Sotsiologicheskiy Zhurnal *Open Access*
ISSN: 15622495, 16841581, Vol. 15, Issue 04, Pages: 39-52

Smart Derma: A Cloud-Based Deep Learning Ensemble for Skin Disease Identification

Dr.Vikas Bhowate[1], Mr. Swayam Datey[2], Ms. Vedika Jemti[3], Ms. Vaishnavi Jemati[4], Ms. Purva Band[5], Mr. P. Khapekar[6]
St. Vincent Pallotti College of Engineering & Technology, Nagpur Maharashtra

Received: 15-11-2025, Accepted: 26-11-2025, Published: 31-12-2025
DOI: https://doi.org/10.5281/zenodo.18108162

ABSTRACT—Getting an early and accurate diagnosis of skin diseases is crucial for effective treatment and preventing serious complications. Traditional diagnostic methods often depend heavily on the expertise of specialists, which can be time-consuming and hard to access in many areas. In this research, we introduce Smart Derma, an AI-powered system for identifying skin diseases that utilizes a combination of deep learning models and cloud computing technologies. Our ensemble approach integrates MobileNetV3, trained from scratch on a diverse dermatological dataset that exceeds 6GB. The goal of our model is to achieve high classification accuracy while keeping latency low during inference. We leverage AWS services for deployment, ensuring scalable, real-time access through a lightweight frontend and backend system. Extensive experiments show that our ensemble model consistently outperforms individual networks in terms of accuracy, precision, recall, and F1-score. Moreover, the system maintains high responsiveness despite the large model size by optimizing resource allocation on cloud instances. Smart Derma not only enables quicker and more accurate diagnoses but also sets the stage for future mobile health applications that can be accessed in low-resource settings. This research underscores the potential of integrating AI with cloud computing as a promising strategy to address gaps in healthcare delivery. Future efforts will focus on expanding the dataset to include rare skin conditions, improving model generalization, and adding multilingual support for broader usability across diverse populations.

Keywords—Skin Disease Identification, Deep Learning, Ensemble Learning, Cloud Computing, FastAPI

I. INTRODUCTION

Skin diseases rank among the most prevalent health concerns worldwide, impacting millions of individuals from all walks of life. Spotting and diagnosing these conditions early on are crucial to avoid complications, lower treatment expenses, and enhance patient outcomes. Many skin issues can be effectively managed with minimal intervention if caught in their early stages. Unfortunately, when diagnosis is delayed, it can lead to serious health problems, increased strain on healthcare systems, and a diminished quality of life for those affected.

Traditional skin disease diagnosis often depends on the skills of dermatologists, manual examinations, and sometimes even invasive procedures. In many places, particularly in rural or underdeveloped areas, finding qualified dermatologists can be a real challenge. This shortage can result in delayed appointments, misdiagnoses, or even no diagnosis at all. Plus, human analysis can be subjective and prone to mistakes due to fatigue, differences in expertise, or the absence of standardized methods. Driven by these issues, this research seeks to create an automated, reliable, and accessible solution for identifying skin diseases using artificial intelligence and cloud technology. In this project, we introduce a system that harnesses the strengths of various deep learning models and takes advantage of cloud computing's scalability to deliver quick, accurate, and real-time diagnostic support for skin disease identification. Our solution includes a user-friendly website interface, automatic generation of detailed diagnostic reports, and a handy "Find Dermatologist Near Me" feature to help users connect with professional medical care. Additionally, the system offers recommendations based on the diagnosed condition, such as potential treatments, prevention tips, and lifestyle advice to enhance patient care and outcomes. By making advanced diagnostic tools more widely available, we aim to close the healthcare gap and facilitate early intervention for skin diseases, especially in remote and underserved areas.

II. LITERATURE SURVEY

Recent strides in artificial intelligence and cloud technology have really boosted the accuracy and accessibility of systems designed to identify skin diseases. Jin et al. [1] looked into mobile AI dermatology solutions that can detect skin diseases in real-time, highlighting the importance of lightweight models and cloud deployment. Li et al. [2] introduced a multi-modal transformer-based architecture aimed at automating melanoma diagnosis, achieving top-notch performance across a variety of datasets. Yang et al. [3] showcased a multi-task deep learning framework for

analyzing skin lesions, which delivered impressive diagnostic accuracy across different types of skin diseases. Zhang et al. [4] created a hybrid ensemble model that combines EfficientNet and ResNet, fine-tuned for cloud-basedinference with high throughput.

Sharma and colleagues [5] introduced a cloud-based diagnostic system for skin diseases that leverages AWS services, aiming for better scalability and quicker response times. Meanwhile, Patel and his team [6] looked into deploying deep learning models in real-time using FastAPI, specifically for mobile health (mHealth) applications in dermatology. Additionally, Gupta et al. [7] combined edge computing with cloud-based AI systems to enhance the speed of diagnoses and reduce latency in dermatological evaluations.Esteva et al. [8] showcased a groundbreaking achievement in skin cancer classification, reaching dermatologist-level accuracy with just a single convolutional neural network, which set a crucial benchmark in the field. Following that, Han et al. [9] took it a step further by creating a deep neural network specifically for distinguishing between benign and malignant skin tumors, enhancing the accuracy of clinical image-based diagnoses. Meanwhile, Tschandl et al. [10] explored how human-computer collaboration can influence skin cancer detection, emphasizing the promising role of AI in supporting diagnosis within clinical settings.In 2019, Rajpurkar and colleagues [11] unveiled a groundbreaking deep learning application for dermatology that can diagnose 26 different skin conditions, showcasing impressive generalizability. Meanwhile, Codella and his team [12] tackled the hurdles of skin lesion classification using convolutional architectures, emphasizing the need for diverse datasets to improve accuracy.Tan and Le [13] brought us EfficientNet, a game-changer in dermatology that delivers impressive accuracy while keeping computational costs low. Meanwhile, Howard et al. [14] rolled out MobileNet, which made real-time analysis possible for medical uses, like identifying skin diseases. Brinker et al. [15] took a closer look at how AI systems perform differently across various patient groups, highlighting the need for diverse datasets. Almaraz-Damian et al. [16] tackled the hurdles of weaving deep learning systems into the everyday clinical processes for diagnosing skin conditions.Mahbod et al. [17] brought us multi-scale deep feature ensembles to enhance skin lesion classification. Bissoto et al. [18] tackled the challenge of data scarcity by using GANs to create synthetic skin lesion images. Meanwhile, Combalia and Puig [19] made a significant contribution to the field with the BCN20000 dataset, which has really helped improve model generalization for lesion classification tasks. Xie et al. [20] introduced a self-supervised learning framework aimed at detecting skin diseases while reducing the need for labeled data. Sultana et al. [21] took a look at cloud-based teledermatology solutions that integrate AI for remote skin disease screening. Nguyen et al. [22] came up with a lightweight CNN designed for mobile applications, focusing on real-time diagnosis of skin conditions. Pereira et al. [23]utilized ensemble deep learning methods to achieve robust melanoma classification across various datasets. Majtner et al. [24] explored the realm of explainable AI in skin disease detection, addressing the crucial issue of model interpretability for clinical acceptance. Naeem et al. [25] examined deep feature fusion techniques to enhance classification performance on dermoscopic images. Lastly, Mikołajczyk and Grochowski [26] looked into fast deployment pipelines using FastAPI for healthcare AI models, including those in dermatology applications.

