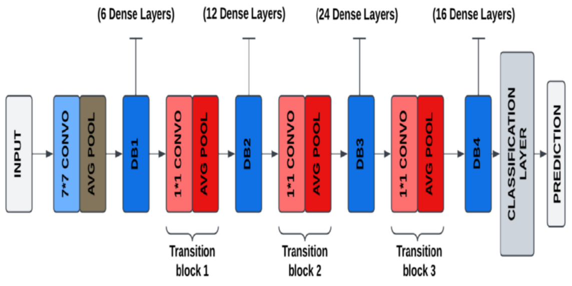

Figure 5.7: Densenet Architecture

5.3.4. EfficientNet

EfficientNet is a family of convolutional neural networks that optimizes model performance through a compound scaling method, which uniformly scales depth, width, and resolution using a fixed set of scaling coefficients. The base model, EfficientNet-B0, accepts input images of size 224 × 224 × 3 and uses a series of mobile inverted bottleneck convolution (MBConv) blocks, which incorporate depthwise separable convolutions to reduce computational cost while preserving accuracy. Each MBConv block includes a squeeze-and-excitation optimization module to recalibrate channel-wise feature responses. Batch normalization and Swish activation functions are applied after each convolution to improve training stability and non-linearity. The network begins with a standard convolutional layer followed by a series of MBConv blocks, and ends with a global average pooling layer and a fully connected dense layer for classification, using a softmax activation function to output probabilities across 1000 classes, as depicted in Figure 12. EfficientNet's balanced scaling and lightweight structure enable high accuracy with fewer parameters and reduced computational overhead compared to conventional deep CNNs.

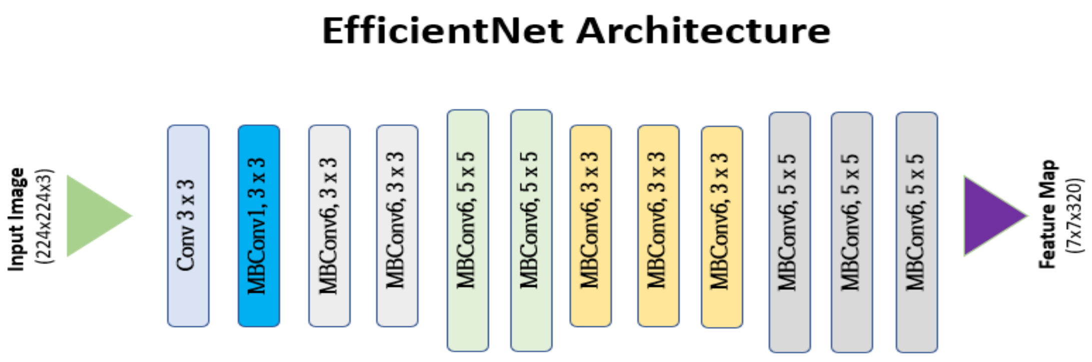

Figure 5.8: EfficientNet Architecture

Table 5.3 : Summarized structure and key features of the transform learning models used

Model/Procedure	**Input Size**	**Key Components**	**Key Features**
VGG16	224 × 224 × 3	13 convolutional layers (3 × 3 filters, stride 1), 5 max-pooling layers (2 × 2, stride 2), 3 fully connected layers (4096, 4096, 1000 neurons), ReLU activation, softmax output	Uniform architecture with small 3 × 3 filters, preserves spatial resolution via padding, designed for ImageNet classification
ResNet	224 × 224 × 3	Initial 7 × 7 convolutional layer (stride 2), residual blocks with 3 × 3 or 1 × 1 filters, skip connections, batch normalization, ReLU activation, max-pooling, global average pooling, fully connected layer, softmax output	Residual connections address vanishing gradient, supports deeper networks (e.g., ResNet-50, -101, -152)

DenseNet	224 × 224 × 3	Dense blocks with layer-to-layer connections, batch normalization, ReLU activation, 7 × 7 convolutional filters, transition layers (1 × 1 convolution, 2 × 2 average pooling), fully connected layer, softmax output	Enhances parameter efficiency, encourages feature reuse via dense connectivity
EfficientNet	224 × 224 × 3	standard convolution, MBConv blocks (depthwise separable convolutions with squeeze-and-excitation), batch normalization, Swish activation, global average pooling, fully connected layer, softmax output	compound scaling balances network depth, width, and resolution; achieves high accuracy with fewer parameters and lower computational cost

5.3.4 Training Procedure for Standard CNN Model

The training procedure for the standard CNN model adheres to the conventional setup used in deep learning. Key training parameters include the batch size, optimizer, and learning rate schedule. A batch size of 32 is used, which provides a good trade-off between memory usage and convergence speed by allowing the model to process 32 images at a time and update the weights based on the average loss computed over the batch. The Adam optimizer is employed for training, starting with an initial learning rate of 0.001. Adam adapts the learning rate for each parameter by leveraging the first and second moments of the gradients, resulting in faster convergence and improved performance. It combines the benefits of AdaGrad and RMSprop by computing adaptive learning rates for each parameter based on both the mean and the uncentered variance of the gradients. The update rule for Adam incorporates these statistics to adjust the learning step dynamically during training.

$$m_t = \beta_1 \, m_{t-1} + (1 - \beta_1) \, \nabla J(\theta_t)$$

$$m_t = \beta_1 \, m_{t-1} + (1 - \beta_1) \, \nabla J(\theta_t)$$

$$v_t = \beta_2 \, v_{t-1} + (1 - \beta_2) \, (\nabla J(\theta_t))^2$$

$$\hat{m_t} = \frac{m_t}{(1 - \beta_1{}^t)} \qquad (1)$$

$$\hat{v}_t = \frac{v_t}{(1 - \beta_1{}^t)} \qquad (2)$$

$$\theta_{t+1} = \theta_t - \alpha \left(\frac{\hat{m_t}}{\sqrt{(\hat{v}_t) + \varepsilon)}} \right) \qquad (3)$$

Equation 5.1 :the actual weight update step (3), combining momentum and adaptive learning rate

The Adam optimization algorithm uses several key components to update model parameters efficiently. The symbols (1) and (2) represent the first and second moment estimates, which are the mean and uncentered variance of the gradients, respectively. The term g_t denotes the gradient of the loss function at time step t The parameters $\beta_1$and β_2, typically set to 0.9 and 0.999, control the exponential decay rates for calculating the moving averages of the gradient and its square. The learning rate, denoted by η, determines the step size for each update, while ε, usually 10^8, is a small constant added to prevent division by zero during parameter updates. Additionally, m_t and v_t are bias-corrected to address their initial zero values, ensuring more stable and reliable convergence, especially early in training.

Loss Function

Sparse categorical cross-entropy is used as the loss function, which is suitable for multi-class classification problems with integer-encoded labels. This function measures the dissimilarity between the predicted probability distribution and the true distribution of the labels.

The formula for sparse categorical cross-entropy is given in **Equation 5.2**:

$$L = -\sum (y_i \; log(\hat{y}_i)) \qquad (1)$$

Equation 5.2: Formula for cross entropy loss function

In the context of model training, the loss function (1) quantifies the difference between the predicted probabilities and the true labels. Here, N represents the total number of classes, y_i denotes the true label (expressed as an integer corresponding to the correct class), and $\pi_i$signifies the predicted probability assigned to the ii-th class. The summation is carried out over all classes, and the formulation of the loss function ensures that greater penalties are imposed when the model assigns high probabilities to incorrect classes. Consequently, the model is incentivized to assign higher probabilities to the correct class. A lower value of the loss function indicates a closer alignment between the model's predictions and the actual labels, reflecting improved performance. The training process spans a fixed number of epochs commonly 15 during which early stopping is employed as a precaution against overfitting. Throughout training, the validation loss is continually monitored, and if it ceases to improve for a specified number of epochs, the training procedure is halted. This strategy conserves computational resources and enhances the generalizability of the model.

