

Table of Contents

Chapter - 1	3
1. Introduction	3
1.1 The Need for Automated Detection Systems	5
1.2 Components of Psoriasis Detection Systems	5
1.3 Recent Advances and Future Directions	6
1.4 Challenges and Considerations	7
1.5 Reference for introduction	8
Chapter – 2.....	9
2. System Overview	9
2.1 Intended purpose	9
2.2 System Architecture	9
2.3 Data Flow.....	10
2.4 Front-End Interface	11
2.5 Backend Infrastructure.....	11
2.6 System goal.....	12
2.7 Intended users	12
2.8 Warnings and Precautions	13
2.9 How to use the DermDetect	13
2.10 System requirements.....	18
Chapter – 3.....	20
3. System Design	20
Chapter – 4.....	21
4. Back-end design.....	21
4.1 Phase 1: Training the dataset with 8 popular CNN model.....	22
4.2 Phase – 2, Ensemble Learning	40
4.3 Phase-3, Optimization Technique.....	50
Chapter – 5.....	53
5. Front-end design	53
5.1 Django: -	53
5.2 Working of Django: -	54
5.3 Advantages of Using Django: -.....	55
5.4 Data-Flow Diagram for Psoriasis Detection System: -	57

Skin Disease Detection System

5.5	Steps in the DFD are explained hereby:.....	57
Chapter – 6	59
6.	Final Testing.....	59
6.1.	Sample Images	59
6.2.	Final result.....	60
Chapter – 7	61
7.1	Future Scope	61
7.2	Conclusion.....	61
7.3	References:	61

Chapter - 1

1. Introduction

Skin disease detection systems represent a cutting-edge intersection of healthcare and technology, leveraging advancements in machine learning, image processing, and artificial intelligence to enhance dermatological diagnostics [1]. These systems are designed to analyze high-quality images of skin lesions, extracting relevant features such as texture, color, and shape, and using sophisticated algorithms to classify various skin conditions [2]. The primary goal is to provide accurate, rapid, and non-invasive diagnosis, which is crucial for timely treatment and management [3]. By automating the detection process, these systems reduce the reliance on subjective clinical evaluations and biopsies, offering a consistent and objective approach to diagnosing skin diseases [4].



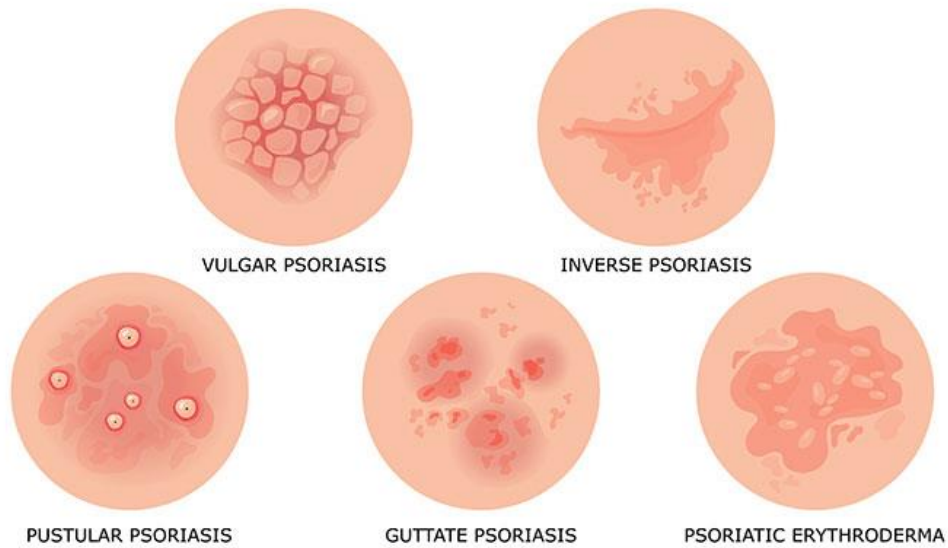
Fig1.1: Skin disease detection system

Psoriasis is a chronic, inflammatory skin condition characterized by red, itchy, and scaly patches. It is an autoimmune disorder that affects around 2-3% of the global population. The worldwide incidence and prevalence of psoriasis have been inadequately understood, prompting the need for comprehensive research. To address this gap, a systematic review of published population-based studies on psoriasis incidence and prevalence was conducted by Parisi et al in the year of 2013 [5]. Three electronic databases were searched from their inception dates to July 2011. This review critically appraised a total of 385 papers, with 53 studies specifically reporting on the prevalence and incidence of psoriasis in the general population. Findings revealed that the prevalence of psoriasis in children ranged from 0% in Taiwan to 2.1% in Italy, while in adults, it varied from 0.91% in the United States to 8.5% in Norway.

For children, the incidence estimate available from the United States was 40.8 per 100,000 person-years. In adults, incidence rates varied widely, from 78.9 per 100,000 person-years in the United States.

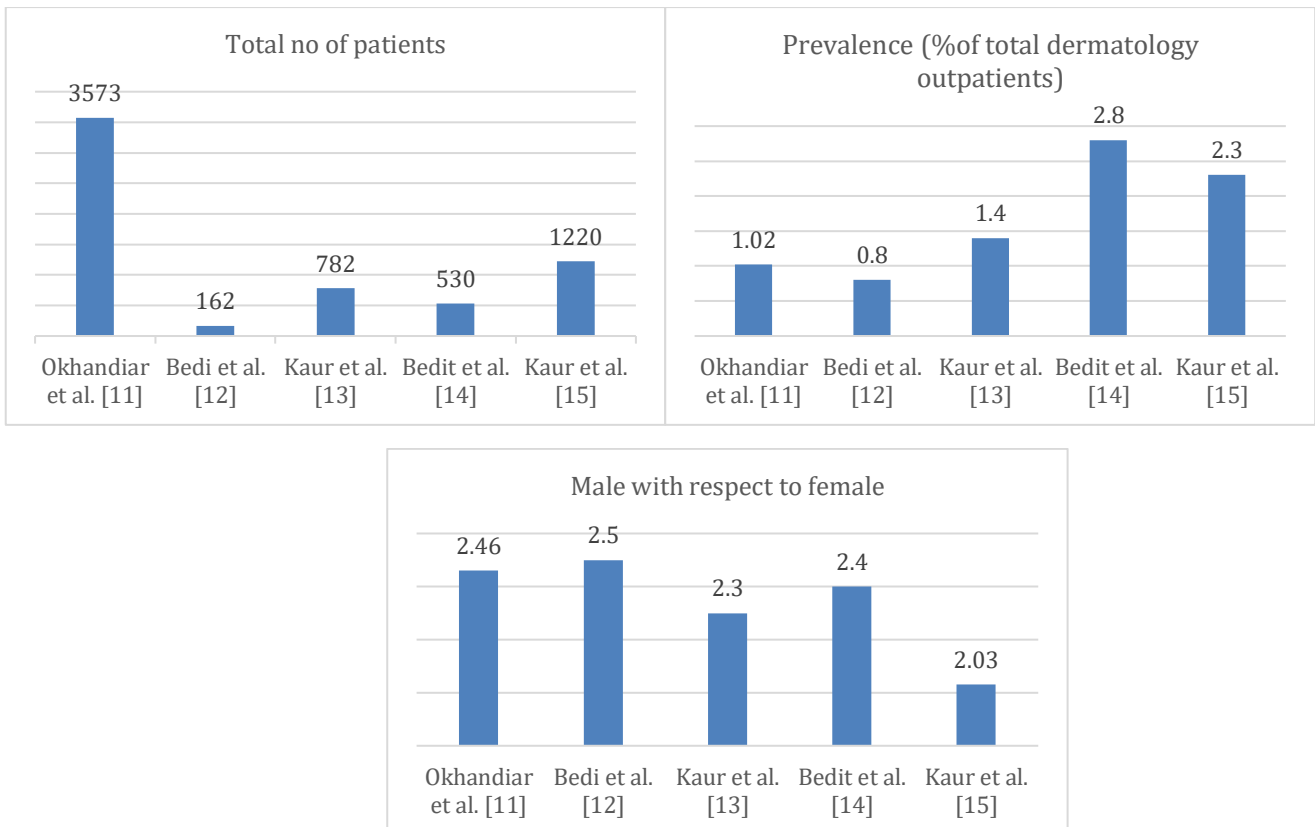
The data indicated significant variation in the occurrence of psoriasis based on age and geographic region, with higher prevalence observed in countries further from the equator. Additionally, prevalence estimates were higher in studies confined to adults compared to those including all age groups. Research on the prevalence and incidence of psoriasis has enhanced the understanding of the disease's burden. However, there remain substantial gaps in knowledge regarding the epidemiology of psoriasis and trends in incidence over time, necessitating further research.

TYPES OF PSORIASIS



Okhandiar et al. [11], Bedi et al. [12],[14] and Kaur et al. [13],[15] are working on an Indian patent for psoriasis detection.

	Okhandiar et al. [11]	Bedi et al. [12]	Kaur et al. [13]	Bedit et al. [14]	Kaur et al. [15]
Total no of patients	3573	162	782	530	1220
Prevalence (%of total dermatology outpatients)	1.020	0.800	1.400	2.800	2.300
Male: female	2.46:1	2.5:1	2.3:1	2.4:1	2.03:1



Accurate and early detection of psoriasis is crucial for effective management and treatment, as it can significantly improve the quality of life for patients [6]. Traditional diagnostic methods involve physical examination by dermatologists and sometimes require biopsy, but these methods can be time-consuming, subjective, and invasive [7].

1.1 The Need for Automated Detection Systems

The advent of advanced technologies in image processing, machine learning, and artificial intelligence has opened new avenues for developing automated systems for psoriasis detection. These systems offer several advantages:

Speed:

- Automated systems can provide quicker diagnosis compared to traditional methods.

Accuracy:

- Machine learning models, especially deep learning, have shown high accuracy in diagnosing various skin conditions.

Non-invasiveness:

- Image-based analysis avoids the need for invasive procedures like biopsies.

Accessibility:

- Such systems can be deployed in remote or underserved areas, increasing access to healthcare.

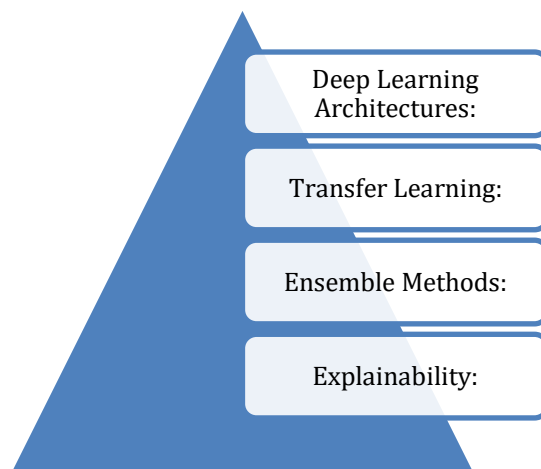
1.2 Components of Psoriasis Detection Systems



- **Image Acquisition:**
 - High-quality images of the affected skin areas are captured using digital cameras or smartphones. Dermatoscopes can be used to obtain detailed images of the skin lesions.
- **Preprocessing:**
 - Images are pre-processed to enhance quality and remove noise. Common preprocessing steps include normalization, resizing, and contrast adjustment.
- **Feature Extraction:**

- Critical features of psoriasis lesions, such as texture, color, and scaling patterns, are extracted. Techniques like edge detection, histogram analysis, and texture analysis (e.g., Gabor filters, Haralick features) are commonly used.
- **Classification:**
 - Machine learning algorithms classify the lesions based on the extracted features. Popular algorithms include Support Vector Machines (SVM), Random Forests, and Convolutional Neural Networks (CNNs).
 - CNNs, a type of deep learning model, have shown exceptional performance in image classification tasks. They automatically learn hierarchical features from raw image data, making them particularly suitable for medical image analysis.
- **Validation and Testing:**
 - The system is validated using labeled datasets, ensuring its reliability and accuracy. Metrics such as sensitivity, specificity, precision, and recall are used to evaluate performance.
- **Deployment:**
 - The final model can be deployed in various platforms, such as standalone software, web-based applications, or mobile apps. Integration with electronic health records (EHR) systems enhances usability in clinical settings.

1.3 Recent Advances and Future Directions



- **Deep Learning Architectures:**
 - Advances in CNN architectures, such as ResNet, Inception, and EfficientNet, have improved the performance of psoriasis detection systems. Hongfeng Li et al. delve into the critical realm of skin cancer diagnosis, acknowledging its global threat and the inherent challenges in its accurate identification [8]. It highlights the recent emergence of deep learning algorithms as potent tools in this domain, showcasing their remarkable performance across various diagnostic tasks. The review meticulously navigates through the landscape of deep learning methodologies applied in skin disease diagnosis. Beginning with a concise overview of skin diseases and dermatological image acquisition techniques, it proceeds to enumerate publicly available skin datasets. A comprehensive exploration of deep learning concepts ensues, elucidating popular architectures and frameworks instrumental in algorithm implementation. Evaluation metrics are then delineated, laying the groundwork for assessing algorithmic performance.

- **Transfer Learning:**

- Utilizing pre-trained models on large datasets and fine-tuning them on medical images helps in achieving higher accuracy with limited data. Li, Hongfeng in their review [9] explore the significant challenge of skin cancer diagnosis and the recent advancements in using deep learning algorithms for this purpose. It covers the basics of skin diseases and image acquisition methods, lists important datasets, discusses deep learning principles and architectures, and examines popular frameworks and evaluation metrics. The review critically analyzes existing literature on deep learning in skin disease diagnosis, highlights challenges, proposes future research directions, and concludes with a summary. Overall, it serves as a comprehensive guide to recent developments in this field, identifying both challenges and opportunities for further exploration.