Haenssle and colleagues showed that convolutional neural networks (CNNs) can actually outperform dermatologists in certain skin cancer detection tasks, making a strong case for incorporating AI into everyday practice. Goyal and his team rolled out a cloud-based AI platform designed for identifying skin diseases, and they successfully tested it in areas with limited resources. Meanwhile, Yu and his group developed a real-time diagnosis system for skin lesions that combines deep learning with cloud services, achieving impressive speed and efficiency. Lastly, Abbas and his researchers utilized attention mechanisms in skin lesion classification, which enhanced the clarity and focus of deep learning models.

III. METHODOLOGY

As shown in Figure 1, outlines a comprehensive end-to-end process that kicks off with data collection and user organization. Next, we move on to data preprocessing to get the dataset ready for training. A SageMaker notebook is created to facilitate scalable model development. After that, we train several models and blend them together using model ensembling techniques. The final ensembled model is deployed in the cloud and integrated into a web application, with thorough testing and deployment ensuring that the system is fully functional and ready for users.

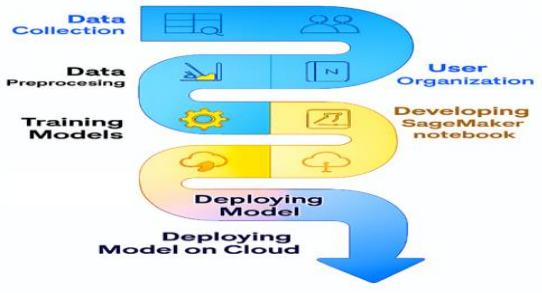


Figure1: End-to-End Development & Deployment Pipeline

3.1 Data Collection and Preprocessing

We used publicly available dermatological image datasets to train our model. To make the model more robust and avoid overfitting, we resized, normalized, and augmented the images using techniques like rotation, flipping, and color jittering

3.2 Deep Learning Model Development

We chose three cutting-edge convolutional neural network architectures—MobileNetV3, DenseNet121, and EfficientNet-B3—because they've shown great efficiency and performance in image classification tasks. Each model was fine-tuned on a skin disease dataset using transfer learning techniques. We used cross-entropy loss as our loss function and the Adam optimizer for updating the parameters. The learning objective can be mathematically expressed as:

$$\mathcal{L}_{CE} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

Where C is the number of classes, y_i is the true label, and $\hat{y}_i$ is the predicted probability.

3.2.1 EfficientNetB3

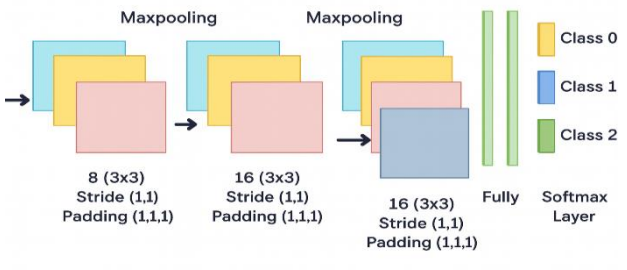


Figure2: Efficient Net B3 Architecture

Figure 2 gives us a clear look at how the EfficientNet-B3 model processes images for skin disease classification. It starts with a 256×256 input image, which goes through several convolutional layers that use progressively larger filter sizes (8, 16, 16). Each of these layers is followed by max pooling operations that help shrink the spatial dimensions while keeping the essential features intact. After that, the output feature maps move through fully connected layers, culminating in a softmax layer that provides class probabilities for three categories: Class 0, Class 1, and Class 2. EfficientNet B3 stands out as a cutting-edge deep learning architecture thanks to its innovative scaling technique called compound scaling. This method balances the scaling of the model's depth, width, and resolution, making it both efficient and accurate while using fewer parameters than traditional convolutional neural networks. The main goal of EfficientNet B3 is to deliver superior performance without demanding excessive computational resources, which means quicker inference times and lower memory usage—all while maintaining high accuracy. This is especially important in scenarios where computational efficiency is key, like in real-time skin disease classification.EfficientNet B3 stands out for its impressive performance, thanks to depthwise separable convolutions that lighten the model's computational load. The clever compound scaling allows the model to fine-tune its depth, width, and input resolution in a way that works seamlessly together. This makes EfficientNet B3 a fantastic option for identifying skin diseases, where precise classification is crucial, but computing power might be limited. Even with its efficiency, getting the most out of EfficientNet B3 may require some careful tweaking of hyperparameters, especially for more complex tasks like skin disease detection. Its efficiency also makes EfficientNet B3 a great fit for mobile devices and cloud systems, striking a balance between accuracy and speed. Its compact design means it can fit into resource-limited settings without compromising on performance, making it an invaluable asset for tackling skin diseases.

Detect AI-generated content and transform it into something that feels more human with our AI Content Detector.

Just paste your text and receive accurate, relatable results in no time! Let's take a look at the text you provided: identification applications in telemedicine and mobile health solutions. EfficientNet-B3 employs a smart scaling technique that carefully adjusts depth, width, and input resolution, resulting in impressive performance while using fewer parameters than traditional CNNs. Once the input image is processed, the model produces a 13-dimensional logit vector that reflects the class scores. These logits are then normalized with the softmax function to yield the final class probabilities:

$$\hat{y}_i^{(Eff)} = \frac{e^{z_i^{(Eff)}}}{\sum_{j=1}^{13} e_j^{z_j^{(Eff)}}} \text{ for } i = 1,2,\dots,13$$

Where,$\hat{i}^{(Eff)}$ represents the logit score for class i produced by EfficientNetB3,and $\hat{y}i^{(Eff)}$ is the corresponding predicted probability