5.4. Training Procedures for Skin Disease Classification Using ResNet, VGG16, and DenseNet

In the task of skin disease classification, models such as ResNet, VGG16, and DenseNet are trained using a standardized procedure to ensure efficient learning and strong generalization capabilities. All images are resized to 224 × 224 pixels in RGB format to match the input size requirements of the pre-trained models. A batch size of 32 is used across all training runs, offering a balanced compromise between computational efficiency and stable gradient estimation, which is particularly important when dealing with medical image datasets prone to overfitting.

The Adam optimizer is selected for all training processes due to its ability to adaptively adjust learning rates for each parameter based on the first and second moments of the gradients. This optimization method helps the models converge more quickly and reliably compared to traditional stochastic gradient descent. The learning rate is initially set to 0.001 and may be dynamically adjusted using learning rate schedulers if the validation performance stagnates or begins to degrade.

Categorical cross-entropy is used as the loss function to address the multi-class nature of the skin disease classification problem. This function measures the difference between the predicted class probability distribution and the true labels, penalizing incorrect high-confidence predictions and thus promoting better-calibrated output probabilities. Each model is trained for a maximum of 20 epochs, with early stopping applied based on the validation loss. If the validation loss does not improve over a predefined number of epochs, training is halted to prevent overfitting and reduce unnecessary computational cost. Despite differences in architectural design, all three models are able to learn discriminative features effectively under this unified training approach. VGG16 uses a deep and sequential architecture with uniform small filters to capture local patterns. ResNet incorporates residual connections that allow for deeper networks to be trained without suffering from vanishing gradients. DenseNet, on the other hand, promotes feature reuse through densely connected layers, enhancing both parameter efficiency and model performance. This systematic training pipeline ensures that each of these models is well-suited to the skin disease classification task.

5.5. Training Procedures for Skin Moisture Classification Using ResNet and EfficientNet

The classification of skin moisture levels, which involves categorizing facial skin images into normal, dry, or oily types, follows a similarly standardized yet task-specific training methodology. In this case, models such as ResNet and EfficientNet are employed due to their proven effectiveness in feature extraction and classification accuracy. All input images are resized to 224 × 224 pixels and normalized using ImageNet statistics to ensure compatibility with pre-trained weights and to promote stable model convergence.

The ResNet model, such as ResNet-18, is loaded with pre-trained weights and fine-tuned on the skin moisture dataset. EfficientNet, which scales depth, width, and resolution in a balanced way, is also used to improve classification performance while maintaining computational efficiency. Both models are modified to include a final fully connected layer with three output neurons, corresponding to the three skin moisture classes. The training data is handled using PyTorch's Dataset and DataLoader classes, with stratified data splitting to preserve class balance across training, validation, and testing phases.

A batch size of 32 is used for training, offering a practical trade-off between GPU utilization and learning stability. The Adam optimizer is again chosen for its adaptive learning rate capabilities, initialized at 0.001. The cross-entropy loss function is employed to guide learning in this multi-class classification problem. Additionally, simple data augmentations such as resizing and normalization are applied to reduce overfitting and improve the model's ability to generalize across different skin types. Training is carried out for up to 20 epochs, with early stopping based on validation loss performance to avoid overfitting. If the validation loss does not improve for a set number of epochs, training is stopped automatically. This prevents model degradation and ensures computational resources are used efficiently. Under this training regime, both ResNet and EfficientNet demonstrate strong performance in identifying skin moisture categories, with ResNet offering solid baseline results and EfficientNet providing higher accuracy with fewer parameters due to its compound scaling strategy.

5.6. Mathematical Formulations

Although we use the same Adam optimizer and cross entropy loss for both our custom CNN model and transfer learning model, we shall review them again in the section 5.6.1 and 5.6.2 to avoid ambiguity and for the sake of revision

5.6.1. Adam Optimizer

Adam updates the weights θ at each time step t using **Equation 5.1,** the actual weight update step (3), combines momentum and adaptive learning rate where α is the learning rate, β_1 and β_2 are decay rates (commonly 0.9 and 0.999), ε is a small constant to prevent division by zero (e.g., 1e-8) and $\nabla J(\theta_t)$ is the gradient of the loss function with respect to the parameters at time t.. The symbols (1) and (2) represent the first and second moment estimates, which are the mean and uncentered variance of the gradients.

5.6.2. Categorical Cross-Entropy Loss

For a multi-class classification problem, the categorical cross-entropy loss is defined as in **Equation 5.2,** Where L is the loss, y_i is the true label (in one-hot encoded format), $\hat{y}_i$ is the predicted probability for class i and the sum runs over all classes i from 1 to C. This loss function penalizes incorrect predictions more heavily when the model is confident but wrong, encouraging accurate and well-calibrated probability outputs.

5.7. Evaluation Metrics

To assess the performance of the trained models, we utilize several evaluation metrics. These metrics help us quantify how well the models perform in classifying skin diseases. We present the evaluation metrics for both the normal CNN model and the ensemble learning model below.

5.7.1. Precision

Definition: Precision measures the proportion of true positive predictions among all positive predictions. It indicates the model's ability to avoid false positives.

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

Equation 5.3: Precision is shown as True positives with respect to True Positives and False Positives

5.7.2. Recall (Sensitivity)

Definition: Recall, also known as Sensitivity or True Positive Rate, measures the proportion of actual positives that were correctly identified. It evaluates the model's ability to detect positive samples.

$$\text{Recall} = \frac{TP}{TP+FN}$$

Equation 5.4: Mathematical representation of sensitivity or recall of training or validation

5.7.3. Accuracy

Definition: Accuracy measures the proportion of correctly classified samples out of the total samples. It provides an overall performance measure of the model.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Equation 5.5: Accuracy mathematically defined as True positives and True Negative values with respect to all True and False positives and negatives.

6. IMPLEMENTATION

In this section, we present the implementation details and performance evaluation of four deep learning models used for skin disease classification: a standard Convolutional Neural Network (CNN), VGG16, ResNet, and DenseNet. Each model was assessed based on accuracy, precision, and recall on both training and validation datasets. These metrics help us understand how well the models are identifying true disease cases (recall), how accurate the positive predictions are (precision), and their overall correctness (accuracy).

6.1. Skin Disease Classification (HAM10000)

Now it's time to examine the results after we implemented each model. In this section we will discuss the results of applying different models on both disease and moisture classification.

6.1.1. Standard CNN Model

The standard CNN model did really well, scoring an accuracy of 0.9818 on both the training and validation datasets, which shows it’s great at learning. But when you dig deeper, the recall is only 0.1049, even though precision is super high at 0.9818. This means the model is good at avoiding wrong positive predictions, but it’s missing a lot of actual disease cases, basically, it’s got too many false negatives. This could be because the dataset has way more of one class than another or because the model is being too cautious due to over-regularization. To fix this, we did try tricks like data augmentation, SMOTE, or tweaking the loss function with class weighting to boost recall and make the model more balanced.