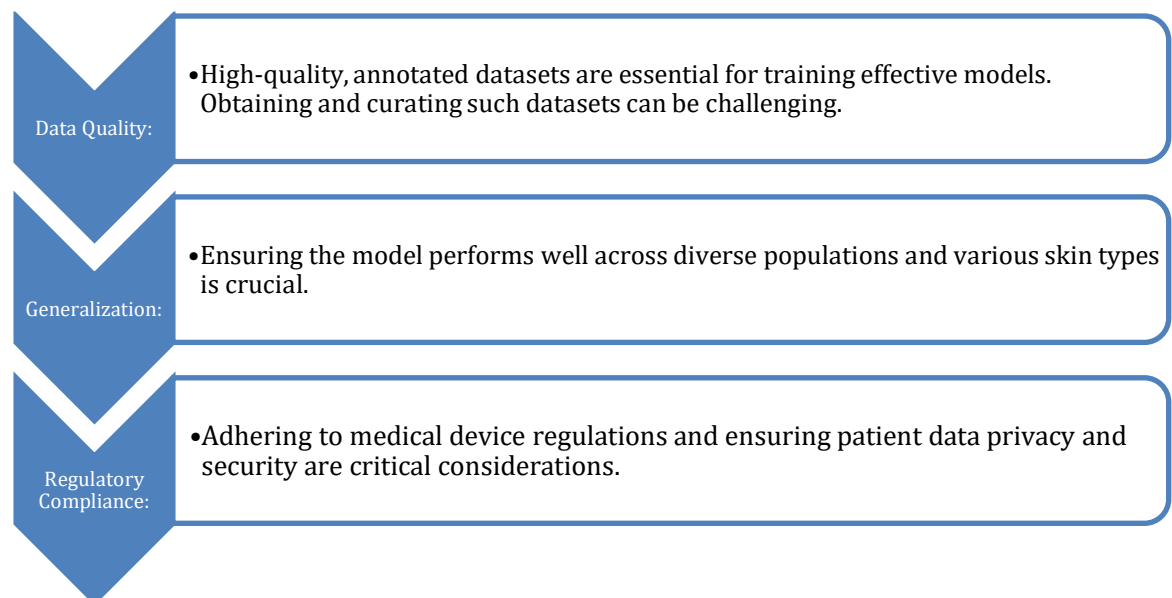
- **Ensemble Methods:**

- Combining predictions from multiple models can enhance robustness and accuracy. Thomas G. Dietterich proposed that [10], ensemble methods are learning algorithms that construct a set of classifiers and classify new data points by taking a (weighted) vote of their predictions. The original ensemble method is Bayesian averaging, but more recent algorithms include error-correcting output coding, Bagging, and Boosting. This review discusses these methods and explains why ensembles often perform better than any single classifier. It examines previous studies comparing ensemble methods and presents new experiments that investigate the reasons why Adaboost does not overfit rapidly.

- **Explainability:**

- Developing models that provide interpretable results is crucial for gaining the trust of healthcare professionals and patients.

1.4 Challenges and Considerations



Automated psoriasis detection systems hold great promise in revolutionizing the diagnosis and management of psoriasis. By leveraging advancements in machine learning and image processing, these systems can provide quick, accurate, and non-invasive diagnostic tools, improving patient outcomes and accessibility to care. Continued research and development in this field will further enhance the capabilities and adoption of these systems in clinical practice.

1.5 Reference for introduction

- [1] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639), 115-118.
- [2] Tschandl, P., Rinner, C., Apalla, Z., Argenziano, G., Codella, N., Halpern, A., ... & Kittler, H. (2020). Human–computer collaboration for skin cancer recognition. *Nature Medicine*, 26(8), 1229-1234.
- [3] Han, S. S., Kim, M. S., Lim, W., Park, G. H., Park, I., & Chang, S. E. (2018). Classification of the clinical images for benign and malignant cutaneous tumors using a deep learning algorithm. *Journal of Investigative Dermatology*, 138(7), 1529-1538.
- [4] Brinker, T. J., Hekler, A., Utikal, J. S., Grabe, N., Schadendorf, D., Klode, J., ... & Esser, S. (2019). Skin cancer classification using convolutional neural networks: Systematic review. *Journal of the American Academy of Dermatology*, 80(4), 1176-1186.
- [5] Parisi, R., Symmons, D. P. M., Griffiths, C. E. M., & Ashcroft, D. M. (2013). Global epidemiology of psoriasis: A systematic review of incidence and prevalence. *Journal of Investigative Dermatology*, 133(2), 377-385.
- [6] Takeshita, J., Grewal, S., Langan, S. M., Mehta, N. N., Ogdie, A., Van Voorhees, A. S., & Gelfand, J. M. (2017). Psoriasis and comorbid diseases: Epidemiology. *Journal of the American Academy of Dermatology*, 76(3), 377-390.
- [7] Elmetts, C. A., Leonardi, C. L., Davis, D. M. R., Gelfand, J. M., Lichtenstein, A., Mehta, N. N., ... & Korman, N. J. (2019). Joint AAD-NPF guidelines of care for the management and treatment of psoriasis with awareness and attention to comorbidities. *Journal of the American Academy of Dermatology*, 80(4), 1073-1113.
- [8] Hongfeng Li, Yini Pan, Jie Zhao, Li Zhang, Skin disease diagnosis with deep learning: A review, *Neurocomputing*, Volume 464, 2021, Pages 364-393,
- [9] Li, Hongfeng. (2020). Skin disease diagnosis with deep learning: a review.
- [10] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1-15). Springer, Berlin, Heidelberg.
- [11] Okhandiar RP, Banerjee BN. Psoriasis in the tropics: An epidemiological survey. *J Indian Med Assoc* 1963;41:550-6.
- [12] Bedi TR. Psoriasis in north India. Geographical variations. *Dermatologica* 1977;155:310-4.
- [13] Kaur I, Kumar B, Sharma VK, Kaur S. Epidemiology of psoriasis in a clinic from north India. *Indian J Dermatol Venereol Leprol* 1986;52:208-12.
- [14] Bedi TR. Clinical profile of psoriasis in North India. *Indian J Dermatol Venereol Leprol* 1995;61:202-5.
- [15] Kaur I, Handa S, Kumar B. Natural history of psoriasis: a study from the Indian subcontinent. *J Dermatol* 1997;24:230-4.

Chapter – 2

2. System Overview

Our proposed Skin diseases detection system, namely DermDetect is an advanced online platform designed for the detection and management of skin conditions using artificial intelligence. Below is an overview of the system's components and functionalities:

2.1 Intended purpose

DermDetect is software application tailored for analysing user-specific skin conditions. Its primary purpose is to detect Psoriasis in the early detection and ongoing monitoring of potential skin ailments. DermDetect degenerates a ranked list of up to probable conditions, prioritizing them by likelihood. This assists users in determining whether further consultation with a healthcare professional is advisable.

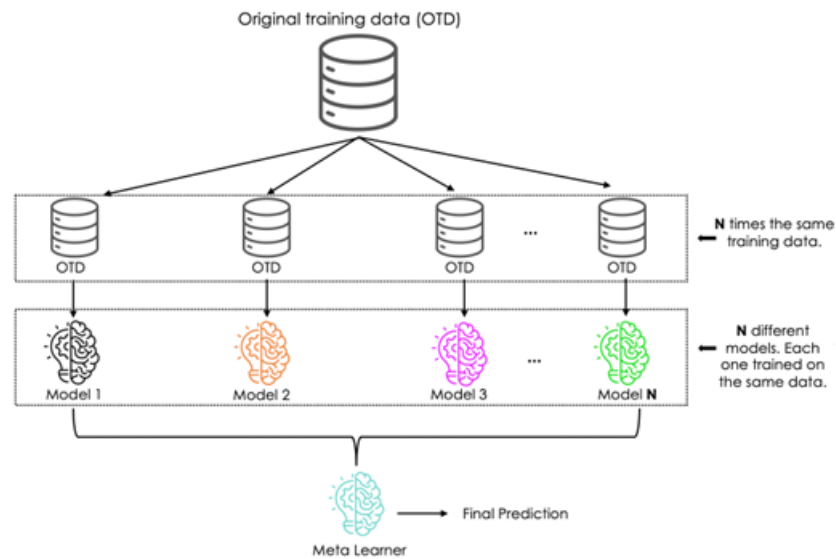
2.2 System Architecture



Deep Learning Algorithms: Utilizes state-of-the-art deep learning models for image analysis and skin condition detection. The specific models and techniques employed are optimized for high accuracy in diagnosing a variety of skin diseases.

Ensemble Learning: Combines multiple AI models to enhance prediction accuracy and reliability, leveraging the strengths of each model to mitigate individual weaknesses.

Skin Disease Detection System



2.3 Data Flow

Image Upload:

- Users can upload images of their skin condition through a web interface.

Preprocessing:

- The uploaded images are pre-processed to meet the input requirements of the AI models, including steps such as resizing, normalization, and enhancement.

Inference:

- The pre-processed images are analyzed by the ensemble of AI models to predict the skin condition.

Output:

- The system provides a detailed diagnosis, including the predicted condition and confidence scores.

2.4 Front-End Interface

User Interface

Responsive Design:

- The platform features a user-friendly, responsive design that works seamlessly across various devices.

Ease of Use:

- Simplified navigation and intuitive controls allow users to easily upload images and view results.

Features

Real-time Feedback:

- Provides immediate feedback and diagnosis upon image submission.

Educational Resources:

- Offers information on various skin conditions to help users understand their diagnosis.

2.5 Backend Infrastructure

Framework

Web Framework:

- Built using robust web technologies to ensure stability and scalability. Likely technologies include Django, Flask, or similar frameworks for backend development.

Database Management:

- Secure storage of user data and images, with a focus on privacy and compliance with data protection regulations.

Integration

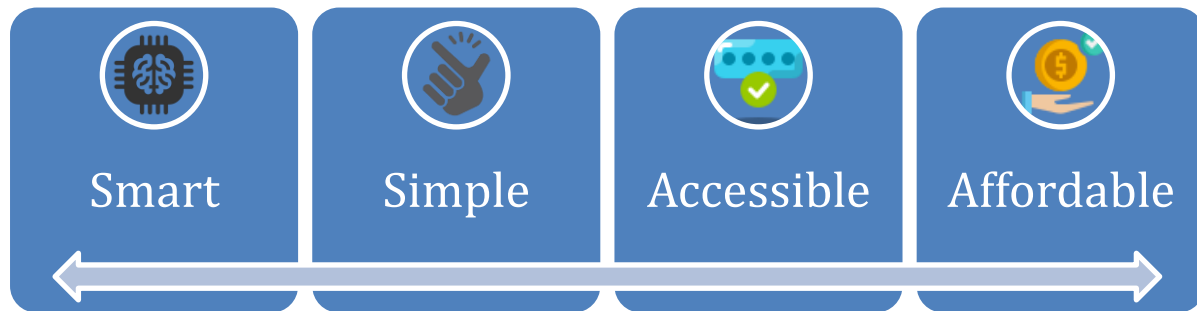
API Services:

- Integrates with AI model serving platforms to facilitate seamless model inference and data processing.

Scalability:

- Designed to handle a large number of concurrent users and image uploads, ensuring smooth operation under high demand.

2.6 System goal



Smart:

- DermDetect has been designed using modern technologies like machine learning to automate the process of detection of psoriasis, a very common skin disease in India. With the help of today's medical science knowledge and advanced technologies, our app has the ability to provide high accuracy results, as good as a professional dermatologist.

Simple:

- Place your phone's camera near the formation on the skin and click a clear photo, upload it to our system, and within 15 seconds you will find out if there is cause for concern.

Accessible:

- DermDetect is available anytime, anywhere. Keep your health in check at your fingertips even when you are on the go.

Affordable:

- DermDetect's leading image analytics features are absolutely free to use. The main objective behind building the system is to provide a seamless and easy user experience to our users so that they can verify their doubts about their skin irritations and get confirming results.

2.7 Intended users

DermDetect is intended to be used by:

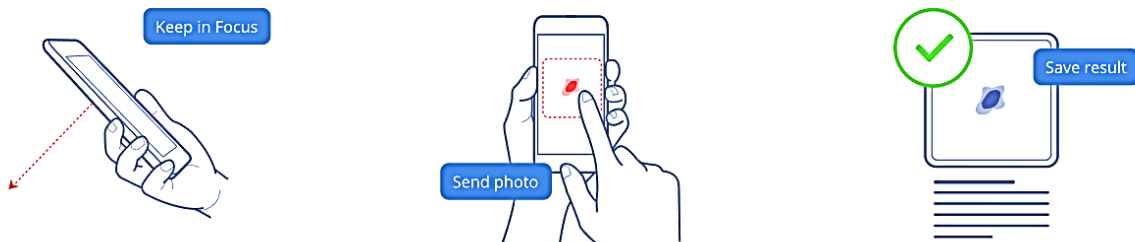
- Lay users with potential skin conditions.
- Children, who shall only use the app under supervision by a parent or guardian.
- Adults seeking initial advice on skin health concerns.
- Individuals looking to monitor changes in their skin over time.
- Users who want to keep a record of their skin condition for future reference during medical consultations.
- Caregivers assisting others with potential skin issues

2.8 Warnings and Precautions

The user needs to adhere to these rules:

- DermDetect does not provide a diagnosis or information for making diagnostic decisions and should not be used as a substitute for professional medical advice. Please see your doctor for a diagnosis. Always consult your dermatologist or another qualified healthcare professional before beginning any treatment. If you are experiencing a medical emergency, please call your emergency line.
- Ensure the photo(s) you submit are of good quality and the information you provide is accurate, as the accuracy of the services depends on the quality of this information.
- During the registration process, the responses entered by the user are crucial for achieving the best results. It is essential that this information is accurately and thoroughly provided by the user.

2.9 How to use the DermDetect

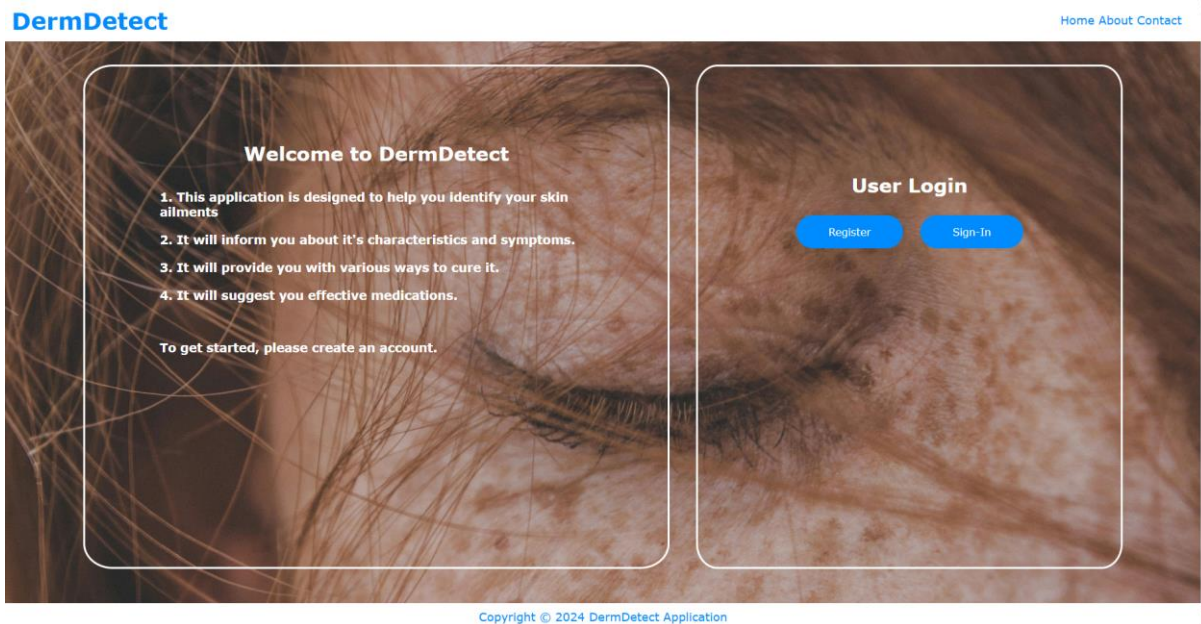


The DermDetect app is designed to be user-friendly, providing prompts and instructions at each step of the process.

While the app itself is self-explanatory, the following additional guidelines are available for users who may find them helpful.