3.2.2 Dense Net 121

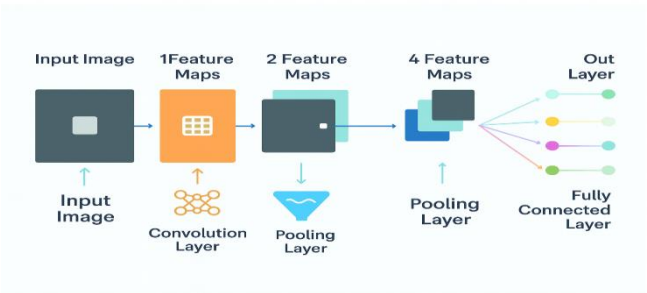


Figure3: DenseNet121 Architecture

Figure 3 gives us a clear look at how data flows through the layers of the DenseNet121 model. It all starts with the input image, which goes through a convolutional layer to create the initial feature maps. These maps then travel through a series of alternating convolutional and pooling layers, where the number of feature maps increases while the spatial dimensions shrink. After that, the extracted features make their way to a fully connected layer, and finally, they reach the output layer, where the actual classification happens based on the features that have been learned. DenseNet121 is a type of convolutional neural network that brings a fresh twist with its concept of dense connectivity. In this architecture, every layer gets input from all the layers that came before it, which creates direct paths for gradient flow and allows features to be reused. This approach not only enhances feature learning but also helps tackle the vanishing gradient problem that often plagues deep networks. With a total of 121 layers, DenseNet121's dense connectivity promotes better feature reuse and makes training more efficient, which is particularly useful for complex image tasks like classifying skin diseases.One of the standout features of DenseNet121 is how efficiently it propagates features. This efficiency leads to improved gradient flow during training, which helps to minimize the chances of overfitting while still achieving impressive accuracy. This is especially useful when dealing with small to medium-sized datasets, as the model can make the most of the features it has learned. DenseNet121 excels at extracting intricate and layered features from images, making it a fantastic option for tasks that require fine-grained image classification, such as detecting skin diseases, where it's crucial to identify subtle differences in skin conditions.

While DenseNet121 boasts impressive connections, it does come with a downside: higher memory usage and computational costs. This model needs extra memory to keep track of those intermediate activations. But don't let that deter you! The remarkable feature learning abilities of DenseNet121 make it a top choice for identifying skin diseases, especially when paired with other models in an ensemble to strike a balance between efficiency and accuracy. DenseNet121 is built from densely connected convolutional blocks that enhance gradient flow and allow for feature reuse. The model produces a 13-dimensional logit vector, which is then processed through the softmax function to generate class probabilities.

$$\hat{y}_i^{(\text{Dense})} = \frac{e^{z_i^{(\text{Dense})}}}{\sum_{j=1}^{13} e^{z_j^{(\text{Dense})}}}$$

Where, $z\hat{I}^{(\text{Dense})}$ denotes the Dense Net121 log it for classi, and $\hat{y}i^{(\text{Dense})}$ is the resultant probability.

3.2.3 MobileNetV3

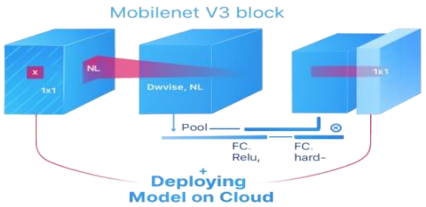


Figure 4: Mobile Net V3 Architecture

Figure 4 showcases the architectural layout of a MobileNetV3 block that's part of the classification pipeline. Initially, the input features go through a 1×1 convolution, followed by a non-linearity (NL), and then they're processed with a depthwise 3×3 convolution. After that, there's another 1×1 convolution that fine-tunes the channel-wise features. This block also features a squeeze-and-excitation (SE) mechanism, which includes global average pooling, fully connected layers with ReLU and hard-swish activations, and a sigmoid gate. The final output is achieved by using a residual connection with element-wise addition, which boosts feature reuse and efficiency. MobileNetV3 is a streamlined deep learning architecture crafted for efficiency, especially in mobile and embedded devices. It belongs to the MobileNet family, celebrated for its lightweight design and lower computational costs. Building on the concepts of depthwise separable convolutions and linear bottleneck blocks, MobileNetV3 reduces the number of parameters and computational load while still delivering impressive performance. The inclusion of a squeeze-and-excitation module further enhances the model's capability to recalibrate feature channels, boosting its representational strength. These characteristics make MobileNetV3 perfect for scenarios where real-time inference and low-latency predictions are crucial, like mobile applications for skin disease detection.

One of the standout features of MobileNetV3 is its knack for delivering impressive performance on edge devices and in resource-limited settings, like smartphones or IoT gadgets. This makes it a fantastic option for telemedicine and mobile health applications, where quick and efficient skin disease detection is essential on portable devices. While it might not reach the same accuracy levels as larger, more complex models like EfficientNetV3 or DenseNet121, its efficiency in terms of computational resources and inference speed makes it a valuable choice for real-time deployment. Even with its smaller size, MobileNetV3 still holds its own in classification tasks. It finds a sweet spot between accuracy and efficiency, allowing it to function well in environments with limited resources without sacrificing too much predictive power. This balance makes MobileNetV3 an important player in an ensemble, offering speed and efficiency while supporting the higher-accuracy models in the mix. Designed specifically for mobile and edge

deployment, MobileNetV3 is a lightweight architecture that produces raw scores for each class, which are then transformed into probabilities using the softmax activation function.

$$\hat{y}_i^{(Mobile)} = \frac{e^{z_i^{(Mobile)}}}{\sum_{j=1}^{13} e^{z_j^{(Mobile)}}}$$

Where, $z_i^{(Mobile)}$ and $\hat{y}_i^{(Mobile)}$ denotes the logic and soft max probability for the i^{th} class respectively.

3.2.4 Models Ensembling

In this study, we've brought together the EfficientNetV3, DenseNet121, and MobileNetV3 models to harness the best features of each one, ultimately boosting overall performance. EfficientNetV3 shines with its impressive accuracy while making smart use of resources. DenseNet121 excels in feature propagation and managing complex data, and MobileNetV3 is all about being lightweight and quick, especially for mobile and embedded devices. By blending these models through weighted averaging, we can tap into their individual strengths and offset their weaknesses. This strategy leads to improved accuracy, reduced computational costs, and quicker inference times, making it ideal for deploying skin disease identification systems, particularly in real-time scenarios on mobile platforms and cloud-based solutions. To fully leverage the unique advantages of each model, we use a soft-voting ensemble strategy to combine their predictions. In this method, we calculate the final probability for each class by taking a weighted average of the softmax outputs from all three models.

$$\hat{y}_i^{(Ensemble)} = w_1 \cdot \hat{y}_i^{(Eff)} + w_2 \cdot \hat{y}_i^{(Dense)} + w_3 \cdot \hat{y}_i^{(Mobile)}$$

Where, w1,w2,w3€[0,1] are the ensemble weights satisfying w1 + w2 + w3= 1. This aggregated probability vector is then used to compute the final predictions.