6.1.2. VGG16

VGG16, a more complex model than the standard CNN, hit a training accuracy of 0.6706 and a validation accuracy of 0.6719. Its recall is much better at 0.6608 for training and 0.6277 for validation, meaning it catches more disease cases with fewer misses compared to the CNN. But, its precision is lower at 0.6722 (training) and 0.6728 (validation), so it’s making more false-positive predictions. The similar training and validation numbers hint at overfitting, where the model isn’t generalizing well and gets stuck. Adding dropout layers, using data augmentation, or stopping training earlier could help it perform better on new data.

6.1.3. ResNet

ResNet, an upgrade over VGG16, uses residual connections to go deeper without losing its edge due to vanishing gradients. It scored a training accuracy of 0.7114, with a precision of 0.8211 and a recall of 0.6145. On the validation set, it hit an accuracy of 0.7357, precision of 0.8380, and recall of 0.6384.

This shows ResNet’s really good at precision, so it doesn’t flag too many false positives. But it's recall is just okay, meaning it’s still missing some actual disease cases. While it’s confident when it predicts a positive, it might need some tweaking or more training data to better catch those trickier or less common cases.

6.1.4. DenseNet

DenseNet, with its unique dense connections and smart feature reuse, outshines the other models in almost every way. It hit a training accuracy of 0.9109, with a precision of 0.9228 and a recall of 0.9023. On the validation set, it still performed strongly, with an accuracy of 0.8192, precision of 0.8368, and recall of 0.7930. These numbers show DenseNet strikes the best balance between precision and recall, making it great at catching disease cases while keeping false predictions low. Its design allows for powerful feature extraction and helps prevent overfitting, making it the standout model in this comparison.

Training Results for Skin Disease Classification Model

Table 6.1 and 6.2 : the training and validation results of each model respectively

Table 6.1

Model	**Accuracy**	**Precision**	**Recall**
Standard CNN	0.9818	0.9818	0.1049
VGG16	0.6706	0.6722	0.6608
ResNet	0.7114	0.8211	0.6145
DenseNet	0.9109	0.9228	0.9023

Validation Results for Skin Disease Classification Model

Table 6.2

Model	**Accuracy**	**Precision**	**Recall**
Standard CNN	0.9818	0.9818	0.1049
VGG16	0.6719	0.6728	0.6277
ResNet	0.7357	0.838	0.6384
DenseNet	0.8192	0.8368	0.793

6.2. Skin Moisture Classification (Facial Skin Type)

Classifying facial skin types into dry, normal, and oily categories posed unique challenges, and we evaluated ResNet50 and EfficientNetB0 across multiple datasets to understand how data quality and size affect model performance. Our experiments used a curated 1074-image dataset, a noisy 1373-image dataset, and a larger 3153-image augmented dataset, with results showing that high-quality data was far more critical than sheer volume for achieving reliable skin moisture classification.

6.2.1. ResNet50

ResNet50 was tested on three datasets to gauge its ability to handle varying levels of data quality and quantity for skin moisture classification. The best performance came from the 1074-image dataset, which was manually curated to ensure accurate labels for dry (294 images), normal (319 images), and oily (459 images) classes. On this dataset, ResNet50 achieved a validation accuracy of 51% with a training accuracy of 96%, showing strong learning capacity but also some overfitting due to the dataset's small size and moderate class imbalance. The careful label verification minimized noise, allowing the model, aided by transfer learning and augmentations like random cropping and color jittering, to focus on meaningful skin moisture features like texture and sheen. This setup outperformed larger datasets, proving that clean, high-quality data is key to effective classification. On the 1373-image noisy dataset, however, ResNet50's performance dropped, reaching a validation accuracy of 50% with a training accuracy of 63%. The gap between training and validation accuracy suggests overfitting, driven by mislabeled images and class imbalance that confused the model's ability to generalize. The worst results came from the 3153-image augmented dataset, where validation accuracy fell to 39% despite a training accuracy of 55%. The heavy reliance on augmentations introduced artificial distortions, like exaggerated lighting or facial structures, which led the model to learn irrelevant patterns instead of true skin moisture characteristics. These findings highlight that ResNet50 thrives on smaller, cleaner datasets with targeted augmentations, leveraging its deep architecture and transfer learning to capture subtle features when data quality is prioritized.

6.2.2. EfficientNetB0

EfficientNetB0, known for balancing performance and computational efficiency, was also evaluated on the noisy and augmented datasets, but it struggled to match ResNet50's performance. On the 1373-image noisy dataset, EfficientNetB0 achieved a validation accuracy of 49% with a training accuracy of 85%, indicating severe overfitting. Its high capacity caused it to memorize noisy patterns, such as mislabeled dry or oily skin samples, rather than learning generalizable features. This was particularly evident in cases where image quality or lighting varied, confusing the model's focus on moisture-related traits. On the 3153-image augmented dataset, EfficientNetB0 matched ResNet50's poor validation accuracy of 39%, with a training accuracy of 55%.

The excessive augmentations introduced irrelevant visual cues, like variations in facial angles or lighting, which distracted the model from detecting true skin moisture indicators. While we didn't test EfficientNetB0 on the curated 1074-image dataset, its performance on the noisier datasets suggests it's highly sensitive to data quality issues. Compared to ResNet50, EfficientNetB0 was less robust in handling noisy or overly augmented data, likely due to its architecture being more prone to overfitting in the presence of mislabels or artificial patterns. These results underscore that for skin moisture classification, EfficientNetB0 requires cleaner data and more careful augmentation to perform effectively, and ResNet50's deeper structure seems better suited for this task when paired with high-quality data.

Results for Skin Moisture classification

Table 6.3: Results of different models when ran on different sized datasets

Num of images	**model**	**Training accuracy**	**Validation accuracy**
1373	Efficient net b0	85%	49%
1373	Resnet 50	63%	50%
1074	Resnet 50	96%	51%
3153	Efficient net b0	55%	39%

7. Analysis of Model Performance

The following sections present a detailed analysis of model performance across two distinct classification tasks. By evaluating training behavior, validation outcomes, and generalization ability, we aim to highlight the strengths and limitations of each model architecture. This analysis also explores how data quality, class imbalance, and augmentation strategies impact the overall effectiveness of the models.

7.1. Skin Disease Classification (HAM10000 Dataset)

The performance evaluation of models trained on the HAM10000 dataset highlights the distinct behaviors of standard CNN, VGG16, ResNet, and DenseNet architectures. While all models show the ability to learn meaningful features to some extent, their varied precision, recall, and generalization capabilities suggest specific strengths and limitations, complicating the integration of these models into a cohesive ensemble.

The standard CNN model demonstrated high training accuracy and low training loss, indicating effective pattern recognition within the training data. However, this strong performance did not translate to the validation set, where a clear gap emerged between training and validation losses. This discrepancy is indicative of overfitting, where the model memorizes training examples rather than learning generalizable features, making it less reliable when faced with unseen data.

VGG16 offered moderate classification performance with acceptable accuracy and recall but suffered from elevated validation loss. This behavior suggests unstable training dynamics and a tendency to overfit the data. VGG16 appears to be sensitive to noise in the dataset and lacks robustness, reducing its reliability in real-world deployment. Techniques such as dropout regularization or more aggressive data augmentation could help stabilize its training and improve generalization.

ResNet achieved a more balanced trade-off between precision and recall but showed a slight drop in recall, which implies it missed several positive instances. This outcome could be attributed to a configuration favoring precision or limitations in data diversity. Enhancing the model through deeper variants of ResNet, adjusting learning rates, or increasing dataset variety might improve its sensitivity and overall effectiveness.