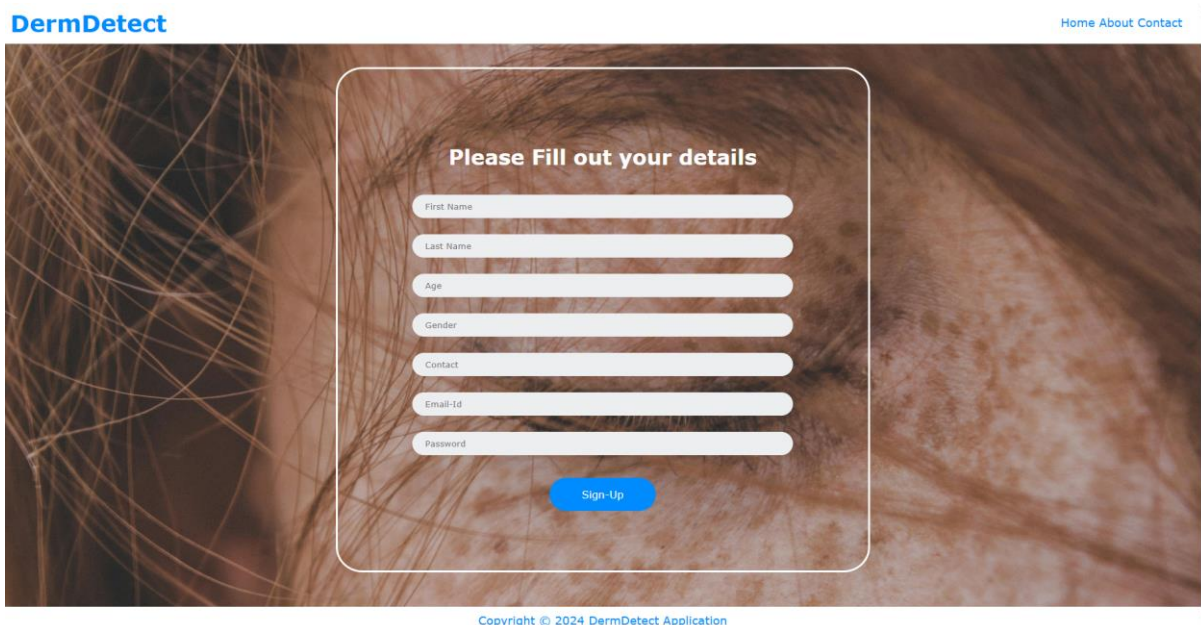
- First, launch the Triage app on your device and either create an account or log in with your existing credentials.
- Carefully follow the on-screen instructions and prompts provided throughout the process. Use a camera with a minimum sensor resolution of 10 megapixels to take clear and focused photos of the skin condition.
- Ensure all required information and photos are accurately provided to achieve the best possible results.
- Finally, review the guidance provided by the app and consider consulting with a healthcare professional for further advice.
- Remember, the additional guidelines are optional and meant to enhance your experience with the app.

Skin Disease Detection System



This is the home page of our DermDetect Application.

- It provides the user with an information panel on the left side.
- It familiarizes the user on how the application works. It informs the user with the steps to use the application.
- To start using the application the user has to first create an account using his/her Gmail account.
- On the right side of the page the user is given the options to 'Sign-In' if they already have an account, or go to the registration page to create an account.

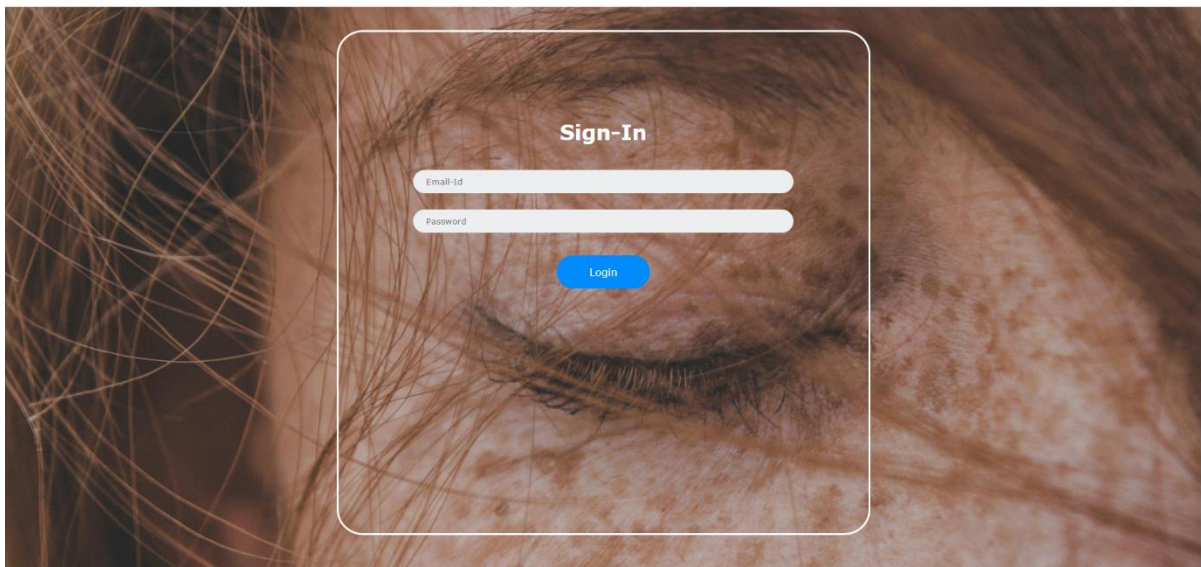


Create account

- The registration page provides a form that the user needs to fill up which will be used to create their account.
- The users will be asked for their name, age, gender, contact information.
- For authentication purposes they will be asked their 'Email-Id' and to create a 'Password'.
- Once they click on 'Sign-Up', the information will be stored discreetly and the user will be assigned a 'Patient-Id', which will be used to identify the user uniquely.
- There can only be one account per email-id. If a user tries to create multiple accounts with the same email-id they will receive an error message and the new account creation is denied.

DermDetect

[Home](#) [About](#) [Contact](#)

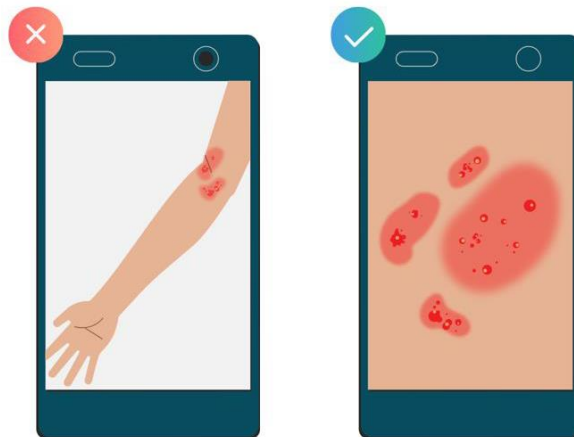


Copyright © 2024 DermDetect Application

Log-in

- The sign-in page has a simple form that asks for the user's email-id and their account password.
- Which will then be verified by the system in the backend to validate the user and then successfully log them in if the account exists.
- If the account does not exist, the user is shown an error message and sign-in is denied.

How to take a photo

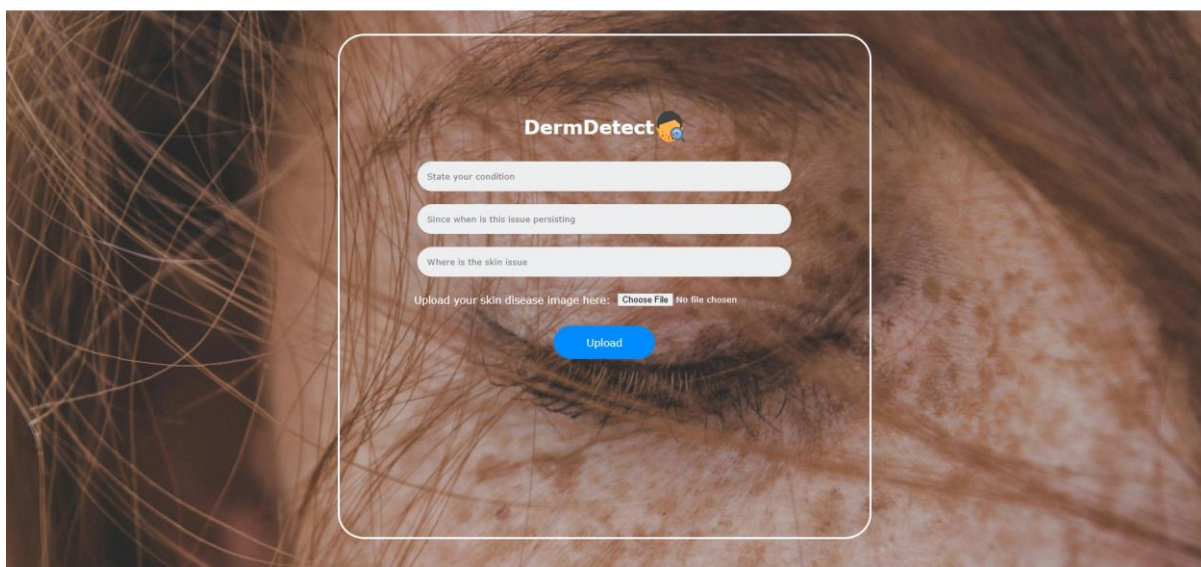


Take a photo and upload the image

- After inputting basic user information, the user will be prompted on how to best capture an image of the skin condition in question
- For users using a smartphone or tablet camera.
- The user will be prompted to take a photo and upload the image of their skin condition. Once the image is uploaded, the system will either accept the image or provide reason for why the image is not acceptable and request that a new image be uploaded.
- An accepted image will be displayed with brief guidance on how best to prepare the image for analysis through cropping and cantering the skin condition.
- Once completed properly, the user can click the “Next” link at the bottom of the page to continue to the next step of the process.

DermDetect

[Ayush](#) [Logout](#) [About](#) [Contact](#)



Skin Disease Detection System

Provide disease details

- Provide the application with details about the skin ailment, like it's current condition, it's location, and the time since it has occurred.
- This data can help the application to correctly identify the symptoms.

DermDetect

Ayush Logout About Contact

DermDetect

State your condition

Since when is this issue persisting

Where is the skin issue

Upload your skin disease image here: Choose File No file chosen

Upload

Copyright © 2024 DermDetect Application

Potential results screen

- The image uploaded will then be saved by the system.
- Then the image will be processed in the backend using machine learning algorithms and the result will be shared with the user
- The user will also get an option to download the report for future references.

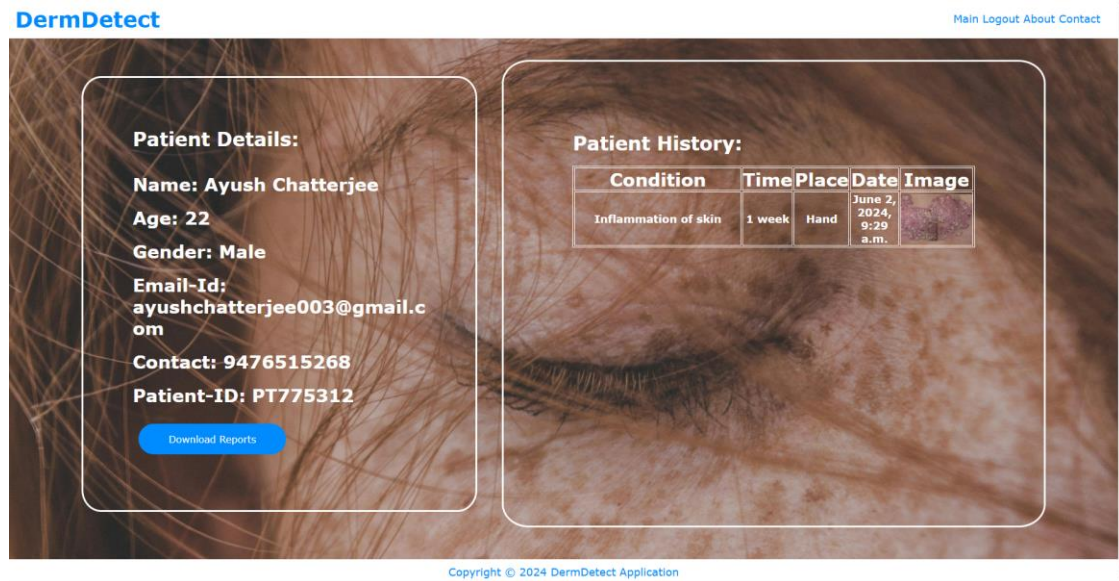
Name: Ayush Chatterjee
Age: 22
Gender: Male
Email-ID: ayushchatterjee003@gmail.com
Contact: 9476515268
Patient-ID: PT775312

Report: 1
Date: 2024-06-02 09:29:05.803812+00:00
Condition: Inflammation of skin
Time: 1 week
Place: Hand
Image:



Results: You have Psoriasis

Skin Disease Detection System



Access User profile and history

- The users can access this page to get details about their account and their past medical conditions. This page helps them to find details about all their cases that they have reported to this application.
- The left side of the page shows the patient details which includes name, age, gender, email-id, contact and Patient-ID.
- The right side of the page shows all the details of the previous records of the user, along with the images.
- All the previous diagnosis reports can be downloaded from here too.

2.10 System requirements

Hardware Requirements:

- **Smartphone with Camera:**
 - Resolution: Minimum 8 MP camera for clear and detailed skin images.
 - Flash: Recommended for better image clarity in low light conditions.
- **Processor:**
 - Minimum: Quad-core processor.
 - Recommended: Octa-core processor for faster image processing.
- **RAM:**
 - Minimum: 2 GB of RAM.
 - Recommended: 4 GB of RAM or more for optimal performance.

Software Requirements:

- **Operating System:**
 - Android: Version 8.0 (Oreo) or higher.
 - iOS: Version 12.0 or higher.
- **Internet Connectivity:**
 - Required for uploading images and receiving analysis results.
 - Minimum: 3G connection.
 - Recommended: 4G/LTE or Wi-Fi for faster uploads and downloads.
- **Permissions:**
 - Camera access for taking photos.
 - Storage access for saving photos and app data.
 - Internet access for data transmission and receiving results.