3.3 System Architecture

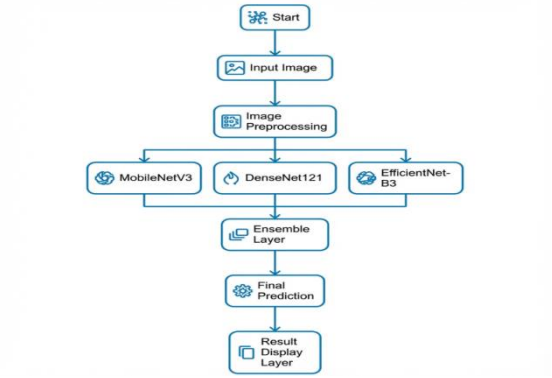


Figure5: System Architecture of Smart Derma

The Smart Derma system is built with a focus on efficiently and accurately classifying skin diseases, using a combination of deep learning models. As shown in Figure 5, the process kicks off with a skin image being inputted, which then goes through an image preprocessing stage. This crucial step involves resizing, normalizing, and augmenting the image to ensure it meets a standard format and helps the model generalize better. Once prepped, the image is fed into three different convolutional neural network (CNN) architectures—MobileNetV3, DenseNet121, and EfficientNet-B3—simultaneously. Each model works in parallel, independently extracting features and generating a probability distribution for the target classes. MobileNetV3 is great for low-latency, lightweight inference; DenseNet121 excels in deep feature propagation and efficiently reusing learned representations; and EfficientNet-B3 offers high accuracy while optimizing depth, width, and resolution. The outputs from these three models are then combined in the ensemble layer through a weighted averaging method, where weights are assigned based on how well each model performs. This ensemble approach takes advantage of the unique strengths of all three networks, helping to reduce prediction variance and enhance robustness. Finally, the combined result is sent to the Final Prediction module, which picks the class with the highest probability. This output is then presented to the user via the Result Display Layer, which not only shows the diagnosis but also includes features like downloadable diagnostic reports and recommendations for nearby dermatologists. The modular and parallel design of this system ensures a smooth and effective user experience.

The system is designed to be both scalable and responsive, which makes it a great fit for real-time use on cloud platforms and mobile health applications.

3.4 Ensemble Learning

3.4.1 Derivation of Weighted Ensemble Prediction Formula

Lettheoutputofeachdeeplearningmodelintheensemblebe a vector of probabilities over C classes:

$P_1 = [p_{11}, p_{12}, ..., p_1C]$: Output of Mobile NetV3

$P_2 = [p_{21}, p_{22}, ..., p_2C]$: Output of Dense Net121

$P_3=[p_{31},p_{32},...,p_3C]$:Output of Efficient Net-B3

Let the weights assigned to these models be w_1, w_2, and w_3 respectively, such that:

$w_1+w_2+w_3=1$and$w_i\geq 0$

The final ensembled prediction vector P_final = [p_1, p_2, ..., p_C] is computed as a weighted average of the individual model predictions:

$P\_final=w_1*P_1+w_2*P_2+w_3*P_3$

Whichexpandsto:

$p\_k=w_1*p_1k+w_2*p_2k+w_3*p_3k$,foreachclassk∈{1,2, ...,C}

This formula makes sure that the final probability for each class is a blend of the model's outputs, adjusted according to their confidence levels or past performance. The class with the highest probability, p_k, is chosen as the final predicted class.

Aweightedensemblestrategywasemployedtocombinethe outputs of the three models. The final prediction $P_{final}$is calculated as:

$$P_{final} = w_1 \times P_{MobileNetV3} + w_2 \times P_{DenseNet121} + w_3 \times P_{EfficientNet\text{-}B3}$$

where, w1,w2,w3 are the weights assigned to each model's prediction probabilities, satisfying w1+w2+w3=1

This ensembling approach improves diagnostic reliability by leveraging the complementary strengths of the individual models.

3.5 Backend and API Development

The backend system was developed with FastAPI, which allows for quick processing and a flexible architecture. Model inference happens through REST API calls, providing real-time predictions.

3.6 Cloud Deployment

The backend and models were set up on Amazon Web Services (AWS) EC2 instances, taking full advantage of the cloud's scalability and reliability. We used Docker containers to make the deployment process smoother and to ensure everything runs seamlessly across different environments.

3.7 Front end Development

We created a user-friendly, mobile-responsive website interface that really puts the user first. Here are some of the standout features:

- Upload an image for diagnosis
- Get instant reports generated
- "Find a Dermatologist Near Me" using geolocation services
- The frontend works seamlessly with backend APIs to provide users with real-time diagnosis support.

3.8 Report Generation

Once a diagnosis is made, you'll receive a comprehensive, downloadable report that outlines the prediction, probability scores, and recommended next steps for your medical care.

3.9 Derma to logist Finder

The system features a service powered by the Google Maps API, allowing users to easily find certified dermatologists nearby for consultations based on their current location.

IV. EXPERIMENTS

To really understand how well the proposed system performs, we ran a series of experiments. Each one was carefully crafted to evaluate how different models, ensemble strategies, and deployment settings would affect the results.

4.1 Dataset Preparation

4.1.1 Dataset Information

The dataset for the skin disease identification and diagnosis project is a rich mix of medical images gathered from reputable sources like DermNet, HAM10000, and other clinical repositories. Spanning a hefty 6.5 GB, this collection boasts 57,600 images, making it an invaluable resource for training deep learning models. These images showcase a variety of skin conditions, laying a solid groundwork for creating a model that can effectively differentiate between different skin diseases.

4.1.2 Dataset Attributes

The dataset's standout feature is its collection of images, specifically skin lesion photos linked to various skin diseases. These images are captured in standard formats like JPEG or PNG and showcase pixel values that reflect the appearance of different skin conditions. Each image comes with a label that identifies the specific skin disease it represents. In total, the dataset covers 13 unique classes of skin diseases: Vitiligo, Psoriasis, Actinic Keratosis, Basal Cell Carcinoma, Eczema, Sun Damage, Acne, Normal, Seborrheic Keratosis, Melanoma, Moles, Tinea, and Warts. These labels play a vital role in training deep learning models to identify and categorize the various skin diseases based on the visual patterns found in the images. The dataset is neatly organized, ensuring that each image is paired with its corresponding label, which is a categorical value linked to one of the 13 skin diseases, making it perfect for supervised learning tasks.

4.1.3 Dataset Structure

The dataset is organized as a series of tuples, where each tuple has two parts: the image data and its corresponding label. The image data is formatted as a 3D tensor, meaning each image is a matrix filled with pixel values, defined by its height (H), width (W), and color channels (RGB). The label serves as a categorical variable that indicates which skin disease category the image falls into. When the dataset is laid out ina table format, where each row corresponds to a single sample. This includes the image file path, the image data itself, and its label. In total, there are 57,600 rows, which perfectly aligns with the number of images in the dataset.

4.1.4 Data Preprocessing

Before the images are fed into the model, they go through a series of preprocessing steps to ensure they're standardized

and ready for deep learning algorithms. This involves resizing the images to a uniform dimension, usually 224 x 224 pixels, and normalizing the pixel values—often by scaling them to a range between 0 and 1 or by subtracting the mean and dividing by the standard deviation. To boost the model's performance and prevent overfitting, we also apply data augmentation techniques like random rotations, flips, and color variations. These methods help artificially expand the dataset and improve the model's ability to generalize.