DenseNet emerged as the best-performing model in terms of both accuracy and balanced metric scores. Its densely connected layers promote extensive feature reuse, leading to efficient learning and improved performance. However, even DenseNet showed a dip in recall on the validation set, indicating difficulty in identifying rare or minority cases. Fine-tuning the model further, expanding the dataset with underrepresented classes, or incorporating targeted augmentations could help address this shortfall.

Attempts to ensemble these models did not yield significant improvements in overall performance. While ensemble methods typically work well when constituent models make uncorrelated errors, these models exhibit overlapping weaknesses, particularly in handling imbalanced classes. The averaging of predictions often amplified biases toward majority classes instead of mitigating them. In addition, differences in input preprocessing and image resolutions among the models such as DenseNet and ResNet handling high-resolution images versus CNNs trained on smaller images—introduced inconsistencies in feature learning. This made it challenging to align and combine model predictions effectively without compromising individual model strengths.

7.2. Skin Moisture Classification

In the skin moisture classification task, the performance analysis focuses primarily on the effectiveness of ResNet18 and EfficientNetB0 when trained on various datasets of different sizes and labeling qualities. The results reveal that the **quality of the dataset** plays a far more critical role than its quantity in determining model generalization and accuracy.

The ResNet18 model trained on a manually curated dataset of 1074 images achieved the highest validation accuracy of 51%, despite being smaller in size compared to the other datasets. This dataset underwent manual label verification, which significantly reduced noise and mislabeling. As a result, the model was better able to learn meaningful features from the images and generalize effectively to the validation set. In contrast, models trained on larger datasets of 1373 and 3153 images, which included noisier or synthetic data, achieved lower validation accuracies of 49–50% and 39%, respectively, despite reaching high training accuracies. This gap between training and validation metrics is a strong indicator of overfitting driven by mislabeling and irrelevant features introduced through excessive data augmentation.

EfficientNetB0, when trained on the 1373-image dataset, achieved a validation accuracy of 49%, while ResNet50 reached 50% on the same set. Although these numbers are close to the 1074-image result, the high training accuracy (up to 85% in some cases) suggests that the models had memorized noisy patterns in the data, such as facial structures or irrelevant textures, rather than learning discriminative skin moisture features. Similarly, the 3153-image dataset, which relied heavily on augmented samples, led to an even lower validation accuracy of 39% with EfficientNetB0, despite a 55% training accuracy. This outcome confirms that excessive synthetic distortions can mislead the model into learning non-generalizable features.

The curated 1074-image dataset paired with ResNet50 demonstrated a validation accuracy of 51% and a training accuracy of 96%, reinforcing the importance of clean, accurately labeled data. The use of transfer learning and carefully selected augmentations allowed the model to leverage the data effectively, achieving greater stability and generalization despite the presence of some class imbalance. Although the cleaner dataset significantly improved performance, challenges such as limited sample diversity and imbalanced class distribution remain. These limitations continue to restrict the model's ability to generalize across broader populations, highlighting the need for more balanced, diverse, and accurately labeled data in future work.

7.3. Limitations

While building this system, we hit a few roadblocks that were not entirely unexpected but are worth mentioning for anyone looking to take this work further. These challenges, spanning both skin disease and skin moisture classification, highlight the complexities of working with the HAM10000 dataset and our curated dataset for skin type classification, offering insights into where improvements could be made.

7.3.1. Skin Disease Classification

While developing these methods we faced numerous hurdles and plethora of things held us back, we took notes of those so that in future works we could avoid them if there are better alternatives available, lets take a look at those in the following section.

7.3.1.1. Imbalanced Dataset

One of the biggest challenges faced while working with the HAM10000 dataset was the presence of visually similar skin lesion classes. The dataset includes various types of skin conditions, such as melanoma and seborrheic keratosis, which share overlapping visual features. This similarity makes it difficult for both automated systems and even experienced dermatologists to distinguish between them without advanced tools like dermatoscopes or histopathological analysis. From a machine learning perspective, this visual ambiguity leads to class confusion, where the model may misclassify one lesion type as another. The issue is further complicated by variations in image quality, scale, and lighting. Despite using deep convolutional networks and data augmentation techniques, subtle inter-class differences limit classification performance. This highlights the importance of high-resolution, well-annotated dermoscopic images to improve model accuracy. This challenge underscores a broader issue in medical imaging: achieving reliable differentiation between conditions that are not only visually similar but also clinically significant in terms of treatment and prognosis.

7.3.1.2. Visually Similar Classes: A Challenge for Classification

One of the most significant challenges we encountered while working with the HAM10000 dataset was the presence of visually similar skin lesion classes. Several types of skin conditions in the dataset such as melanoma (mel) and seborrheic keratosis (seborrheic keratoses and similar lesions, or "seb") share highly overlapping visual features, especially in low-resolution images or under inconsistent lighting conditions. These similarities are not just problematic for automated systems; even experienced dermatologists can find it difficult to distinguish between them without advanced tools like dermatoscopes or histopathological analysis. In the realm of computational dermatology, one of the most formidable impediments to achieving precise automated classification arises from the inherent visual congruity among distinct skin lesion types. This challenge is particularly pronounced within datasets such as HAM10000, wherein various dermatological pathologies ranging from benign nevi to malignant melanoma exhibit substantial morphological overlap. The inability to discern these intricacies with absolute certainty using conventional image processing techniques underscores the necessity for cutting-edge methodologies that push the boundaries of deep learning-based medical diagnostics.

From a machine learning standpoint, this visual ambiguity engenders inter-class misclassification, wherein a predictive model erroneously attributes a lesion to an incorrect category due to the subtle and often imperceptible distinctions in texture, pigmentation, and structure. Such class confusion is exacerbated by non-standardized image acquisition parameters, including variable illumination conditions, inconsistent focal depths, and disparities in image resolution. These compounding factors not only introduce noise into the dataset but also diminish the robustness of feature extraction algorithms, thereby impeding the accuracy of convolutional neural networks (CNNs) tasked with lesion identification. Despite extensive implementation of augmentation techniques such as rotational transformations, histogram equalization, and synthetic perturbations to enhance model generalization, the challenge remains deeply entrenched. High-resolution dermoscopic imagery annotated with expert-driven segmentation continues to be the gold standard for mitigating misclassification errors. These meticulously curated datasets enable the refinement of deep feature hierarchies within CNN architectures, fostering improved differentiation between visually analogous lesion types. This predicament epitomizes a broader conundrum inherent in medical imaging: the exigency of developing algorithmic solutions capable of delineating clinically significant pathologies with unwavering precision. The ramifications of classification inaccuracies extend beyond mere academic concerns misdiagnosed malignant lesions may precipitate delayed interventions, adversely affecting patient prognosis. As such, the convergence of AI-driven dermatological analytics with rigorous clinical validation remains a pivotal objective in the pursuit of enhanced diagnostic accuracy.

7.3.2. Skin Moisture Classification

Classifying skin moisture levels (dry, normal, oily) using our curated dataset of 1074 images presented its own set of challenges. Finding a dataset specifically for this task was a major hurdle, as most dermatological datasets, including HAM10000, focus on lesions rather than moisture characteristics. Public datasets for skin type classification are rare, and many relevant ones are either proprietary or locked behind institutional access. This scarcity forced us to manually curate a smaller 1074-image dataset (294 dry, 319 normal, 459 oily), which, while carefully verified for label accuracy, wasn't ideal due to its limited scope and lack of diversity in ethnicity, lighting, or camera quality.