Skin Disease Detection System

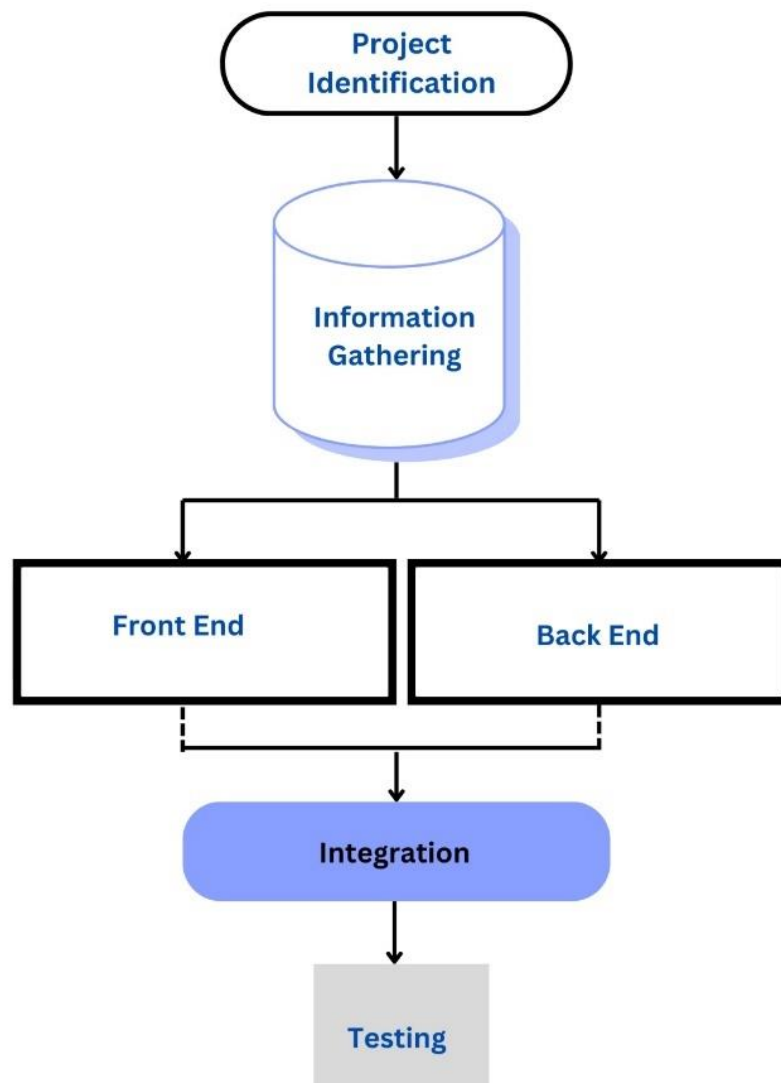
- **Browser Requirements:**

- Google Chrome: Version 90 or higher.
- Mozilla Firefox: Version 88 or higher.
- Safari: Version 14 or higher.
- Microsoft Edge: Version 90 or higher.
- Internet Explorer: Version 11 (limited support, upgrade recommended).

Chapter – 3

3. System Design

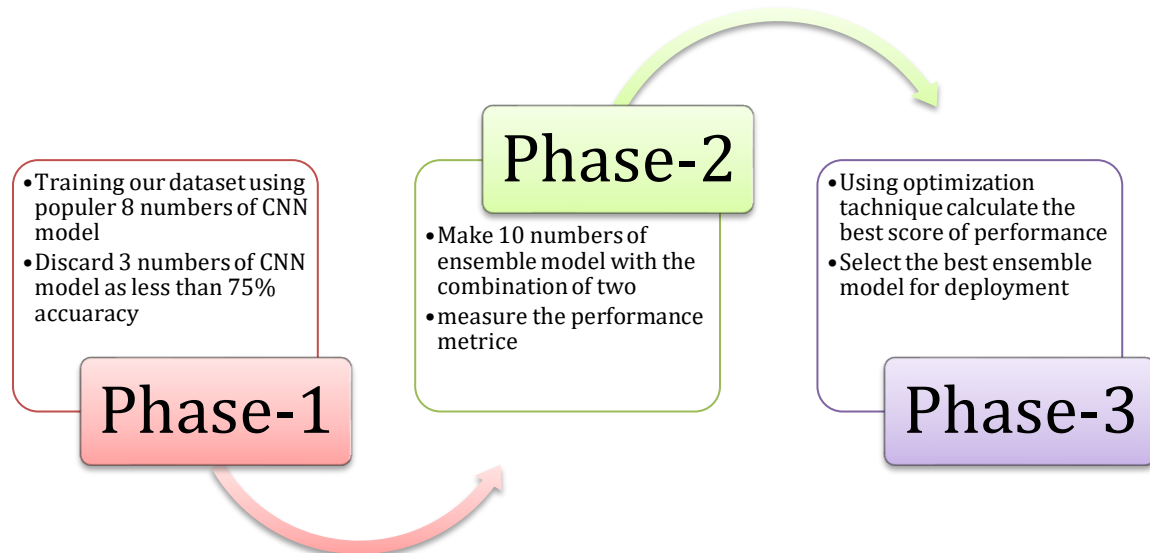
The design of the Skin Disease Detection System is divided into two main parts: a Django-based front end and a machine learning-driven backend. The front end serves as the user interface, enabling user authentication, image upload functionality, and the display of diagnostic results. It comprises a home page with system information, an upload page for submitting skin images, a results page for viewing diagnoses, and a history page for tracking previous submissions and results. The backend is responsible for processing the uploaded images through steps such as image preprocessing, and loading and executing pre-trained machine learning models for skin disease detection. The processed results are then returned to the front end for user display. This integrated design ensures seamless interaction, allowing users to effortlessly upload images and receive accurate diagnostic feedback.



Chapter – 4

4. Back-end design

We design the back end in three phases

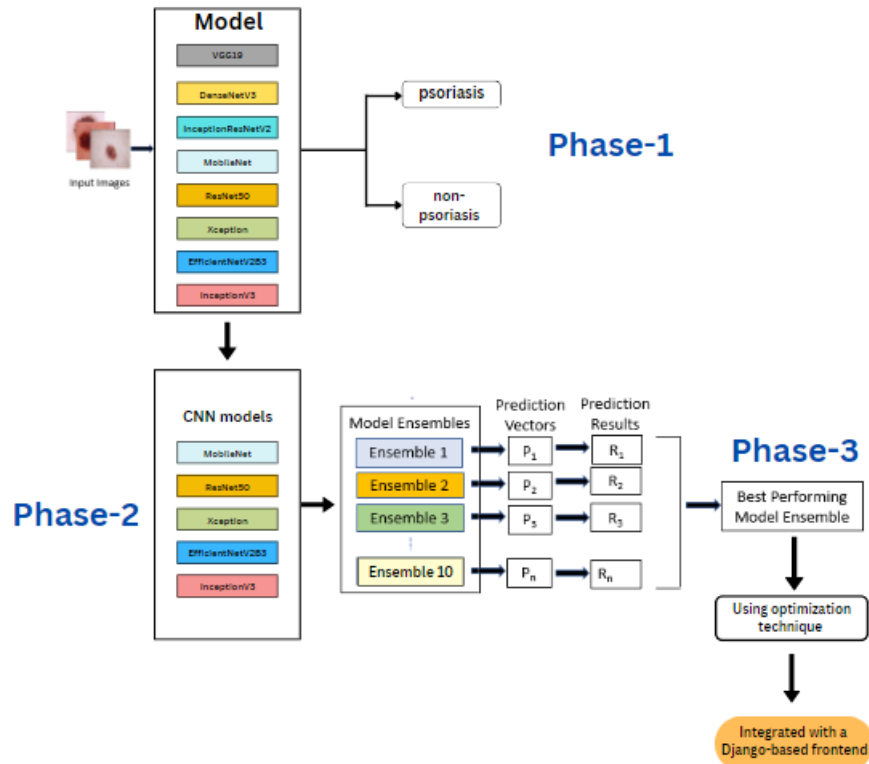


In the first phase, we selected eight popular pre-trained models from Keras and trained them using our dataset. We then evaluated the accuracy of each model. Based on the results, we discarded three models that achieved an accuracy of less than 75%.

In the second phase of our backend development, we took the five models that each achieved over 75% accuracy in the initial evaluation, we combined two of these models at a time, resulting in a total of ten ensemble models. These ensembles produced prediction vectors and results.

In the third phase, we observed that each model in the ensemble exhibited varying performance metrics. To identify the best-performing model, we applied an optimization technique using a weighted sum approach. This technique allowed us to systematically evaluate and select the model with the optimal balance of performance metrics, ensuring the highest overall accuracy and reliability for the skin disease detection system.

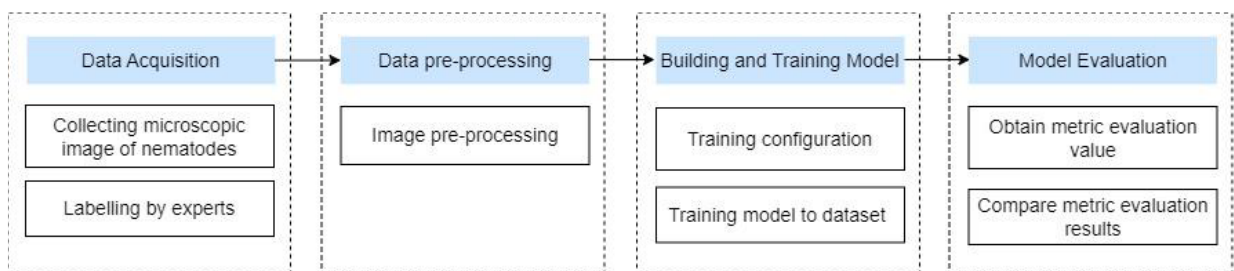
Skin Disease Detection System



4.1 Phase 1: Training the dataset with 8 popular CNN model

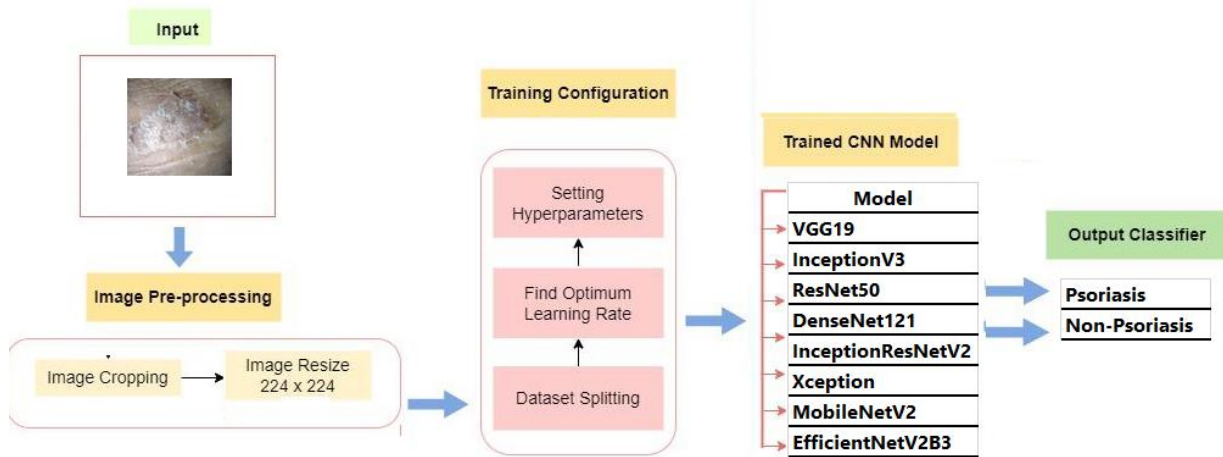
4.1.1 Research Workflow

The main procedure for this investigation is shown in Figure 1. The psoriasis dataset was first gathered from the doctor's office and categorised by a him prior to additional processing. Several pictures pre-processing methods, including edge detection, cropping, and grayscale conversion, are included in the data pre-processing. The chosen CNN models are then built and configured using a number of hyperparameter settings: DenseNet121, EfficientNetV2B3, InceptionV3, ResNet50v2, VGG19, and Xception. The CNN models are trained using the pre-processing findings as input. To determine which CNN models are the best, the results are acquired and compared using a variety of assessment criteria, including test accuracy, F1, precision, and recall, among others.



4.1.2 CNN Implementations.

Figure 4 shows the procedure for utilising CNN models to train the Psoriasis data. The CNN architecture's input picture was created using the data acquisition outcome from the process shown in Figure 2. Next, the photos underwent pre-processing utilising the identical technique as other studies. To determine the border, edge detection was used. During training, the optimizer will decide how weight changes are implemented. In order to train deep convolutional neural networks, this research uses momentum and SGD because of its improved validation and test outcomes. Additionally, SGD is more generalizable to unobserved data.



Convolutional neural network training requires careful consideration of hyperparameters since they have a direct impact on the actions of training algorithms and greatly affect the performance of the models [20]. Every model will be trained with the identical hyperparameter values for batch size, activation layer, loss function, and epoch in order to maintain consistency. Because of its enhanced generalisation performance and memory saving, a batch size of 32 was used for the CNN models.

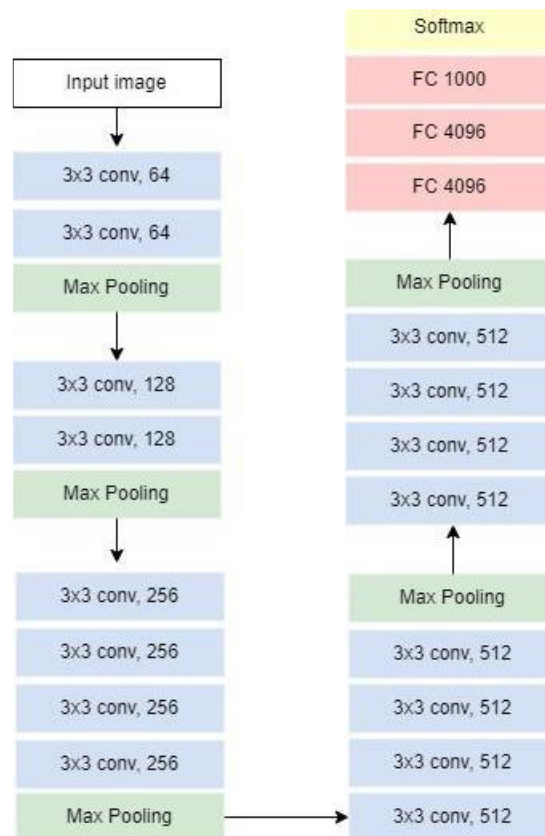
Training convolutional neural network models is best done with large datasets. Still, the overall data set in this study is deemed limited. Reducing overfitting resulting from fewer datasets is a common use of transfer learning. This technique transfers information to the fresh psoriasis dataset from the model learned on the ImageNet dataset. The computational cost of creating the architecture prevented this study from starting from scratch when it came to training.

This work examines eight widely used pre-trained deep learning models from Keras, which are learned with weights from ImageNet. Based on the ImageNet benchmark's picture classification performance and computational constraints, the models were chosen. InceptionV3, ResNet50, DenseNet121, InceptionResNetV2, Xception, MobileNetV2, and EfficientNetV2B3 are the models that were chosen. By using convolutional architecture, all models have the same underlying architecture. Tables 2 and 3 provide a summary of the parameters, an internal value associated with the model architecture, and the central spine of each convolutional neural network architecture used in this investigation. After that, every model was trained on Google Colab Pro using the NVIDIA P100 or T4 GPU specs, a Xeon CPU running at 2.3GHz, and up to 25GB of RAM, depending on availability.

4.1.3 VGG19

It is a convolutional neural network model developed by the Visual Geometry Group (VGG) at the University of Oxford. It is known for its simplicity and depth, comprising 19 layers with a uniform architecture that uses small 3x3 convolutional filters throughout the network. VGG19 is widely used in image classification tasks due to its balance of depth and simplicity, which allows it to capture complex features effectively.