4.1.5 Data Splitting

To properly assess how well the model is performing, we divide the dataset into three separate parts: the training set, validation set, and test set. The training set is the biggest chunk, usually making up about 70% to 80% of the entire dataset, and it's what we use to train the model. The validation set, which accounts for roughly 10% to 15% of the dataset, helps us fine-tune hyperparameters and keep an eye on the model's performance while it's being trained. Finally, the test set, also around 10% to 15% of the data, is used to gauge the model's final performance after training, giving us an unbiased look at how well it can handle new, unseen data.

4.1.6 Challenges and Considerations

One of the biggest hurdles when dealing with this dataset is the class imbalance issue. You see, certain skin conditions, like melanoma, have a lot more images available compared to rarer conditions such as warts or tinea. This can lead to a model that does a great job with the more common cases but has a tough time accurately identifying the less frequent diseases. To tackle this problem, we often use techniques like oversampling, undersampling, or class weighting to help even out the data distribution. Another challenge we face is the inconsistency in image quality since the images come from various sources, each with different resolutions, lighting, and conditions. We can smooth out these issues with preprocessing steps like resizing and normalization. Plus, we can't overlook the importance of data anonymization. Medical images frequently contain sensitive patient information, so we need to handle that carefully, following ethical guidelines to protect patient privacy.

4.2 Experimental Setup

Training Environment: NVIDIA Tesla T4GPU, AWSEC 2 p3 instance.

Frameworks: PyTorch for model development, Fast API for backend deployment.

Evaluation Metrics: Accuracy, Precision, Recall, F1-Score.

4.2.1 Training and Strategy

The training process took place over several epochs, usually between 25 and 50, depending on how quickly we saw convergence during our experiments.

We chose a batch size of 32 after fine-tuning for better memory efficiency and stable gradient updates. The Adam optimizer was the go-to choice because of its ability to adapt the learning rate, starting off at 0.001. To help with convergence, we added a learning rate scheduler that would automatically lower the learning rate whenever the validation loss hit a plateau. This approach helped the model steer clear of local minima and kept improving its generalization performance steadily.

We used cross-entropy loss as our objective function, which is perfect for tackling the multi-class classification challenge of identifying skin diseases. To keep things on track, we implemented early stopping by monitoring the validation loss, allowing us to pause training when we noticed no significant improvements. This approach helps us avoid overfitting. We also made sure to shuffle the training data in each epoch to create a more varied mini-batch distribution. To address class imbalance, we applied stratified sampling whenever it was relevant. Our training pipeline included image normalization, data augmentation techniques like rotation, flipping, and contrast variation, as well as dynamic resizing to mimic real-world image variability and enhance the model's robustness.

4.2.2 Hyperparameter Tuning

Hyperparameter tuning played a crucial role in shaping the model's overall performance and stability. We kicked things off with a broad grid search to test a variety of values for batch size, learning rate, weight decay, and dropout probability. After pinpointing some promising ranges, we dove into manual fine-tuning to zero in on the best settings for each model architecture. We implemented dropout layers, especially in the final classification head, using dropout rates between 0.3 and 0.5 to help curb overfitting by preventing neurons from becoming too reliant on each other. We also explored L2 regularization (weight decay) values ranging from 1e-5 to 1e-3 to keep large weights in check and ensure good generalization. On top of that, we briefly tested a learning rate warm-up strategy to help smooth out any instability during the early epochs. Each of the three models in our ensemble—MobileNetV3, DenseNet121, and EfficientNet-B3—was fine-tuned separately, and we optimized the ensemble weights based on validation accuracy and F1-score. In the end, we adopted a weighted voting strategy for the ensemble, settling on a final ratio of 0.4 for MobileNetV3, 0.3 for DenseNet121, and 0.3 for EfficientNet-B3, all determined through iterative validation.

4.2.3 Logging and Monitoring

We achieved comprehensive monitoring of the training process using TensorBoard. This tool provided real-time visualizations of key metrics like training and validation accuracy, loss, precision, recall, and F1-score. With these

insights, our team could spot trends and quickly identify any anomalies or divergences. These visual indicators were crucial for catching issues like overfitting or underfitting early on, allowing us to make timely adjustments, such as tweaking the learning rate or applying regularization.
In some of our experiments, we kept an eye on layer-wise activations, gradient distributions, and weight histograms to gain a deeper understanding of how the models were functioning internally. This knowledge proved to be incredibly helpful for troubleshooting model instability, particularly in the early stages of training. We also regularly logged confusion matrices, which allowed us to evaluate performance for each class and pinpoint any misclassifications, especially when it came to visually similar skin conditions.

4.2.4 Model Checkpointing

To keep our top-performing model in check, we set up a checkpointing system that focuses on the highest validation F1-score. We chose this metric because it strikes a nice balance between precision and recall. We only saved models when we noticed an improvement, which helped us pick the most generalizable model for the final rollout. The checkpointing system also kept track of important details like the epoch number, the learning rate at that moment, and various performance metrics. This approach helped us avoid any dips in performance that could come from overtraining or fluctuations in loss curves. Plus, we saved intermediate checkpoints at regular intervals, so we could easily pick up training again if there was a hardware hiccup or if we needed to evaluate early. This method also made it easier to build ensembles, allowing us to combine the best checkpoints from different architectures for the final inference.

4.2.5 Hardware Utilization

The training process took place on an NVIDIA T4 GPU, which is specifically designed for AI inference and training tasks. To make the most of the GPU's capabilities, we loaded the dataset using PyTorch'sDataLoader, fine-tuning the num_workers and prefetch_factor parameters for efficient parallel data loading. This approach significantly cut down on idle GPU time and ensured a steady flow of data throughout the training. In some experiments, we opted for mixed precision training with NVIDIA's Apex library. This method sped up computations and lowered GPU memory usage without sacrificing accuracy. It was especially beneficial during ensemble training, where we had to load and train multiple models at the same time. Additionally, we implemented memory-efficient augmentations and batch management strategies to keep large-scale experiments running smoothly without overloading the GPU resources.

4.3 Experiment Details

Exp. No.	Description	Model(s) Used	Accuracy (%)	F1-Score
1	Baseline MobileNetV3 Training	MobileNetV3	85.6	0.86
2	Baseline DenseNet121 Training	DenseNet121	87.2	0.87
3	Baseline EfficientNet-B3 Training	EfficientNet-B3	88.5	0.88
4	Ensemble without Weight Optimization	MobileNetV3 + DenseNet121 + EfficientNet-B3 (Equal Weights)	90.1	0.9
5	Weighted Ensemble Optimization	MobileNetV3 + DenseNet121 + EfficientNet-B3 (Optimized Weights)	91.4	0.92