To stretch this small dataset, we relied heavily on augmentation techniques like random cropping, flipping, and color jittering, but this led to issues. Overusing augmented images introduced artificial distortions, like exaggerated sheen or texture, that didn't always reflect real skin conditions, causing our model to sometimes learn irrelevant patterns instead of true moisture-related features. The dataset's size was another limitation; with only 1074 images split across three classes, there weren't enough samples to fully capture the variability of dry, normal, and oily skin, especially for underrepresented groups, which likely capped our model's performance at 51% validation accuracy. We also encountered mislabeling issues, where some images were incorrectly tagged i.e oily skin labeled as dry or normal, for example, adding noise to the training process. Fixing these errors required manual review, which was time-consuming and not entirely foolproof without extensive dermatological expertise. Finally, there was a lack of established methods for skin moisture classification. Unlike lesion classification, there are few pretrained models or standardized pipelines for this task, so we had to experiment extensively with ResNet18 and other architectures, which slowed progress and required significant tuning. These challenges highlight the need for larger, more diverse datasets with accurate labels, specialized methods for skin type classification, and careful augmentation to avoid overfitting to synthetic patterns.

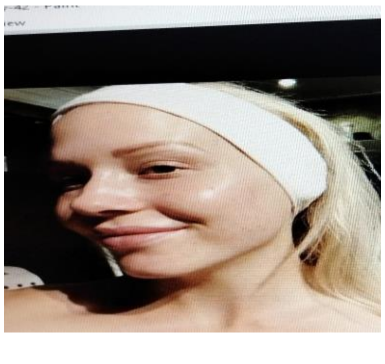

Figure 7.1: Mislabelled visually oily skin person's image labelled as dry

8. MODEL DEPLOYMENT on ANDROID APP

Getting machine learning models onto mobile devices is a big deal, as it brings AI-powered tools right to users' fingertips. For this project, we built two different Android apps, each designed for specific purposes and user needs. The first is a simple, lightweight app made with Kivy that focuses on predicting skin diseases by snapping a photo and running the model offline. The second is a more robust healthcare app that not only handles skin disease classification but also packs in features like user login, appointment scheduling, blog reading, online shopping, and profile customization. This section dives into both apps, covering the tools and technologies used to create and deploy them.

8.1. Kivy App with Minimal Functionality

The Kivy app represents the minimalistic approach to mobile AI deployment. Built with Kivy,

8.1.1. Kivy

Kivy is a user-friendly Python framework that makes it easy to build apps, and this app shows how a machine learning model can run directly on an Android device without needing an internet connection. Kivy is great for projects like this because it supports multi-touch, gestures, and a variety of UI elements, all while letting developers stick to Python. This means you don't have to learn Android-specific languages like Java or Kotlin, which speeds up the development process.

The Kivy app's main feature is super simple: users take a photo of a skin lesion or affected area with their device's camera. The app then processes the image using the built-in model right on the phone. Since it works offline, people in areas with spotty or no internet can still use it, plus it's privacy-friendly because the images stay on the device. To turn the app into an Android APK, we used Buildozer, a handy command-line tool that automates the whole process, handling all the tricky dependencies and platform-specific setup. Without Buildozer, packaging a Python app for Android would be a hassle, but it makes things smooth and beginner-friendly. The app's lightweight design keeps it fast, uses minimal resources, and is easy to navigate, delivering real-time skin disease predictions in a convenient, portable package.

8.2. Full-Stack Healthcare Application

While the Kivy app is a simple proof-of-concept for offline skin disease prediction, the full-stack healthcare app aims for a more engaging and complete user experience, built for ongoing health management. This app blends advanced machine learning with a solid backend and a slick frontend to create a comprehensive health platform. For the frontend, we used React Native, which lets us build a native-feeling app for both Android and iOS from one codebase, saving time and effort. On the backend, we went with Django, paired with MySQL as the database, to handle all the server-side logic and data storage smoothly.

8.2.1. React Native for Client Side

React Native, an open-source framework from Facebook, lets developers build mobile apps using JavaScript and React, a popular library for creating user interfaces [51]. Unlike traditional mobile development, which means writing separate code in Java or Kotlin for Android and Swift or Objective-C for iOS, React Native allows you to use one codebase for both platforms [51]. This makes app development faster and easier, blending the best of web and mobile worlds. Born in 2015 from a Facebook hackathon, React Native came from experiments with applying React's component-based approach to mobile apps [52]. When it went public, developers loved it for combining the look and feel of native apps with the flexibility of JavaScript [52].

At its heart, React Native follows React.js principles, breaking the app into reusable components that handle their own logic and state, making the code modular and easy to maintain [51]. Unlike React for the web, which tweaks the browser's DOM, React Native works directly with native UI elements, like UIView on iOS and View on Android, for smooth, native-like performance [53]. The app's structure is a tree of components, starting with a root that manages the layout and state [53]. Components come in two types: functional components, simple JavaScript functions often using hooks for state and lifecycle tasks, and class components, which are ES6 classes with more advanced features [51].

React Native offers core components like View for layouts, Text for text, Image for pictures, ScrollView for scrolling, and TextInput for user input, all designed to work consistently across platforms but flexible for customization [51]. Developers can also create custom components for specific needs, keeping projects clean and reusable [53]. Its declarative UI approach is a game-changer: instead of manually updating elements, developers describe how the UI should look, and React Native handles the rest, cutting down on bugs and simplifying complex apps [51].

Since its debut, React Native has grown into a robust ecosystem with a strong community and tons of third-party libraries to add features [52]. It's used by startups and big companies alike for everything from simple apps to complex, feature-packed products [52]. In short, React Native blends JavaScript's speed and ease with the performance users expect from native apps, making it a top pick for modern mobile development.

8.2.1.1 Other Optimizers

Another option we had was to use Flutter but why we used react native can be seen in the COMPARISON TABLE

Table 8.1: Comparison of React Native and flutter

Feature	**React Native**	**Flutter**
Developer	Meta(Facebook)	Google
First Released	2015	2017
Language	Javascript(React)	Dart
UI Rendering	Native components via bridge	Custom engine(skia)
Performance	Very good	Excellent
Community	Large and mature	Fast growing-small
Ease of Learning	Easier for web devs(JS)	Steeper(Dart)
Hot Load	Yes	No
UI Flexibility	Follows platform's native UI	Same looks across all platforms
Plugin Support	Extensively(many packages available)	Good but fewer than react native
App Size	Usually small	Larger due to rendering engine
Maturity	More established	Newer but improving fast

8.2.1.2. Advantages of React Native

React Native is a cross-platform framework that allows developers to build apps once and deploy them on both Android and iOS. It supports faster development through reusable components and hot reloading, making the iteration process more efficient. Being JavaScript-based, it integrates easily with web tools and libraries, streamlining the development workflow. With a large community, it offers extensive support, tutorials, and third-party libraries. Additionally, it supports native module integration, allowing the use of platform-specific code when needed.

8.2.2. Backend Technologies

Django is a powerful, open-source web framework built in Python that makes developing web apps fast, clean, and practical. Started in 2003 by web developers at the Lawrence Journal-World newspaper and released publicly in 2005, Django aims to simplify the tricky process of creating robust, scalable web applications [46]. It's a "batteries-included" framework, meaning it comes loaded with tools for common tasks like user login, URL routing, database management, and admin interfaces, so developers can focus on building unique features instead of starting from scratch [46]. Django uses the Model-View-Template (MVT) pattern, a twist on the classic Model-View-Controller (MVC) setup. The Model handles the data structure and talks to the database, the View takes care of the app's logic and user requests, and the Template controls how everything looks on the screen, keeping things neatly organized [47].