Skin Disease Detection System



Key Features of VGG19

- **Simple and Uniform Architecture:** VGG19 employs a straightforward design with 3x3 convolutional layers stacked on top of each other, making it easier to implement and modify.
- **Depth:** With 19 layers, VGG19 is deep enough to learn complex features and hierarchical representations from images.
- **Pre-trained Models:** Availability of pre-trained models on large datasets like ImageNet allows for transfer learning, making it easier to adapt VGG19 for specific tasks with limited data.

Reason to choose VGG19 for Skin Disease Detection

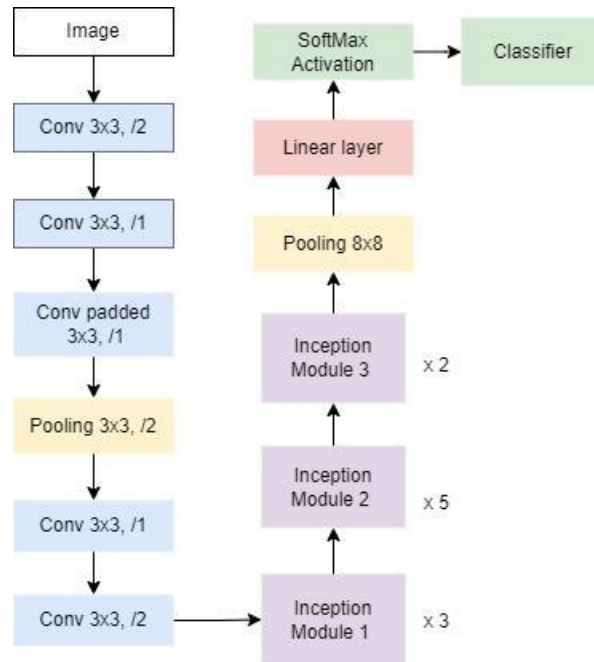
- **High Accuracy:** VGG19's depth enables it to capture intricate details in skin lesion images, leading to high accuracy in classifying various skin conditions.
- **Transfer Learning:** The use of pre-trained VGG19 models allows for quick adaptation to the specific domain of skin disease detection, leveraging knowledge from extensive datasets to improve performance even with smaller, specialized datasets.
- **Robust Feature Extraction:** The network's ability to extract robust features from images makes it well-suited for distinguishing between different types of skin lesions, which is critical for accurate diagnosis.

Sources

- Simonyan, K., & Zisserman, A. (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition." arXiv preprint arXiv:1409.1556.

4.1.4 InceptionV3

InceptionV3 is a convolutional neural network architecture developed by Google as part of the Inception family, specifically designed for image classification tasks. It builds on the success of previous Inception models by introducing several enhancements that improve both efficiency and performance.



Key Features of InceptionV3

- **Inception Modules:** These modules combine multiple convolutional operations with different kernel sizes (1x1, 3x3, 5x5) in parallel within each module, allowing the network to capture various spatial features and patterns at different scales effectively.
- **Factorized Convolutions:** InceptionV3 employs factorized convolutions, such as splitting a 7x7 convolution into two 3x3 convolutions, which reduces computational cost and allows for deeper networks.
- **Auxiliary Classifiers:** Intermediate auxiliary classifiers help improve gradient flow during training, encouraging the network to learn better features and aiding in faster convergence.
- **Efficient Grid Size Reduction:** By replacing large pooling operations with more efficient convolutions, InceptionV3 maintains computational efficiency without sacrificing performance.

Reason to choose InceptionV3 for Skin Disease Detection

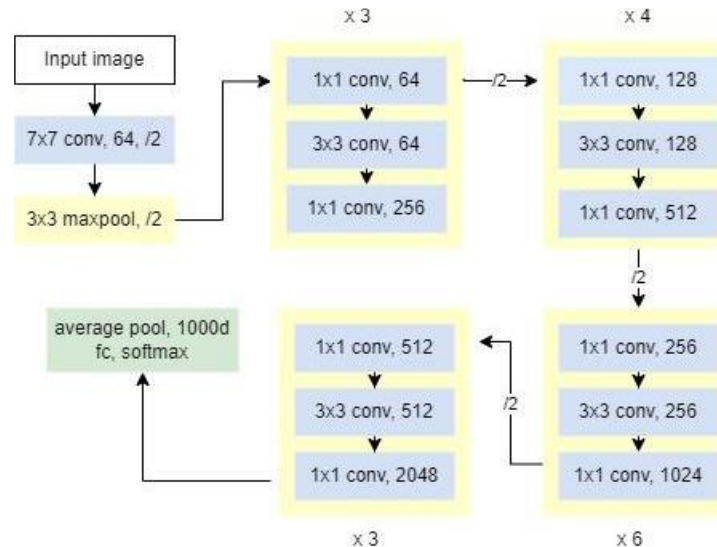
- **High Accuracy:** The multi-scale feature extraction capability of InceptionV3 allows it to accurately identify and classify various skin conditions, making it highly effective for diagnosing skin diseases.
- **Efficiency:** The architectural optimizations in InceptionV3 ensure that it can run efficiently even on devices with limited computational resources, such as smartphones and tablets. This is essential for telemedicine and mobile health applications.
- **Robustness:** InceptionV3's design makes it robust to variations in image quality and presentation, ensuring reliable performance across diverse clinical settings and different types of skin lesion images.

Sources

- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). "Rethinking the Inception Architecture for Computer Vision." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2818-2826.

4.1.5 ResNet50

ResNet50 (Residual Network with 50 layers) is a deep convolutional neural network developed by Microsoft Research. It is part of the ResNet family, which introduced the concept of residual learning to address the vanishing gradient problem in training deep networks. ResNet50 is known for its powerful feature extraction capabilities and robustness in various image classification tasks.



Key Features of ResNet50

- **Residual Blocks:** These blocks use shortcut connections to bypass one or more layers, allowing gradients to flow directly through the network. This helps in training very deep networks without degradation.
- **Deep Architecture:** With 50 layers, ResNet50 can learn complex features and representations from images, making it highly effective for detailed image analysis.
- **Efficiency:** Despite its depth, ResNet50 is designed to be computationally efficient, making it suitable for various applications, including those requiring high-performance real-time processing.

Reason to choose ResNet50 for Skin Disease Detection:

- **High Accuracy:** The depth and residual connections enable ResNet50 to achieve high accuracy in distinguishing between various skin conditions, capturing intricate patterns in skin lesion images.
- **Transfer Learning:** Pre-trained ResNet50 models on large datasets like ImageNet can be fine-tuned for skin disease detection, leveraging prior knowledge to improve performance on medical image datasets with limited samples.
- **Robustness:** ResNet50's architecture is robust to overfitting and can generalize well to new, unseen data, which is crucial for reliable skin disease diagnosis in diverse clinical settings.

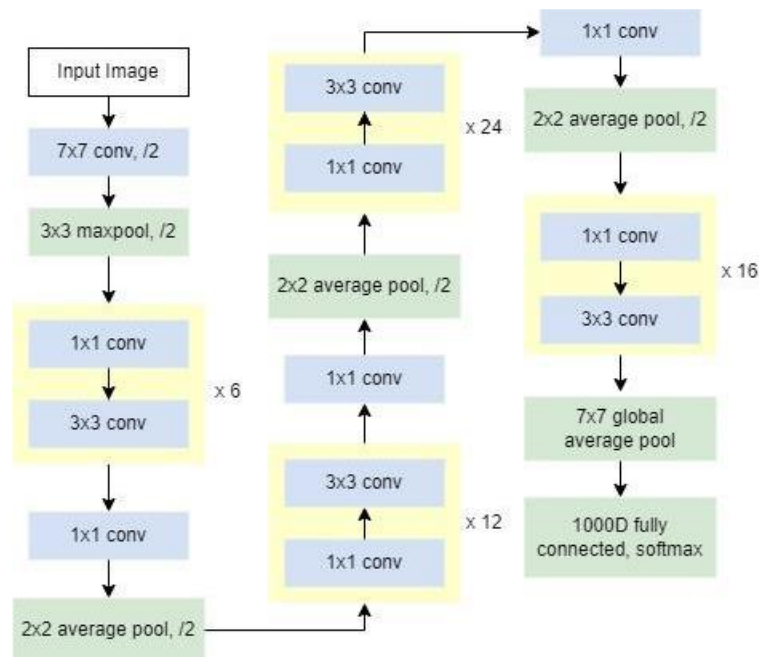
Sources:

- He, K., Zhang, X., Ren, S., & Sun, J. (2016). "Deep Residual Learning for Image Recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

4.1.6 DenseNet121

The foundation of Dense Convolutional Networks (DenseNet) is the feed-forward link between each layer [28]. Numerous noteworthy benefits are provided by this neural network design, such as the resolution of the vanishing-gradient problem, enhanced feature propagation, enhanced feature reuse, and greatly reduced parameter needs. The networks are segmented into several densely linked units, sometimes

known as dense blocks, in order to enable the feature map altering downsampling layers. The version of this model that was different was the dense block's network setup. Figure shows the DenseNet121 setup.



Key Features of DenseNet121:

- **Dense Connectivity:** Each layer in DenseNet121 is connected to every other layer in a feed-forward fashion, enabling maximum information and gradient flow between layers.
- **Feature Reuse:** By concatenating feature maps from previous layers, DenseNet121 promotes the reuse of features, leading to improved parameter efficiency and reduced overfitting.
- **Compact Model Size:** Despite its depth, DenseNet121 is relatively compact in terms of the number of parameters, which helps in faster training and inference times.

Reason to choose for DenseNet121 for Skin Disease Detection:

- **High Accuracy:** The dense connectivity pattern allows DenseNet121 to capture intricate patterns in skin lesion images, leading to high accuracy in diagnosing various skin conditions.
- **Efficiency:** DenseNet121's compact model size and efficient feature reuse make it suitable for deployment on devices with limited computational resources, such as mobile phones and tablets, which are essential for telemedicine applications.
- **Robustness:** The model's ability to effectively propagate gradients and reuse features ensures robust performance across different datasets, which is crucial for reliable skin disease diagnosis in diverse clinical settings.

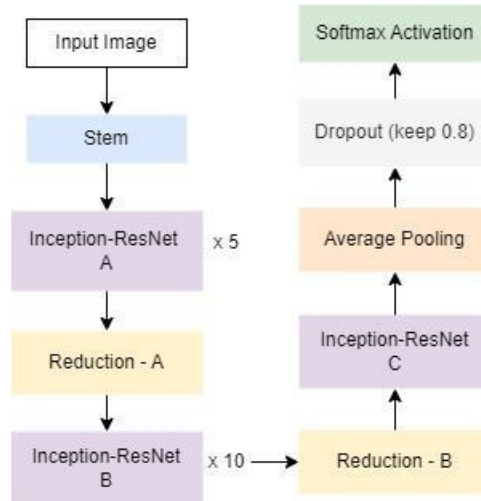
Sources:

- Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). "Densely Connected Convolutional Networks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4700-4708.

4.1.7 InceptionResNetV2

InceptionResNetV2 is a convolutional neural network architecture that combines the strengths of Inception modules and residual connections. Developed by Google, this model integrates the multi-scale feature extraction capabilities of Inception networks with the training efficiency and improved gradient flow of residual networks (ResNets).

Skin Disease Detection System



Key Features of InceptionResNetV2

- **Inception Modules:** These modules capture features at multiple scales by performing parallel convolutions with different kernel sizes, effectively capturing spatial hierarchies and patterns.
- **Residual Connections:** Residual connections help in addressing the vanishing gradient problem by allowing gradients to flow through the network more effectively, leading to easier training of deeper networks.
- **Hybrid Architecture:** The combination of Inception and ResNet architectures allows InceptionResNetV2 to leverage the advantages of both approaches, achieving high performance with efficient computation.

Reason to choose InceptionResNetV2 for Skin Disease Detection

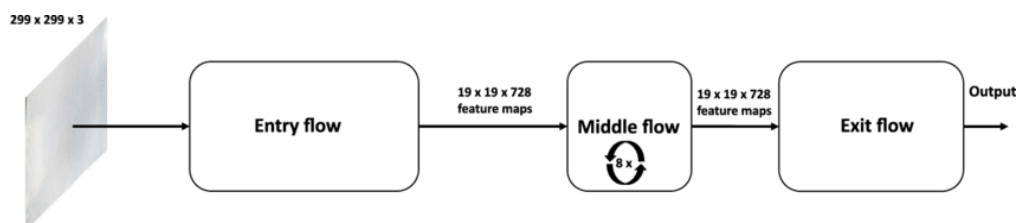
- **High Accuracy:** InceptionResNetV2's hybrid architecture allows it to capture complex features and patterns in skin lesion images, leading to high accuracy in diagnosing a variety of skin conditions.
- **Efficient Training:** The residual connections facilitate better gradient flow, making it easier to train deeper networks and improving the model's ability to learn from medical image datasets.
- **Robust Feature Extraction:** The model's ability to extract robust multi-scale features ensures reliable performance across different skin lesion types and image qualities, which is crucial for accurate diagnosis.

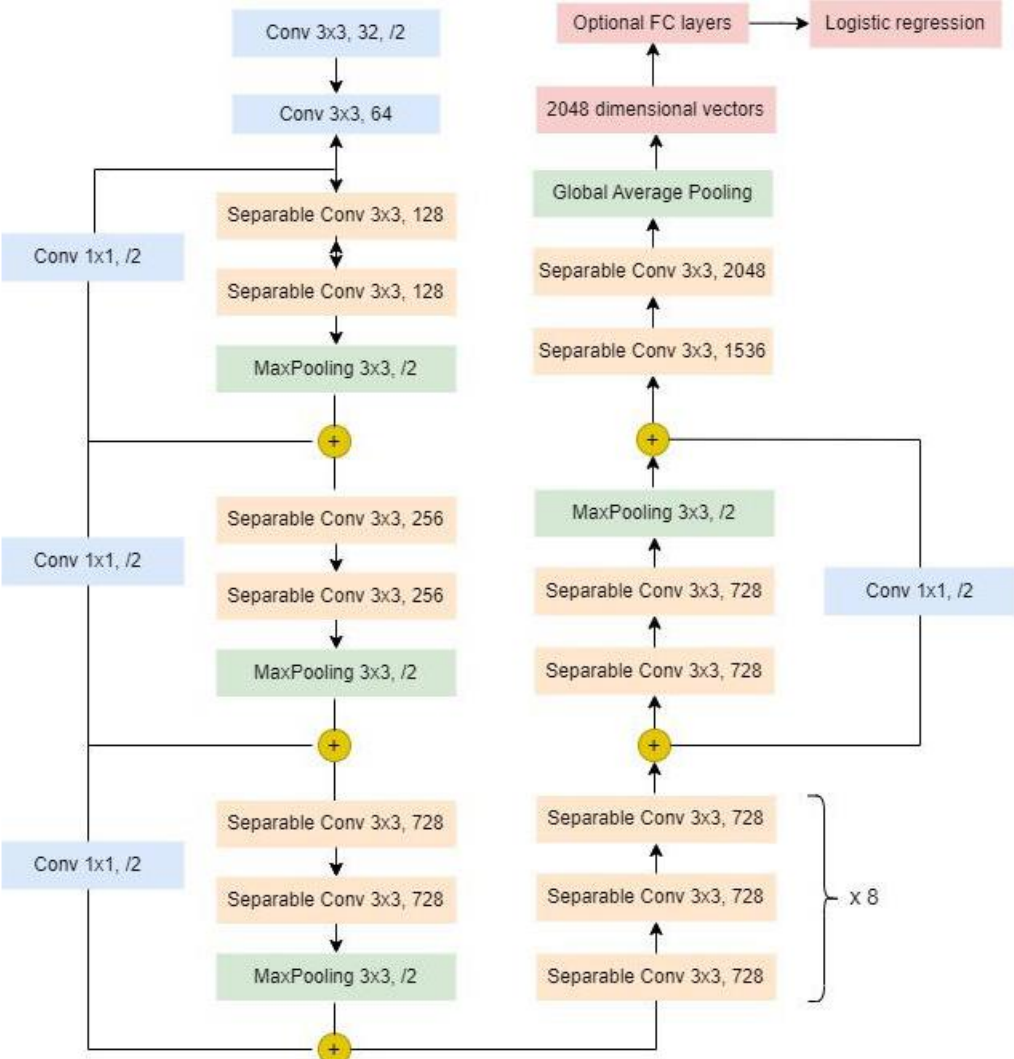
Sources

- Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017). "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning." Proceedings of the AAAI Conference on Artificial Intelligence.