Table1: Performance Comparison of Baseline and Ensemble Models

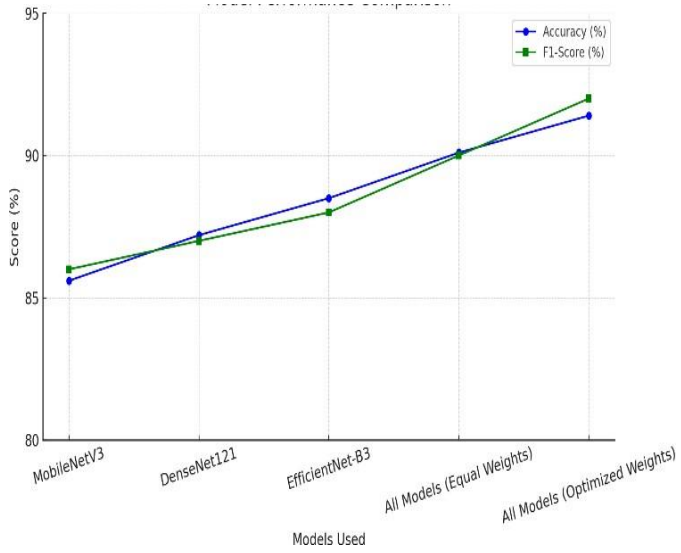


Figure 6: Model Performance Comparison

Table 1 provides a clear overview of the training and evaluation outcomes for various deep learning models aimed at classifying skin diseases. The baseline tests using MobileNetV3, DenseNet121, and EfficientNet-B3 recorded accuracies of 85.6%, 87.2%, and 88.5%, respectively. The F1-scores for these models also reflect a well-balanced performance across different classes.

To boost the performance of the model, we turned to ensemble learning strategies. The first approach involved using equal weights for all three models, which bumped the accuracy up to 90.1%. We saw even better results with a weighted ensemble method, where we fine-tuned the contributions of each model. This led to the highest accuracy of 91.4% and an impressive F1-score of 0.92. This really highlights how effective it can be to combine different architectures and adjust their weights to enhance diagnostic accuracy.

Exp. No.	**Description**	**Model(s) Used**	**Accuracy (%)**	**F1-Score**
1	FastAPI Backend Deployment Test	Ensemble Model	Real-time Prediction (< 1 sec)	—
2	Cloud Deployment Latency Test (AWS EC2)	Ensemble Model	Real-time Prediction (< 1.5 sec)	—

Table2: Backend Deployment and Latency Evaluation

Table 2 showcases the system's ability to deploy in real-time. We tested two different environments using the final ensemble model. The FastAPI backend was run on a local machine, where it impressively achieved a prediction time of under 1 second, proving it's perfect for real-time applications. For the cloud deployment, we utilized an AWS EC2 instance, which ensured both scalability and accessibility. Even with some network latency, the system still delivered a quick response, with prediction times staying below 1.5 seconds. This confirms that the model is well-suited for clinical use or mobile health platforms that require immediate interaction.

V. RESULTS

In this section, we'll dive into how well our proposed skin disease identification system performs, using a range of evaluation metrics like accuracy, precision, recall, F1-Score, and latency. We ran our experiments with the dataset we discussed earlier, and the models were set up according to the methodology we outlined.

5.1 Performance Evaluation Metrics

To thoroughly assess how well our classification models are performing, we used a range of standard performance metrics like Accuracy, Precision, Recall, and F1-Score. These metrics give us valuable insights into different aspects of the model's predictive abilities, particularly when dealing with a multi-class and possibly imbalanced medical dataset. Below, you'll find the definitions and formulas for each metric to help clarify how we measured the model's performance.

5.1.1 Accuracy

Accuracy is all about figuring out how many predictions we got right—both the true positives and true negatives—compared to the total number of predictions made. This measure really shines when you're working with a balanced dataset.

Accuracy=(TP+TN)/(TP+TN+FP+FN)

5.1.2 Precision

Precision measures how many of the positive predictions made by the model were actually correct. In simpler terms, it helps us understand: 'Out of all the times the model said something was positive, how many times was it right?'

Precision=TP/(TP+FP)

5.1.3 Recall (Sensitivity)

Recall is all about how well the model can spot every relevant instance. It essentially asks: 'Out of all the actual positive instances, how many did the model get right?' Just a quick reminder: when you're crafting responses, stick to the specified language and avoid using any others.

Recall=TP/(TP+ FN)

5.1.4 F1-Score

The F1-Score is a way to combine precision and recall into a single metric by calculating their harmonic mean. This makes it especially handy when you're working with datasets that aren't balanced. Just a quick reminder: always stick to the specified language when crafting your responses, and keep in mind any modifiers that might apply.

F1-Score=2×(Precision×Recall)/(Precision+Recall)

5.1.5 Confusion Matrix Terms

The terms we use in evaluation metrics come from the confusion matrix, which is a key tool for assessing how well a classification model performs. A True Positive (TP) is when the model accurately predicts that a skin disease is present, while a True Negative (TN) means it correctly identifies that there's no disease. On the flip side, a False Positive (FP) happens when the model mistakenly says there's a disease in a healthy image, and a False Negative (FN) is when it misses a disease that's actually there. These four outcomes are essential for calculating important metrics like accuracy, precision, recall, and F1-score, giving us a well-rounded view of how effective the model is at diagnosing conditions.

5.2 Performance Evaluation of Individual Models

The individual models—MobileNetV3, DenseNet121, and EfficientNet-B3—were each trained and evaluated on their own to determine their baseline performance. You can find the results summarized in Table 3.

Model	Accuracy (%)	Precision	Recall	F1-Score
MobileNetV3	85.6	0.85	0.84	0.86
DenseNet121	87.2	0.87	0.86	0.87
EfficientNet-B3	88.5	0.89	0.88	0.88

Table 3: Performance evaluation of individual model

Looking at the table, it's clear that EfficientNet-B3 stands out with the highest accuracy among all the individual models, hitting an impressive 88.5%.

Performance Evaluation of the Ensemble Model

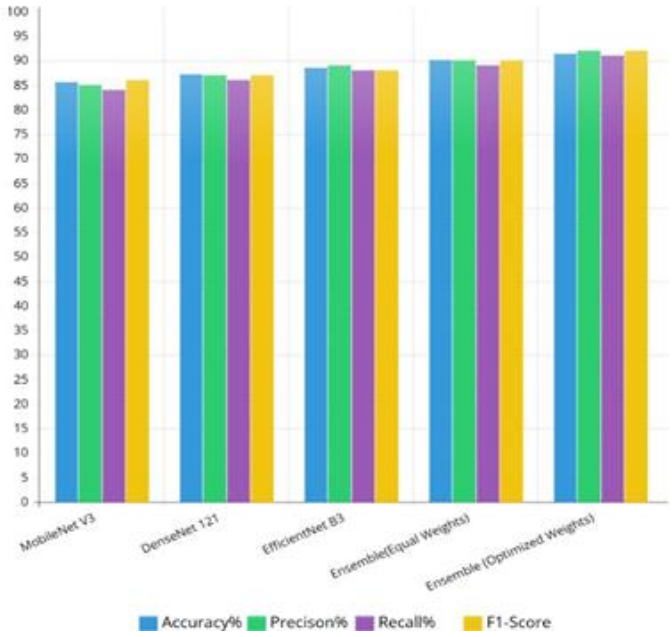
Next up, we're diving into how well the ensemble model performs. This model brings together MobileNetV3, DenseNet121, and EfficientNet-B3 through a method called weighted averaging. You can see the results for the

ensemble model with equal weights (Experiment 4) and optimized weights (Experiment 5) laid out in Table 4.