Its Object-Relational Mapping (ORM) system lets developers work with databases like MySQL, PostgreSQL, or SQLite using Python code instead of complex SQL, speeding things up and cutting down on mistakes [47]. Security is a big deal in Django, with built-in protections against threats like SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and clickjacking, so developers can build safe apps without being security experts [46]. Another awesome feature is the automatic admin interface, which creates a customizable dashboard for managing app data based on your models, saving tons of time for developers and clients [46].

Django is super versatile and scalable, perfect for everything from small projects to massive sites like Instagram or Pinterest [48]. Its active community contributes thousands of reusable packages, making it even more powerful [46]. In short, Django blends Python's simplicity with a rich set of tools to help developers build secure, efficient web apps with less hassle, making it a top choice for web development today.

8.2.5. Database

MySQL is one of the most popular open-source relational database management systems (RDBMS) out there, loved for its reliability, speed, and ease of use. It was first created in the mid-1990s by MySQL AB, a Swedish company, and became a go-to for storing, organizing, and pulling data using Structured Query Language (SQL), which is both powerful and approachable [49]. In 2008, Sun Microsystems bought MySQL AB, and later Oracle took over, keeping it open-source under the GNU General Public License while continuing to improve it [49].

MySQL uses a relational model, storing data in tables with rows and columns, where each table represents something specific, and tables connect through keys. This setup is great for handling complex data while keeping everything accurate [50]. It supports tons of data types and offers cool features like indexing, transactions, replication, and clustering, so it can manage anything from small apps to huge, high-traffic websites [50]. A big plus for MySQL is how flexible and compatible it is. It runs smoothly on Linux, Windows, and macOS, and plays nicely with languages like PHP, Python, Java, and Django, making it a top pick for web apps [49]. Its client-server setup lets multiple users and apps query the database at once, with strong security features like user authentication, SSL support, and role-based access control to keep data safe [50]. MySQL's query optimizer and caching make data retrieval fast, even when the system's under heavy pressure [49].

The MySQL community is huge and active, offering tools like MySQL Workbench for easy management, plus backup solutions and monitoring options to keep things running smoothly [49]. In short, MySQL is a reliable, scalable, and versatile choice for managing data, making it a key player in web development and big business systems worldwide.

8.2.4. App Integration & Full-Stack Flow

Once we finalized the model, we converted it into a TensorFlow Lite format to make it lightweight and mobile-friendly. The mobile app was developed using **React Native** for the frontend and **Django** for the backend API. For skin moisture classification we will use pytorch with Django. When the user opens the app and logs in, the frontend functionality built with React Native handles the authentication process as shown in **Figure .** After logging in, the user captures an image using their phone camera, which is then sent to the Django backend via an API. The Django server receives the image, loads a pre-trained model, and runs a prediction using the provided image. Once the prediction is complete, the result is returned to the app in JSON format and displayed to the user.

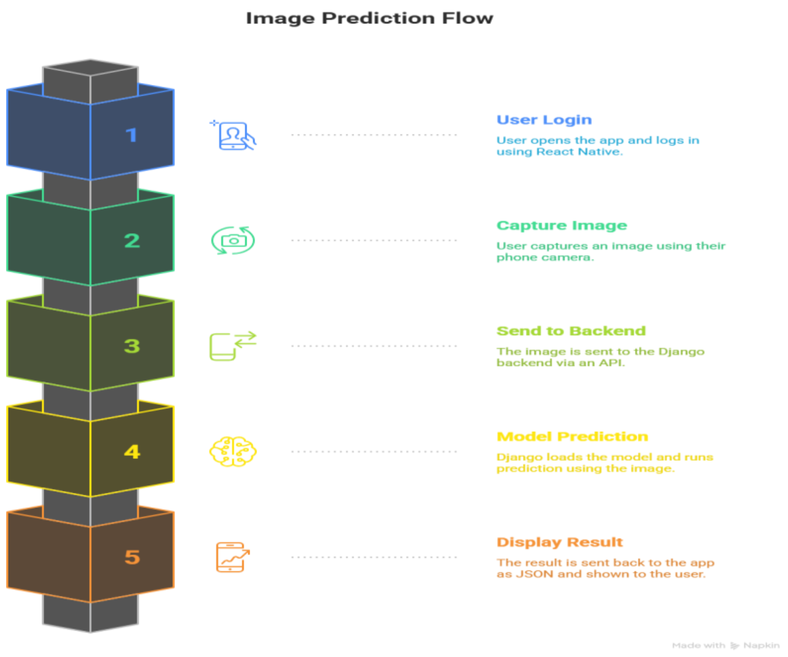
Image Prediction Flow
1
User Login
User opens the app and logs in using React Native.
2
Capture Image
User captures an image using their phone camera.
3
Send to Backend
The image is sent to the Django backend via an API.
4
Model Prediction
Django loads the model and runs prediction using the image.
5
Display Result
The result is sent back to the app as JSON and shown to the user.
Made with Napkin

Figure 8.2: Normal workflow of our application

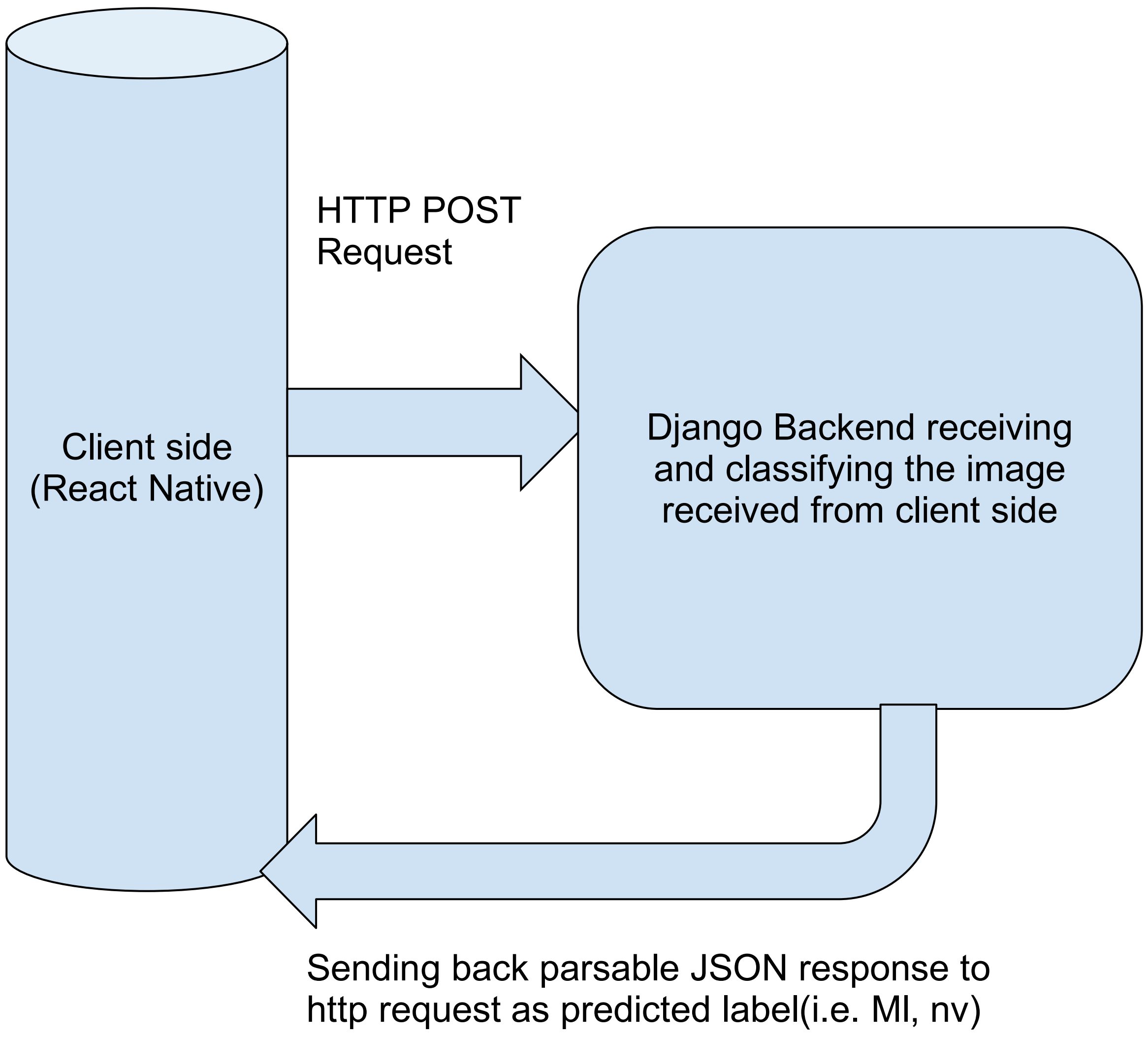
Client side
(React Native)
HTTP POST Request
Django Backend receiving and classifying the image received from client side
Sending back parsable JSON response to http request as predicted label(i.e. Ml, nv)