4.1.8 Xception

Xception (Extreme Inception) is a deep convolutional neural network architecture developed by François Chollet. It builds upon the Inception architecture but replaces the standard Inception modules with depthwise separable convolutions, making the network both more efficient and performant. Xception leverages the power of depthwise separable convolutions to reduce model complexity while increasing computational efficiency, leading to enhanced performance on various image classification tasks.





Key Features of Xception:

- **Depthwise Separable Convolutions:** These convolutions split the convolution operation into two parts: depthwise convolutions that apply a single filter to each input channel, followed by pointwise convolutions that apply a 1x1 convolution to combine the outputs of the depthwise layer. This significantly reduces computational cost and model parameters.
- **Residual Connections:** Similar to ResNet, Xception incorporates residual connections, which help in training deeper networks by allowing gradients to flow more effectively through the network, reducing the vanishing gradient problem.
- **Efficient Architecture:** The combination of depthwise separable convolutions and residual connections results in a highly efficient architecture that delivers high performance on complex tasks while maintaining a relatively low computational overhead.

Reason to choose Xception for Skin Disease Detection

- **High Accuracy:** Xception's architecture is designed to achieve state-of-the-art accuracy on image classification tasks, making it well-suited for the precise identification of various skin conditions.
- **Efficiency:** Despite its depth and complexity, Xception is computationally efficient due to the use of depthwise separable convolutions. This efficiency allows for faster inference times, which is critical for real-time applications in skin disease detection.

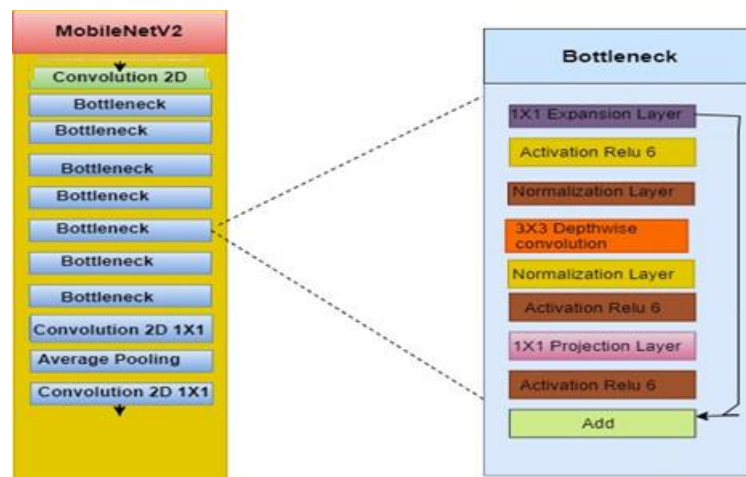
- Scalability: Xception can be scaled effectively for more complex tasks without a significant increase in computational requirements, making it suitable for handling high-resolution skin images and a large variety of skin disease categories.

Sources

- Chollet, F. (2017). "Xception: Deep Learning with Depthwise Separable Convolutions." arXiv preprint arXiv:1610.02357.
- Szegedy, C., et al. (2016). "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning." arXiv preprint arXiv:1602.07261.

4.1.9 MobileNetV2

MobileNetV2 is a lightweight and efficient convolutional neural network architecture tailored for mobile and resource-constrained environments. Introduced by Sandler et al. (2018), it improves upon its predecessor, MobileNetV1, by using depth wise separable convolutions and introducing an inverted residual structure with linear bottlenecks. This design significantly reduces the number of parameters and computational complexity while maintaining high accuracy and robustness in visual tasks.



Key Features of MobileNetV2:

- Depth wise Separable Convolutions: This technique splits the convolution into two layers, a depth wise convolution and a pointwise convolution, reducing the computation required compared to standard convolutions.
- Inverted Residuals: These structures help in maintaining the model's performance by effectively capturing spatial features with fewer parameters.
- Linear Bottlenecks: The linear bottleneck layer between the inverted residual blocks helps prevent loss of information and improves the representational capacity of the network.

Cause of MobileNetV2 for Skin Disease Detection:

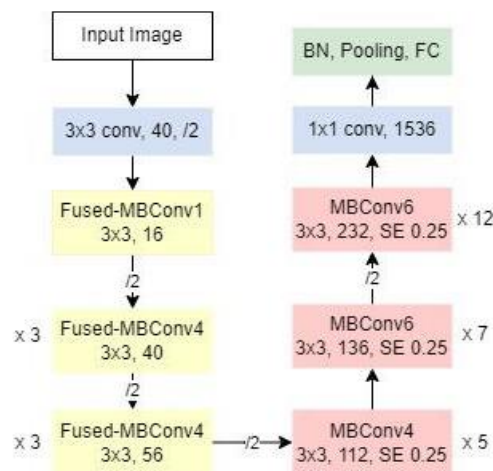
- Efficiency: MobileNetV2's architecture allows for real-time analysis and diagnosis on devices with limited computational power, such as smartphones and embedded systems. This is crucial for deploying skin disease detection systems in remote and underserved areas where access to advanced medical equipment is limited.
- Accuracy: Despite its lightweight nature, MobileNetV2 maintains high accuracy in image classification tasks, making it suitable for detecting a variety of skin conditions with high reliability.
- Portability: The compact size of MobileNetV2 models facilitates easy deployment in mobile applications, enabling on-the-go skin disease diagnosis without requiring constant internet connectivity or powerful hardware.

Sources:

- Howard, A.G., et al. (2017). "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." arXiv preprint arXiv:1704.04861.
- Sandler, M., et al. (2018). "MobileNetV2: Inverted Residuals and Linear Bottlenecks." arXiv preprint arXiv:1801.04381.

4.1.10 EfficientNetV2B3

- By using a smaller model and a faster convergence time than previous architectures, the EfficientNet model improves performance. This model's core scales the model's resolution, depth, and breadth to enhance the model's capacity using a novel scaling method with an easy-to-understand compound coefficient [29]. Neural Architecture Search (NAS) is then used to generate a new baseline model using MBConv blocks, which is subsequently scaled using the compound coefficient to create EfficientNet [30]. The fundamental properties of the network, MBConv and Fused-MBConv, reduce model size while improving training efficiency.
- **EfficientNetV2B3** is part of the EfficientNetV2 family, a series of convolutional neural networks that build on the success of the original EfficientNet models. These models are designed to achieve a balance between accuracy and efficiency through a combination of neural architecture search (NAS) and advanced scaling techniques. EfficientNetV2B3 specifically is known for its improved training speed and performance compared to earlier versions.



Key Features of EfficientNetV2B3:

- **Compound Scaling:** EfficientNetV2B3 employs compound scaling, which uniformly scales network depth, width, and resolution. This approach ensures a balanced increase in model size, leading to better accuracy and efficiency.
- **Neural Architecture Search (NAS):** The architecture is derived from NAS, which optimizes the model structure for high performance and resource efficiency.
- **Advanced Techniques:** Incorporates techniques like Fused-MBConv, improved regularization, and progressive learning to enhance training speed and accuracy.

Why EfficientNetV2B3 for Skin Disease Detection:

- **High Accuracy:** EfficientNetV2B3 provides state-of-the-art performance in image classification tasks, making it well-suited for accurately identifying various skin conditions from images.
- **Efficiency:** The model's optimized architecture allows for efficient computation, making it suitable for deployment on devices with limited resources, such as mobile phones and tablets, which are essential for telemedicine applications.
- **Faster Training:** EfficientNetV2B3's improved training speed enables quicker adaptation to new datasets, facilitating faster development of skin disease detection systems.

Skin Disease Detection System

Sources:

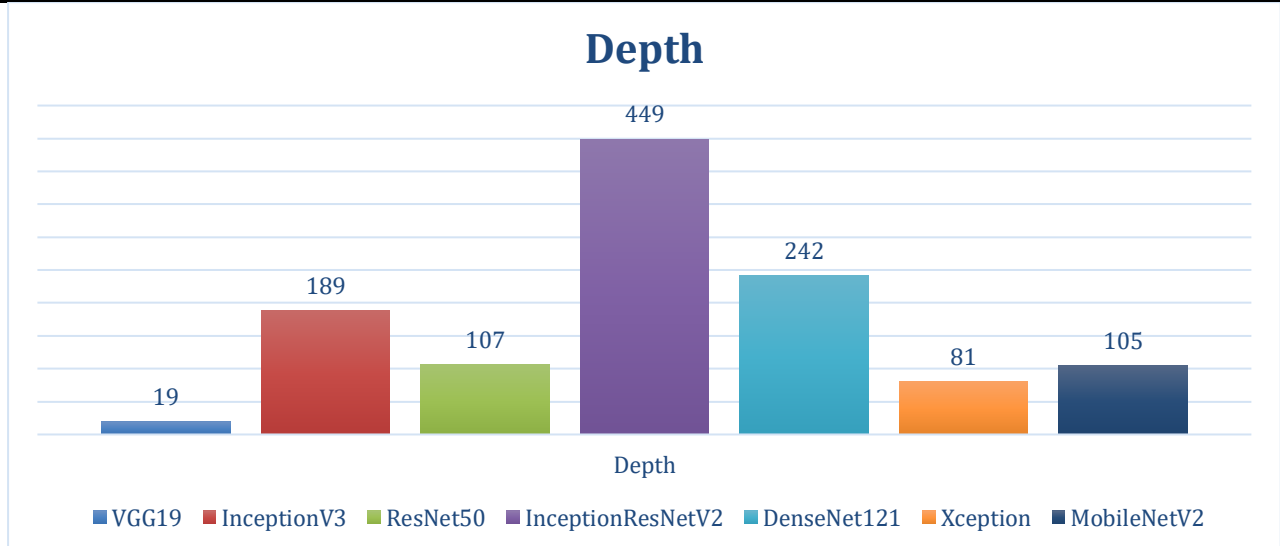
Tan, M., & Le, Q. (2021). "EfficientNetV2: Smaller Models and Faster Training." arXiv preprint arXiv:2104.00298.

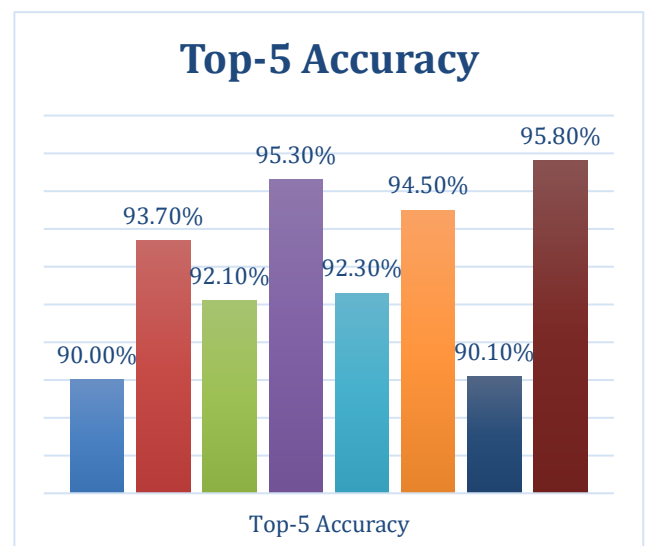
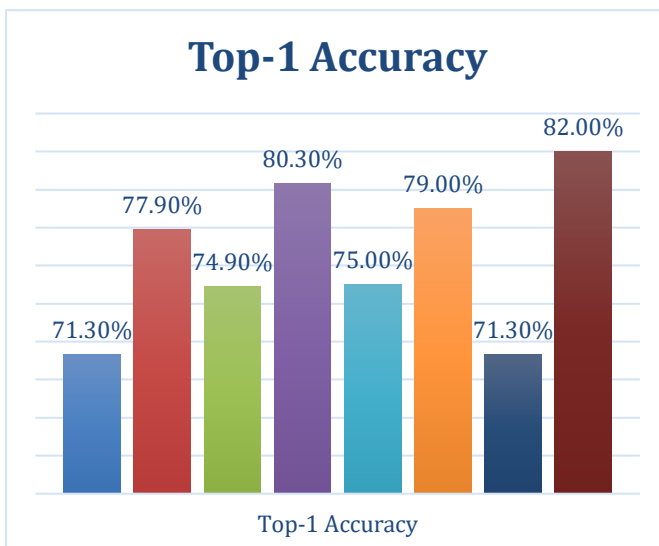
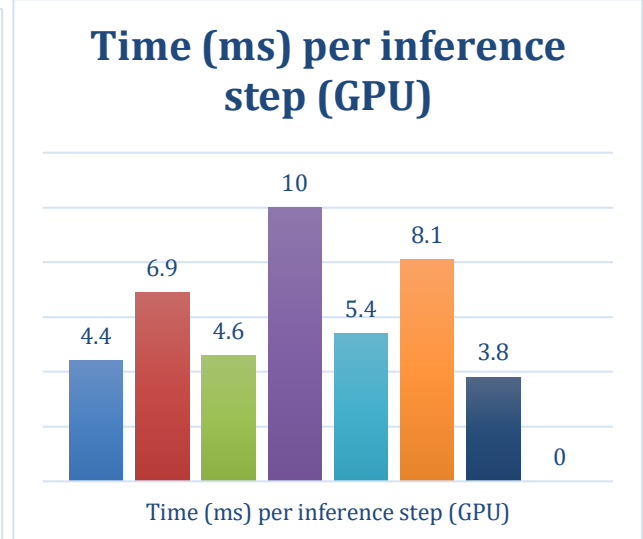
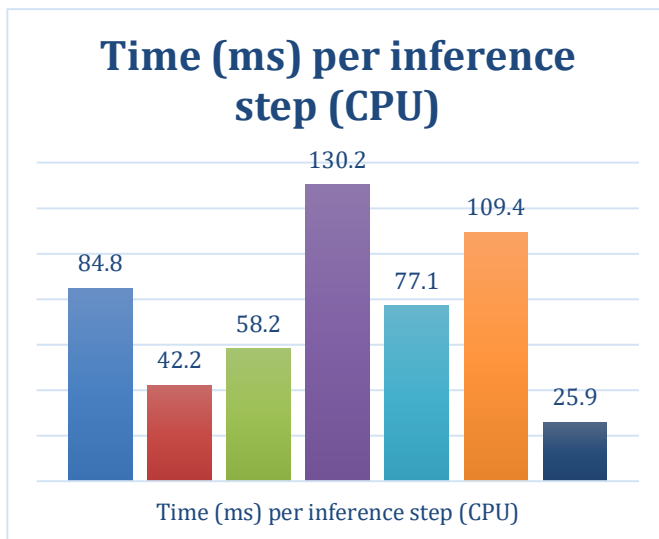
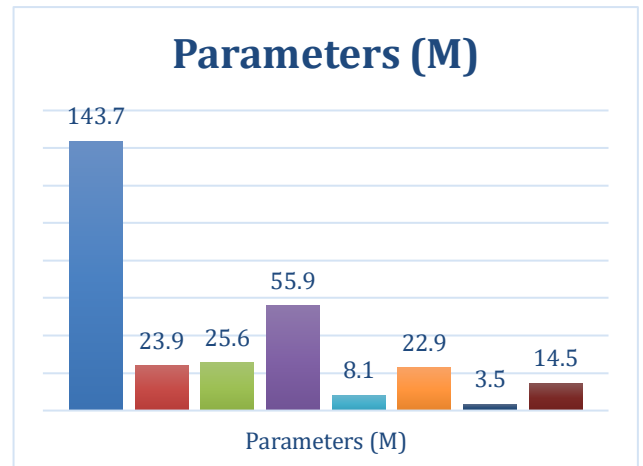
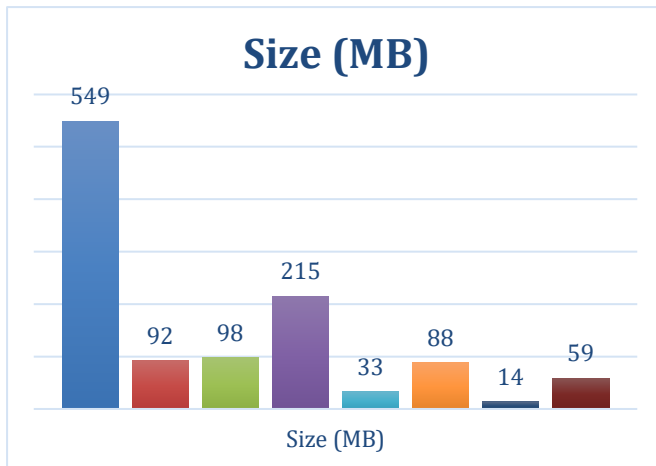
Table1