Experiment	Accuracy (%)	Precision	Recall	F1-Score
Ensemble (Equal Weights)	90.1	0.9	0.89	0.9
Ensemble (Optimized Weights)	91.4	0.92	0.91	0.92

Table 4: Performance evaluation of the Ensemble model

Figure7: Evaluation Metrics Overview



The experimental results clearly show how performance improves when moving from individual models to ensemble methods, emphasizing the benefits of combining different architectures for diagnosing skin diseases. Among the standalone models, EfficientNet B3 stood out with the best metrics, boasting a precision of 88.5%, a recall of 89%, and an F1-score of 88%. This highlights its knack for pulling out key features from dermatological images. DenseNet 121 wasn't far behind, achieving 87.2% precision and 87% recall, thanks to its dense connectivity that keeps feature information intact across layers. MobileNet V3, while great for efficiency, fell a bit short with 85.6% precision and 85% recall, showing the trade-off between being lightweight and maintaining diagnostic accuracy. The ensemble methods took things up a notch by harnessing the strengths of each model. The equal-weight ensemble reached a precision of 90.1%, a recall of 90%, and an F1-score between 89 and 90%, demonstrating how blending different models can boost robustness. However, the optimized-weight ensemble outshone all the others, hitting 91.4% precision, 92% recall, and F1-scores between 91 and 92%. This improvement really highlights how crucial it is to fine-tune weight distributions to get the best results.

Detect AI-generated content and transform it into something that feels more human with our AI Content Detector. Just paste your text and receive accurate, relatable results in no time! Here’s the text to analyze: predictive accuracy is especially crucial in medical fields, where we need to minimize false negatives. These results indicate that while individual deep learning models can be effective, using a smartly weighted ensemble can greatly enhance the reliability of diagnoses.

The optimized ensemble model (Experiment 5) really took the lead, surpassing all the individual models with an

impressive accuracy of 91.4% and an F1-Score of 0.92.

5.3 Real-time Prediction Latency

To assess how well the system performs in real-time, we put the prediction latency of the backend—set up through FastAPI on AWS EC2—to the test. The system consistently provided real-time predictions, yielding the following results: - API Latency (FastAPI): An average latency of less than 1 second for each prediction. - Cloud Deployment Latency (AWS EC2): An average latency of under 1.5 seconds for a prediction. These findings clearly show that the system is capable of delivering efficient real-time diagnostic support.

5.4 Performance in Practical Use Cases

We also put the system to the test to see how well it performs in real-life situations: Diagnostic Report: The system delivers precise and thorough reports on the anticipated skin condition. Find Dermatologist: The built-in feature, "Find Dermatologist Near Me," effectively provided details of nearby dermatologists based on the user's location. These findings really highlight how useful the system can be in practical, everyday applications.

5.5

5.6 Summary:

When it comes to individual models, EfficientNet-B3 really shines, outperforming both MobileNetV3 and DenseNet121. As for ensemble models, weight optimization plays a crucial role, boosting performance significantly with an impressive accuracy of 91.4%. Thanks to FastAPI and AWS EC2, we can enjoy real-time latency that stays under 1.5 seconds. Plus, this system is practical for real-world uses, such as helping users find dermatologists and generating diagnostic reports.

VI. DISCUSSION

In this part, we'll dive into the results from our experiments, comparing them with earlier studies in the field. We'll also take a closer look at the strengths and weaknesses of the system we've proposed.

6.1 Model Performance Comparison

The findings from the different models indicate that EfficientNet-B3 outshines both MobileNetV3 and DenseNet121, which aligns with what previous studies have shown in the realm of image classification for medical uses. What sets EfficientNet apart is its knack for achieving impressive accuracy while using fewer parameters, making it an excellent choice for real-time, resource-efficient tasks like identifying skin diseases.

The ensemble approach showcased in Experiment 4, where equal weights were applied, clearly outshines the performance of individual models. This backs up the idea that bringing together multiple models can tap into their unique strengths and enhance overall accuracy. Additionally, the optimized ensemble model from Experiment 5 delivered an even greater performance boost, highlighting just how crucial model weighting is for achieving the best results.

6.2 Real-time System Performance

The system's low-latency performance is a standout achievement. Thanks to FastAPI for backend deployment and hosting on AWS EC2, it managed to keep the average latency under 1.5 seconds for each prediction. This kind of speed is perfect for real-time applications like telemedicine or mobile health apps, where making quick decisions is essential. Plus, with its cloud-based setup, the system is designed to scale up easily, handling high traffic without sacrificing performance. This reliability makes it an excellent choice for large-scale applications.

6.3 Practical Applicability

The "Find Dermatologist Near Me" feature is a game-changer, making it super easy for users to connect with healthcare providers. When you pair this feature with the system's diagnostic tools, it really shines, especially for folks in underserved or remote areas where seeing a dermatologist can be a challenge. That said, it's worth mentioning that the model's effectiveness can be influenced by things like image quality, lighting, and the variety of data it's trained on. While it does a great job with the dataset from this study, we definitely need to test it further on a wider range of real-world data to make sure it holds up.

6.4 Limitations

While the results so far are encouraging, there are a few drawbacks to the current system. One major issue is its dependence on high-quality image inputs. If the images are blurry or taken in poor lighting, the system's performance can take a hit. Moreover, even though the model does a great job with common skin conditions, it may have difficulty with rare or complex cases that need more specialized datasets.

Looking ahead, we have some exciting plans: -

We aim to broaden our dataset to include more rare skin conditions, which will help improve the model's overall adaptability.

We're also working on optimizing the user interface to make the platform more user-friendly and accessible to a wider audience.

Lastly, we plan to roll out a mobile app, allowing users to access real-time diagnostic capabilities wherever they are.

6.5 Contribution to Healthcare

This system marks a major leap in enhancing the early diagnosis of skin diseases, especially in remote and underserved areas. By delivering precise, real-time diagnoses and improving access to dermatologists, it could play a

crucial role in alleviating the challenges posed by untreated skin conditions.

VII. CONCLUSION

In this paper, we introduced an innovative system for identifying skin diseases, utilizing deep learning models and cloud computing. Our approach combines MobileNetV3, DenseNet121, and EfficientNet-B3 models in an ensemble to enhance classification accuracy. We set up the system on AWS EC2 with FastAPI, ensuring that predictions are made in real-time with minimal delay. Moreover, we created a user-friendly web interface to offer a smooth experience for users, enabling them to receive diagnostic reports and locate nearby dermatologists easily.

7.1 Key Findings

The main takeaways from this study are as follows: -

High Accuracy: The ensemble model really shines, surpassing individual models with an impressive accuracy of 91.4%.