Figure 8.3: Workflow of our application in terms of RUST Framework

8.3. Advantages and Challenges

Deploying the model on an Android app makes it more accessible, allowing for real-time diagnosis without the need for high-performance computing resources. However, optimizing the model for mobile hardware and ensuring low latency remain challenges, especially for large neural network models.

9.4. Future Work

To improve the current study, future research could explore several meaningful directions. One approach is to apply advanced data augmentation methods like generative adversarial networks, which can generate realistic synthetic images for skin disease categories that are underrepresented in the dataset. This would help create a more balanced set of training images and improve the model's ability to recognize less common conditions. Another promising direction is to use transfer learning from larger models that have already been trained on extensive datasets. These models can offer valuable features that may boost the performance of the skin disease classification model and help it generalize better to new, unseen cases. Additionally, combining image data with clinical information such as patient history or symptoms could lead to the development of a more comprehensive model. This type of model would consider both visual and medical context, making its predictions more informed and potentially more accurate in real-world settings.

For Skin Moisture Classification Future work includes enhancing dataset diversity (ethnicity, lighting, camera quality), mitigating class imbalance, and exploring advanced techniques like test-time augmentation, ensemble models, or attention mechanisms. Plans also involve deploying the model on mobile platforms for real-time inference and investigating multi-label classification (e.g., skin tone + type). Although all these things would make a big difference but main focus on future works regarding skin moisture classification should be building a bigger and robust dataset without any falsely labelled classes, only then can a higher validation accuracy be achieved in this area of study.

9. CONCLUSION

This study demonstrates the strong potential of Convolutional Neural Networks (CNNs) in detecting and classifying skin diseases using the HAM10000 dataset as a standard reference. With a validation accuracy of around 97%, the model showcases how deep learning can assist in the automatic identification of a wide range of skin conditions, including serious diseases like melanoma. These encouraging results point to the growing role of artificial intelligence in dermatology, especially in aiding early and accurate diagnoses that can significantly impact patient outcomes.

Our findings add to a growing body of research that supports the use of machine learning and neural networks in healthcare. As seen in similar studies, CNNs are particularly effective in learning complex visual patterns from medical images, which makes them highly suitable for tasks like skin lesion classification. The success of our model aligns with previous efforts in the field and reinforces the idea that AI has the potential to reshape dermatological care, particularly in areas with limited access to specialists. Automated diagnostic tools like this can help address common issues such as long patient queues, shortages of experienced dermatologists, and the high cost of specialized medical services.

However, while the model's performance is promising, real-world clinical deployment is not without challenges. Just as highlighted in existing literature, factors like variability in image quality, differences across patient populations, and ensuring robustness under different conditions remain major concerns. Furthermore, the ethical dimensions of using AI in healthcare must not be overlooked. Over-reliance on automated systems without sufficient human oversight can introduce risks, including misdiagnosis or missed diagnoses.

In conclusion, while deep learning technologies like CNNs have shown great promise in skin disease classification, they should be seen as supportive tools rather than replacements for clinical judgment. The results of this study provide a solid foundation for future research, but any AI-based system should be implemented with care. Final medical decisions should always rest with qualified professionals. As such, the future of AI in dermatology lies in thoughtful integration, using these tools to assist and enhance, rather than replace, the expertise of healthcare providers.

In addition to skin disease classification, this study also explored the potential of deep learning models in classifying skin moisture levels. Skin moisture is an important dermatological parameter that can reflect overall skin health and hydration status, which is particularly relevant in diagnosing conditions such as eczema, dermatitis, and general skin dryness. By incorporating image-based analysis for moisture classification, the model aims to provide a more comprehensive skin assessment. Preliminary results indicate that CNNs can effectively distinguish between different moisture levels based on subtle visual cues captured in the images. This capability further broadens the scope of AI applications in dermatology, enabling more holistic skin evaluations. As with disease classification, continued research and validation are necessary to ensure accuracy and reliability across diverse populations and real-world conditions.