Year	Architecture	Key Features	Developed By
2014	VGG19	Simple and uniform architecture with 19 weight layers.	Visual Geometry Group, Oxford
2015	InceptionV3	Inception modules with multiple kernel sizes, factorized convolutions, auxiliary classifiers.	Google
2015	ResNet50	Residual connections to address vanishing gradient problem, 50 layers.	Microsoft Research
2016	DenseNet121	Dense connectivity pattern promoting feature reuse, 121 layers.	Cornell, Tsinghua, Facebook AI
2016	InceptionResNetV2	Combines Inception modules with residual connections, efficient training and high performance.	Google
2017	Xception	Replaces Inception modules with depthwise separable convolutions, high efficiency.	Google
2018	MobileNetV2	Designed for mobile/embedded applications, inverted residuals, linear bottlenecks.	Google
2021	EfficientNetV2B3	Compound scaling, optimized for accuracy and training speed, advanced training techniques.	Google

Table2

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters (M)	Depth	Time(ms) /inference step (CPU)	Time(ms) /inference step (GPU)
VGG19	549	71.30%	90.00%	143.7	19	84.8	4.4
InceptionV3	92	77.90%	93.70%	23.9	189	42.2	6.9
ResNet50	98	74.90%	92.10%	25.6	107	58.2	4.6
InceptionResNetV2	215	80.30%	95.30%	55.9	449	130.2	10
DenseNet121	33	75.00%	92.30%	8.1	242	77.1	5.4
Xception	88	79.00%	94.50%	22.9	81	109.4	8.1
MobileNetV2	14	71.30%	90.10%	3.5	105	25.9	3.8
EfficientNetV2B3	59	82.00%	95.80%	14.5	-	-	-





4.1.11 Performance metric

Evaluating the performance of a Machine learning model is one of the important steps while building an effective ML model.

To evaluate the performance or quality of the model, different metrics are used, and these metrics are known as performance metrics or evaluation metrics.

These performance metrics help us understand how well our model has performed for the given data.

In this way, we can improve the model's performance by tuning the hyper-parameters.

Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows visualization of the performance of an algorithm.

Here's how it works:

True Positives (TP): These are the cases when the actual class is positive and the model correctly predicts it as positive.

True Negatives (TN): These are the cases when the actual class is negative and the model correctly predicts it as negative.

False Positives (FP): Also known as a "Type I error," these are the cases when the actual class is negative but the model incorrectly predicts it as positive.

False Negatives (FN): Also known as a "Type II error," these are the cases when the actual class is positive but the model incorrectly predicts it as negative.

A confusion matrix is usually represented in the following format:

Total population = P + N		Predicted condition	
		Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

- **Accuracy:** - It reflects the fraction of correctly classified instances out of the total instances in the dataset. Accuracy provides a simple and intuitive way to evaluate how well a CNN model is performing. It quantifies the model's ability to make correct predictions.

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

- **Selectivity:** - Selectivity evaluates the model's capacity to accurately recognize negative instances or non-target classes. A high level of Selectivity suggests that the model is adept at minimizing false alarms and accurately categorizing non-target instances.

$$Selectivity = TNR = \frac{TN}{N} = \frac{TN}{TN + FP}$$

- **Precision:** - Precision assumes particular significance when the objective is to minimize false positives, providing insights into the model's effectiveness in correctly identifying true positives while minimizing the occurrence of false positives.

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

- **Sensitivity:** - Sensitivity quantifies the model's capacity to accurately recognize positive instances among all the genuine positive instances present in the dataset. In the context of testing a CNN model, this metric gauge the model's proficiency in identifying objects or specific features of interest.

$$Sensitivity = TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

- **F1 Score:** - Also known as Dice's coefficient or Dice Similarity Coefficient, it combines two important metrics, precision, and recall, into a single value, providing a balanced measure of a model's accuracy. The F1 score is particularly useful when dealing with imbalanced datasets where one class significantly outnumbers the other.

$$F1 = \frac{2TP}{2TP + FP + FN}$$

- **Jaccard Index:** - The Jaccard Index, also known as the Jaccard similarity coefficient or IoU (Intersection over Union), it serves as a means to gauge how effectively a CNN model localizes objects within an image. It precisely measures the spatial agreement between the predicted region (such as a bounding box or segmented area) and the actual ground truth region, offering a quantifiable indication of the model's accuracy in identifying objects of interest.

$$JI = \frac{TP}{TP + FP + FN}$$

- **Negative Predictive Value:** - NPV serves as a pivotal measure as it gauges the model's capability to accurately recognize true negatives while avoiding false negatives. NPV specifically scrutinizes the model's efficacy concerning true negatives, which are instances where the model correctly predicts a negative outcome.

$$NPV = \frac{TN}{TN + FN}$$

- **False Negative Rate:** - The False Negative Rate quantifies the proportion of actual positive instances (based on ground truth) that the model erroneously categorizes as negative (resulting in false negatives). Essentially, it quantifies the rate at which the model overlooks or fails to identify pertinent objects or features in the dataset.

$$FNR = \frac{FN}{P} = \frac{FN}{FN + TP}$$

- **False Discovery Rate:** - It helps us to ensure that the conclusions drawn from multiple tests in CNN model testing are more robust and reliable, reducing the likelihood of drawing erroneous or overly optimistic conclusions about the effectiveness of a particular model or configuration.

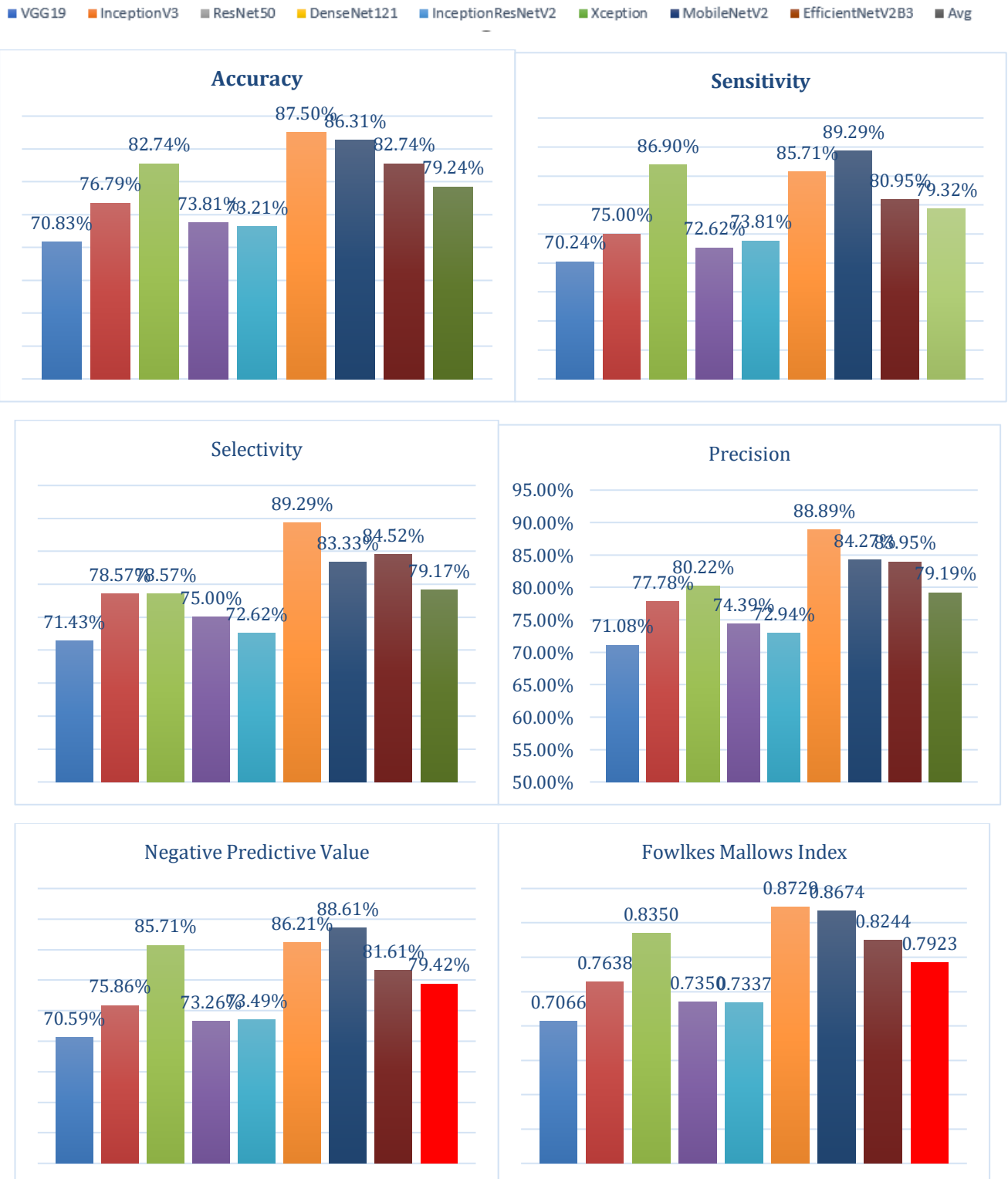
$$FDR = \frac{FP}{FP + TP}$$

- **Fowlkes Mallows Index (FMI):**- The FMI is a measure of the similarity between two clusters, calculated as the geometric mean of precision and recall. It balances the trade-off between precision and recall, providing a single metric that reflects the clustering quality. High FMI indicates that the clustering results are both precise and comprehensive.
- **Miss Rate, False Negative Rate (FNR):**- The miss rate measures the proportion of actual positives that are incorrectly identified as negatives (FN/(FN+TP)). It is the complement of sensitivity and indicates the rate at which the model fails to identify true positives. Low miss rate is desirable, meaning fewer positive instances are missed by the test.
- **Fall out, False Positive Rate (FPR):**- The fall-out measures the proportion of actual negatives that are incorrectly identified as positives (FP/(FP+TN)). It is the complement of specificity and indicates the rate of false alarms. Low fall-out is desirable, meaning fewer false positives occur.
- **False Omission Rate (FOR):**- FOR measures the proportion of negative test results that are false negatives (FN/(FN+TN)). It indicates the likelihood of a negative result being incorrect, important in contexts where missing a positive case is costly. Low FOR is desirable, meaning most negative predictions are true negatives.
- **Positive Likelihood Ratio (PLR):**- PLR is the ratio of the probability of a positive test result in people with the condition to the probability in people without the condition (Sensitivity / (1 - Specificity)). It indicates how much more likely a positive result is to occur in someone with the condition compared to someone without. High PLR means a positive test result is a strong indicator of the condition.
- **Negative Likelihood Ratio (NLR):**- NLR is the ratio of the probability of a negative test result in people with the condition to the probability in people without the condition ((1 - Sensitivity) / Specificity). It indicates how much less likely a negative result is to occur in someone with the condition compared to someone without. Low NLR means a negative test result is a strong indicator of not having the condition.
- **Diagnostic Odds Ratio (DOR):**- DOR is the ratio of the odds of the test being positive if the subject has the condition to the odds if the subject does not have the condition (PLR / NLR). It combines sensitivity and specificity into a single metric, useful for overall test performance. High DOR indicates that the test is effective at distinguishing between those with and without the condition.
- **Threat Score, Critical Success Index (CSI):**- CSI measures the ratio of true positives to the sum of true positives, false negatives, and false positives (TP / (TP + FN + FP)). It assesses the accuracy of positive predictions, especially in scenarios where the prevalence of the condition is low. High CSI indicates that the test accurately identifies positives while minimizing false positives and false negatives.
- **Cohen's Kappa Coefficient (κ):**- Cohen's Kappa measures inter-rater agreement for categorical items, accounting for agreement occurring by chance ((Po - Pe) / (1 - Pe)). It provides a robust measure of agreement beyond mere percentage agreement. High κ indicates strong agreement between raters beyond chance levels.
- **Jaccard Index:**- The Jaccard Index measures similarity between two sets, defined as the size of the intersection divided by the size of the union (TP / (TP + FP + FN)). It is used in clustering and