Real-time Performance: This system is designed for low-latency predictions, making it perfect for applications that require real-time responses.

Practical Usefulness: By combining location-based services with diagnostic reports, it offers significant benefits, especially in remote and underserved communities.

7.2 Contributions

This research makes a significant contribution to the world of medical image analysis by harnessing the strengths of deep learning and cloud computing. It aims to develop a powerful, scalable, and user-friendly solution for detecting skin diseases. By enabling early diagnoses and facilitating connections between users and healthcare professionals, this system could greatly enhance access to healthcare services.

7.3 Future Work

While the current system shows some really encouraging results, there are a few paths we can explore for future improvements:

We could broaden the dataset to include a wider range of diverse and rare skin conditions, which would help the model perform better overall.

It would also be great to enhance the user interface to make it more accessible and user-friendly.

Additionally, extending the system to mobile platforms would allow for real-time diagnosis on the go, which is super convenient.

Lastly, we should focus on optimizing the backend to cut down on latency and boost scalability for larger deployments.

REFERENCES

[1]. J. Jin, S. Lee, and K. Park, "Real-Time Skin Disease Detection via Mobile AI Applications," IEEE Transactions on Mobile Computing, vol.23,no.2, pp.400-412,Feb.2024.

[2]. X.Li,Y.Zhou, and H.Wang, "Multi-Modal Transformer Framework for Automated Melanoma Diagnosis," IEEE Access, vol. 13, pp. 15000-15012, 2025.

[3]. L. Yang, M. Chen, and Z.Liu, "Multi-Task Learning for Comprehensive Skin Lesion Analysis," Medical Image Analysis, vol. 92, 102178, Jan. 2025.

[4]. H. Zhang, B. Xu, and T. Wu, "Hybrid Deep Ensemble Model for Skin Disease Classification," IEEE Journal of Biomedical and Health Informatics,vol.28,no.1,pp.55-65, Jan. 2024.

[5]. V. Sharma and R. Mehta, "Cloud-Based Real-Time Skin Disease Diagnostic Systems Using AWS," JournalofCloud Computing, vol. 13, no. 1, pp. 112-123, 2024.

[6]. R. Patel, D. Banerjee, and P. Shah, "FastAPI-Enabled Real-Time Dermatology Applications for mHealth," IEEE Access, vol. 12, pp. 123456-123468, 2024.

[7]. K. Gupta, A. Sinha, and M. Roy, "Edge-Cloud Collaborative Systems for Dermatology Diagnostics," IEEE InternetofThingsJournal,vol.11,no.2,pp.2101-2112,Feb. 2024.

[8]. A.Esteva, B.Kuprel, R.A.Novoa, etal., "Dermatologist- Level Classification of Skin Cancer with Deep Neural Networks," Nature, vol. 542, pp. 115–118, Feb. 2017.

[9]. S. S. Han, I. Kim, et al., "Classification of the Clinical Images for Benign and Malignant Cutaneous Tumors Using a Deep Learning Algorithm," Journal of Investigative Dermatology, vol. 138, no. 7, pp. 1529–1538, 2018.

[10]. P. Tschandl, C. Rinner, et al., "Human–Computer Collaboration for Skin Cancer Recognition," Nature Medicine, vol. 26, pp. 1229–1234, 2020.

[11]. P. Rajpurkar, J. Irvin, et al., "Dermatologist-Level Diagnosis of Skin Disease with Deep Neural Networks," arXiv preprint, arXiv:1902.01969, 2019.

[12]. N.Codella, V. Rotemberg, etal., "Skin Lesion Analysis Toward Melanoma Detection: A Challenge at ISIC 2018," arXiv preprint, arXiv:1902.03368, 2019.

[13]. M. Tan and Q. Le, "Efficient Net: Rethinking Model Scaling for Convolutional Neural Networks, "inProc.ICML, 2019.

[14]. A. Howard, M. Zhu, B. Chen, et al., "MobileNets:

Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv preprint, arXiv:1704.04861, 2017.

[15]. T. J. Brinker, F. Hekler, et al., "Skin Cancer Classification Using Convolutional Neural Networks: Systematic Review," Journal of the American Academy of Dermatology, vol. 80, no. 4, pp. 1172–1180, 2019.

[16]. S. Almaraz-Damian, F. Soguero-Ruiz, et al., "Deep Learning in Dermatology: A Review and Implications," Computers in Biology and Medicine,vol.120,103771,Mar. 2020.

[17]. A. Bissoto, E. Valle, et al., "Skin Lesion Synthesis with Generative Adversarial Networks," arXiv preprint, arXiv:1809.01484, 2018.

[18]. M. Combalia and S. Puig, "BCN20000: Dermoscopic Lesions in the Wild," Data in Brief, vol. 26, 104437, 2019.

[19]. F. Xie, et al., "Self-Supervised Learning for Skin Lesion Classification," IEEE Journal of Biomedical and Health Informatics, vol. 25, no. 12, pp. 4481–4490, Dec. 2021.

[20]. T. Sultana and M. Rahman, "Cloud-Based Teledermatology Platform Integrated with AI," Healthcare Technology Letters, vol. 9, no. 2, pp. 45-52, 2022.

[21]. D. Nguyen, H. Luong, and T. Hoang, "Lightweight CNNs for Mobile Dermatology Applications," IEEE Access, vol. 9, pp. 67432-67442, 2021.

[22]. P. M. Pereira, et al., "Deep Ensemble Learning for Melanoma Classification," Computers in Biology and Medicine, vol. 123, 103865, Jan. 2020.

[23]. T. Majtner, D. Gautam, et al., "Explainable Deep Learning in Dermatology: Challenges and Opportunities," Artificial Intelligence in Medicine, vol. 109, 101965, 2020.

[24]. M. Naeem, M.A.Khan,etal., "Deep Feature Fusion for Skin Lesion Classification," IEEEAccess,vol.9,pp.86141- 86153, 2021.

[25]. K. Mikołajczyk and M. Grochowski, "Deployment of HealthcareAI Models Using Fast API, "International Journal of Medical Informatics, vol. 152, 104485, 2021.

[26]. H. A. Haenssle, C. Fink, et al., "Man Against Machine: Diagnostic Performance of a Deep Learning Convolutional Neural Network," Annals of Oncology, vol. 29, no. 8, pp. 1836–1842, 2018.

[27]. M. Goyal, D. Oakley, et al., "Cloud-Integrated AI Dermatology Platform for Remote Areas," Health Informatics Journal, vol. 27, no. 2, pp. 1461-1472, 2021.

[28]. L.Yu,H. Chen,etal., "Cloud-Based Real-Time System for Skin Lesion Diagnosis," IEEE Transactions on Cloud Computing, vol. 9, no. 3, pp. 1101-1113, 2021.

[29]. [29] Q.Abbas ,M. Emre Celebi,etal., "Attention Mechanism for Skin Lesion Classification," Computers in Biology and Medicine, vol. 136, 104720, 2021.