10. REFERENCES

[1] World Health Organization, “First WHO global meeting on skin-related neglected tropical diseases,” WHO, Mar. 2023. [Online]. Available: https://www.who.int/news-room/events/detail/2023/03/27/default-calendar/first-who-global-meeting-on-skin-ntds
[2] World Health Organization, “Skin-related neglected tropical diseases (skin NTDs),” WHO, 2023. [Online]. Available: https://www.who.int/health-topics/neglected-tropical-diseases#tab=tab_1
HYPERLINK "https://www.who.int/health-topics/neglected-tropical-diseases#tab=tab_1"[3] World Health Organization, “Universal Health Coverage and SDGs,” WHO, 2023. [Online]. Available: https://www.who.int/health-topics/universal-health-coverage#tab=tab_1
HYPERLINK "https://www.who.int/health-topics/universal-health-coverage#tab=tab_1"[4] A. Esteva et al., “Dermatologist-level classification of skin cancer with deep neural networks,” *Nature*, vol. 542, no. 7639, pp. 115–118, Feb. 2017. [Online]. Available: https://doi.org/10.1038/nature21056
[5] American Cancer Society, “Melanoma skin cancer,” 2023. [Online]. Available: https://www.cancer.org/cancer/melanoma-skin-cancer.html
[6] National Cancer Institute, “Skin cancer treatment,” 2023. [Online]. Available: https://www.cancer.gov/types/skin/hp/skin-treatment-pdq
[7] D. Popescu, M. El-Khatib, and L. Ichim, “Skin lesion classification using collective intelligence of multiple neural networks,” in *Proc. Int. Conf. e-Health and Bioengineering (EHB)*, 2022, pp. 1–6. [Online]. Available: https://doi.org/10.1109/EHB55594.2022.9991728
[8] T. Akram et al., “A multilevel features selection for skin lesion classification,” *J. Healthcare Eng.*, vol. 2020, pp. 1–12, 2020. [Online]. Available: https://doi.org/10.1155/2020/8610493
[9] P. Tschandl, C. Rosendahl, and H. Kittler, “The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions,” *Sci. Data*, vol. 5, no. 180161, 2018. [Online]. Available: https://doi.org/10.1038/sdata.2018.161
[10] M. A. Al-masni, D.-H. Kim, and T.-S. Kim, “Multiple skin lesions diagnostics via integrated deep convolutional networks,” *Appl. Sci.*, vol. 10, no. 22, pp. 1–15, 2020. [Online]. Available: https://doi.org/10.3390/app10228009
[11] A. M. Q. et al., “Skin Lesion Classification Using Hybrid Convolutional Neural Networks,” *PubMed*, 2021. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/33750716/
[12] M. N. Islam, S. S. Haque, and M. A. Hossain, “A review of deep learning-based multiple-lesion recognition from dermoscopic images,” *Comput. Methods Programs Biomed.*, vol. 228, p. 107200, Jun. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0010482523001919
[13] A. Esteva et al., “Dermatologist-level classification of skin cancer with deep neural networks,” *Nature*, vol. 542, pp. 115–118, 2017. [Online]. Available: https://www.nature.com/articles/nature21056
[14] M. M. et al., “Improved Tumor Segmentation Using MR Images Based on ResNet,” *Biocybernetics and Biomedical Engineering*, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666496823000286
[15] D. J. D. et al., “Liver Tumor Segmentation Using U-Net Architecture on LiTS Dataset,” *Medical Image Analysis*, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841522003085
[16] S. I. et al., “Detection of Mitotic Nuclei in Breast Histopathology Images Using Bio-Inspired Techniques,” *PubMed*, 2017. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/28268817/
[17] A. C. et al., “fMRI-Based Alzheimer's Disease Detection Using the SAS Method,” *PubMed*, 2023. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/37371371/
[18] Z. R. et al., “Brain Tumor Classification Using MRI Images and Deep Learning

Techniques," *PubMed Central*, 2024. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC12063847/
[19] A. N. Q. et al., "Ensemble Learning for Disease Prediction: A Review," *PubMed Central*, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10298658/
[20] A. Esteva et al., "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, pp. 115–118, 2017. [Online]. Available: https://doi.org/10.1038/nature21056
[21] P. D. Tschandl et al., "Deep learning in dermatology: A systematic review of current approaches, challenges, and future directions," *J. Eur. Acad. Dermatol. Venereol.*, vol. 34, no. 6, pp. 1146–1154, Jun. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022202X20300634
[22] H. A. Haenssle et al., "Man against machine: Diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists," *Ann. Oncol.*, vol. 29, no. 8, pp. 1836–1842, 2018. [Online]. Available: https://www.ejcancer.com/article/S0959-8049(19)30349-1/fulltext
[23] K. Melbin and Y. J. Vetha Raj, "Integration of modified ABCD features and support vector machine for skin lesion types classification," *Multimedia Tools and Applications*, vol. 80, pp. 8909–8929, 2021. [Online]. Available: https://doi.org/10.1007/s11042-020-10056-8
[24] D. Popescu, M. El-Khatib, and L. Ichim, "Skin lesion classification using collective intelligence of multiple neural networks," *Sensors*, vol. 22, no. 12, p. 4399, 2022. [Online]. Available: https://doi.org/10.3390/s22124399
[25] L. Ichim et al., "Skin lesion classification using collective intelligence of multiple neural networks," *Sensors*, vol. 20, no. 21, p. 6150, 2020. [Online]. Available: https://doi.org/10.3390/s20216150
[26] J. Esteva et al., "Deep learning-enabled skin cancer classification using Inception-v3," *Nature*, vol. 542, 2017. [Online]. Available: https://doi.org/10.1038/nature21056
[27] D. Pomponiu, "Skin cancer detection using AlexNet convolutional neural network," in *Proc. Int. Conf. on Computer Vision and Graphics*, 2016. [Online]. Available: https://doi.org/10.1007/978-3-319-46530-6_45
[28] H. Almaraz-Damian et al., "Skin lesion classification using ResNet and handcrafted features," *Biomedical Signal Processing and Control*, vol. 59, 2020. [Online]. Available: https://doi.org/10.1016/j.bspc.2020.101899
[29] M. Al-Masni et al., "Skin lesion segmentation and classification using DenseNet and fully convolutional networks," *Computers in Biology and Medicine*, vol. 121, 2020. [Online]. Available: https://doi.org/10.1016/j.compbiomed.2020.103738
[30] Y. Gong et al., "Multi-network voting system for skin lesion classification," *IEEE Access*, vol. 8, pp. 169438–169450, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3024206
[31] P. Tschandl, C. Rosendahl, and H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Sci. Data*, vol. 5, no. 1, pp. 1–9, 2018.
[32] ISIC Archive, "International Skin Imaging Collaboration Dataset," [Online]. Available: https://www.isic-archive.com
[33] J. M. Mendonça et al., "PH2 - A dermoscopic image database for research and benchmarking," in *Proc. 35th Annu. Int. Conf. IEEE EMBS*, pp. 5437–5440, 2013.
[34] DermNet NZ, "DermNet New Zealand Skin Disease Images," [Online]. Available: https://dermnetnz.org
[35] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[37] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. European Conf. on Computer Vision (ECCV)*, 2014, pp. 818–833.
[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015. [Online]. Available: https://arxiv.org/abs/1409.1556
[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90
[41] G. Huang et al., "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 2261–2269. [Online]. Available: https://doi.org/10.1109/CVPR.2017.243
[42] Kivy, "Kivy: Cross-platform Python framework for NUI," Kivy Organization, 2023. [Online]. Available: https://kivy.org/doc/stable/
[43] Buildozer, "Buildozer: A tool for creating application packages," Kivy Organization, 2023. [Online]. Available: https://buildozer.readthedocs.io/en/latest/
[44] Facebook, "React Native: A framework for building native apps," 2023. [Online]. Available: https://reactnative.dev/docs/getting-started
[45] Django Software Foundation, "Django: The web framework for perfectionists with deadlines," 2023. [Online]. Available: https://www.djangoproject.com/
[46] Django Software Foundation, "Django: The web framework for perfectionists with deadlines," [Online]. Available: https://www.djangoproject.com
[47] A. Holovaty and J. Kaplan-Moss, *The Definitive Guide to Django: Web Development Done Right*, 2nd ed. New York, NY, USA: Apress, 2009.
[48] J. Kaplan-Moss, "Django's architecture: The history and evolution of a web framework," in *Proc. PyCon 2017*, Portland, OR, USA, 2017.
[49] Oracle Corporation, "MySQL 8.0 Reference Manual," [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/
[50] M. DuBois, *MySQL*, 5th ed. Upper Saddle River, NJ, USA: Addison-Wesley, 2015.
[51] Facebook, "React Native: A framework for building native apps using React," [Online]. Available: https://reactnative.dev
[52] C. Coyier, "The history of React Native," in *Proc. React Conf 2018*, Henderson, NV, USA, 2018.
[53] D. Abramov and R. Perez, *React Native Programming*, Boston, MA, USA: O'Reilly Media, 2020.