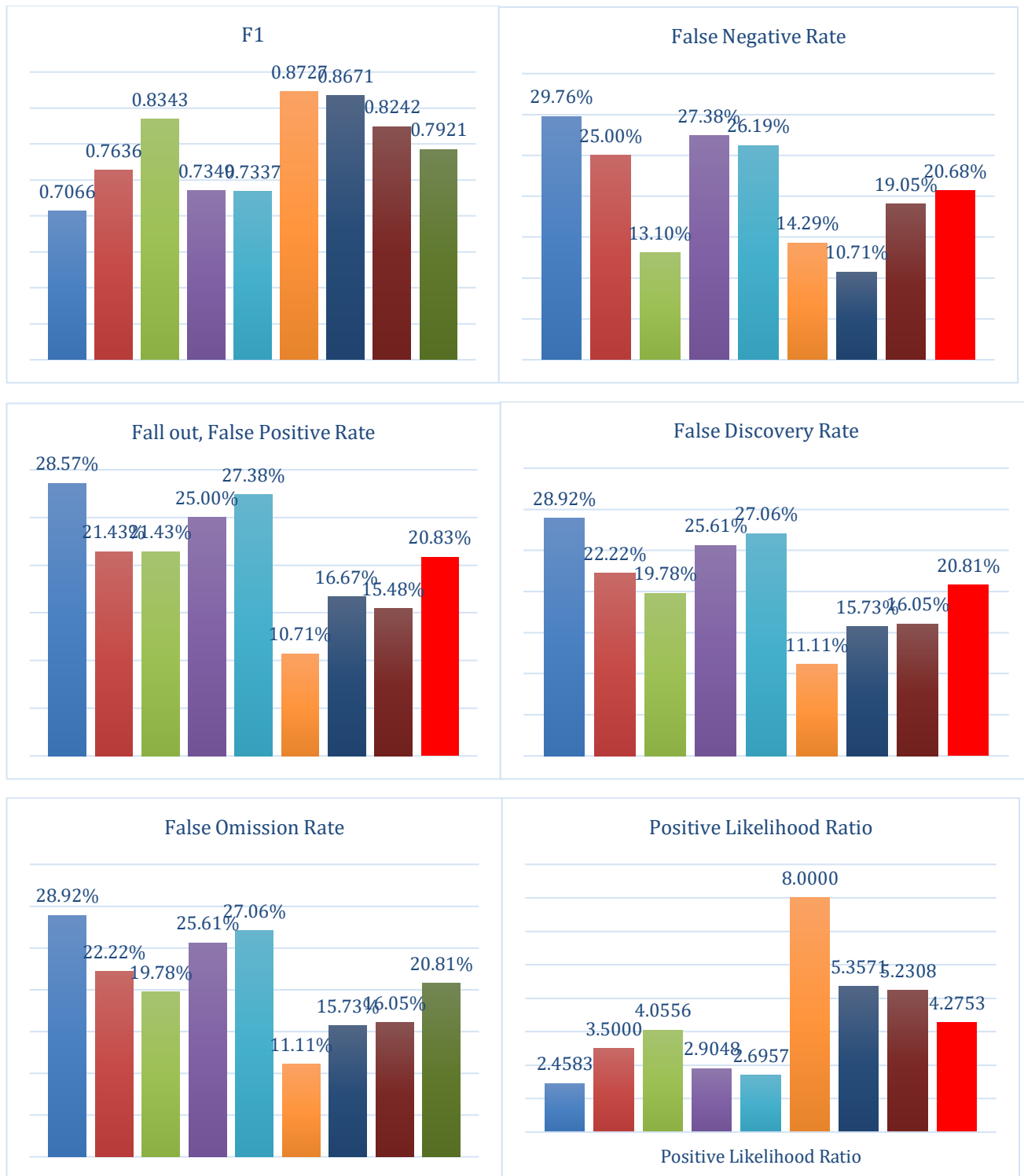
classification to compare the similarity of sample sets. High Jaccard Index indicates high similarity between the sets being compared.

- **Phi coefficient (ϕ), Matthews Correlation Coefficient (MCC):-** MCC is a measure of the quality of binary classifications, considering true and false positives and negatives (($TPTN - FPFN$) / $\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}$). It is particularly useful for imbalanced datasets. High MCC indicates a strong correlation between observed and predicted classifications, providing a balanced measure of test performance.

4.1.12 Result of 1st phase

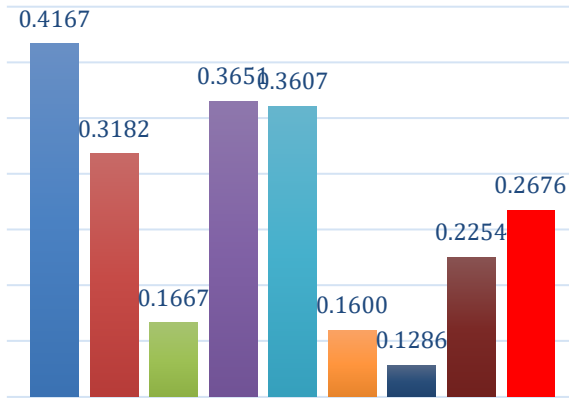


Skin Disease Detection System

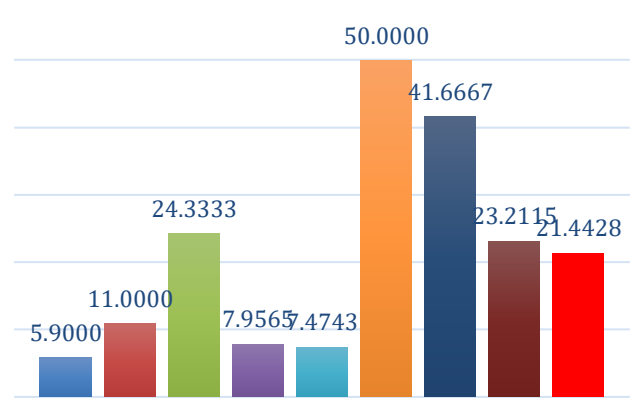


Skin Disease Detection System

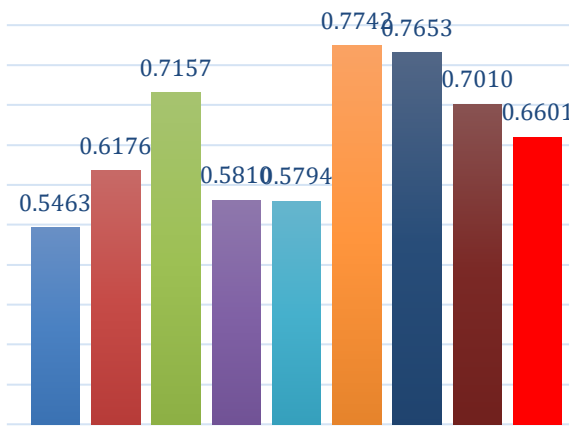
Negative Likelihood Ratio



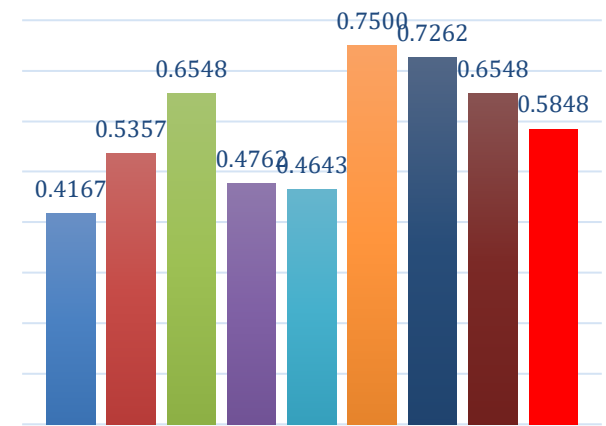
Diagnostic Odds Ratio



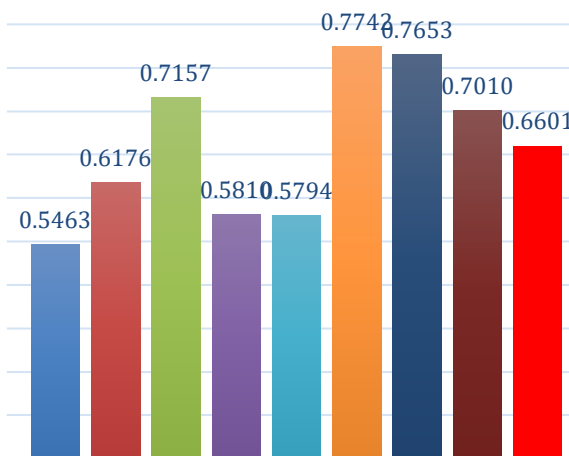
Threat Score, Critical Success Index



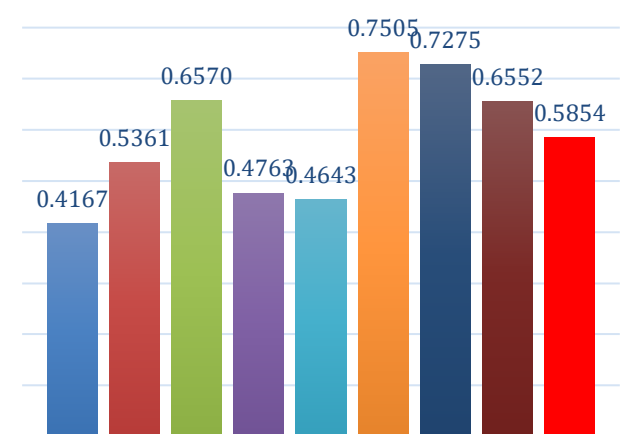
Cohen's Kappa Coefficient (κ)



Jaccard Index



Phi coefficient (ϕ), Matthews Correlation Coefficient



4.2 Phase – 2, Ensemble Learning

Ensemble learning refers to the process of combining multiple learning algorithms to obtain a better predictive performance than could be obtained from any of the constituent learning algorithms alone. The fundamental principle behind ensemble learning is that a group of weak learners can come together to form a strong learner. These weak learners are strategically combined to produce a single predictive model. The primary objective of ensemble learning is to improve the overall performance, stability, and robustness of the predictive model. By aggregating the outputs of multiple models, ensemble methods aim to achieve better generalization on unseen data compared to individual models.

4.2.1 Reasons to employ Ensemble Learning

Ensemble learning offers several compelling advantages that make it a valuable technique in machine learning. Here are the key reasons to use ensemble learning, summarized in points:

Improved Accuracy: Ensemble methods typically achieve higher predictive accuracy than individual models by combining their outputs. Aggregating the results of multiple models reduces the likelihood of errors that any single model might make.

Reduced Overfitting: By combining multiple models, ensemble methods can help mitigate the risk of overfitting to the training data. Techniques like bagging reduce variance, which can prevent models from becoming too tailored to the specificities of the training dataset.

Increased Robustness: Ensemble models are generally more robust to noisy data and outliers, as the impact of anomalies is averaged out across multiple models. This robustness makes ensemble models more reliable in real-world applications.

Model Stability: Single models can be highly sensitive to variations in the training data, leading to instability. Ensemble methods smooth out these variations, resulting in more stable predictions.

Versatility: Ensemble methods can be applied to a wide range of machine learning problems, including classification, regression, and clustering. They are flexible and can be used with various types of base models, from decision trees to neural networks.

Bias-Variance Trade-off: Ensemble learning helps balance the bias-variance trade-off by combining models with different biases and variances. Methods like boosting reduce bias by focusing on the mistakes of previous models, while bagging reduces variance by averaging the predictions of multiple models.

Handling Complex Data: Complex datasets with intricate patterns and interactions can often be better modelled by combining multiple simpler models. Ensembles can capture a wider array of data patterns than a single model might.

Improved Generalization: By leveraging multiple models, ensemble methods enhance the generalization capabilities of the predictive model, leading to better performance on unseen data.

Utilization of Diverse Models: Ensemble methods allow the integration of diverse models, including those with different strengths and weaknesses, to create a more comprehensive predictive system. Combining models that approach the problem differently can cover more aspects of the data, leading to better overall performance.

Adaptability: Ensemble methods like boosting adaptively adjust to the training data by focusing more on hard-to-predict instances. This adaptability helps in refining the model iteratively, leading to improved predictive performance.

Enhanced Predictive Power: Ensemble methods can often outperform even the best individual models, providing a means to achieve superior results without needing highly complex individual models. This is particularly useful when the individual models are limited in their capacity but can collectively provide strong predictions.

Error Reduction: By averaging the predictions of multiple models, ensemble methods reduce the impact of random errors. This error reduction contributes to more accurate and reliable predictions.

4.2.2 Mathematical Foundations of Ensemble Learning

Ensemble learning is underpinned by several mathematical principles that explain its effectiveness in improving model performance. Here, we delve into the key mathematical concepts that form the basis of ensemble methods:

Law of Large Numbers

The Law of Large Numbers states that as the number of trials or observations increases, the average of the results obtained from these trials will converge to the expected value. In the context of ensemble learning, this law implies that by averaging the predictions of multiple models, the overall prediction will converge to a more accurate and stable result.

$$\hat{y}_{ensemble} = \frac{1}{n} \sum_{i=1}^n \hat{y}_i$$

where \hat{y}_i is the prediction of the i -th model and n is the total number of models in the ensemble.

Bias-Variance Tradeoff

The bias-variance tradeoff is a fundamental concept in machine learning that describes the tradeoff between two sources of error that affect model performance:

- **Bias:** The error due to overly simplistic models that do not capture the underlying patterns in the data.
- **Variance:** The error due to models that are too complex and sensitive to fluctuations in the training data.

Ensemble methods, particularly bagging and boosting, help balance this tradeoff by combining models that might individually have high bias or high variance.

- **Bagging:** Reduces variance by averaging the predictions of different models trained on different subsets of the data.
- **Boosting:** Reduces bias by focusing on the errors of previous models and correcting them in subsequent models.

Bias-Variance Decomposition: ***Expected Error = Bias² + Variance + Irreducible Error***

Error Reduction

Ensemble learning can reduce different types of errors in a predictive model, such as bias, variance, and noise. The principle of error reduction through averaging is particularly crucial in ensemble methods.

- **Combined Error Reduction:** For an ensemble of n models, the variance of the ensemble's prediction can be reduced by a factor of n if the errors of the individual models are uncorrelated.

$$\sigma_{ensemble}^2 = \frac{1}{n^2} \sum_{i=1}^n \sigma_i^2 = \frac{\sigma^2}{n}$$

where σ_i^2 is the variance of the i -th model's prediction, and σ^2 is the variance of each individual model (assuming they are similar).

Majority Voting and Weighted Averaging

Ensemble methods often rely on majority voting (for classification) or weighted averaging (for regression) to combine predictions from multiple models.

- **Majority Voting:** Each model in the ensemble votes for a class, and the class with the majority of votes is chosen as the final prediction.

$$\hat{y}_{ensemble} = \text{mode}(\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n)$$

- **Weighted Averaging:** Each model's prediction is weighted by its performance, and the final prediction is the weighted average of individual predictions.

$$\hat{y}_{ensemble} = \sum_{i=1}^n w_i \hat{y}_i$$

where w_i is the weight assigned to the i -th model.

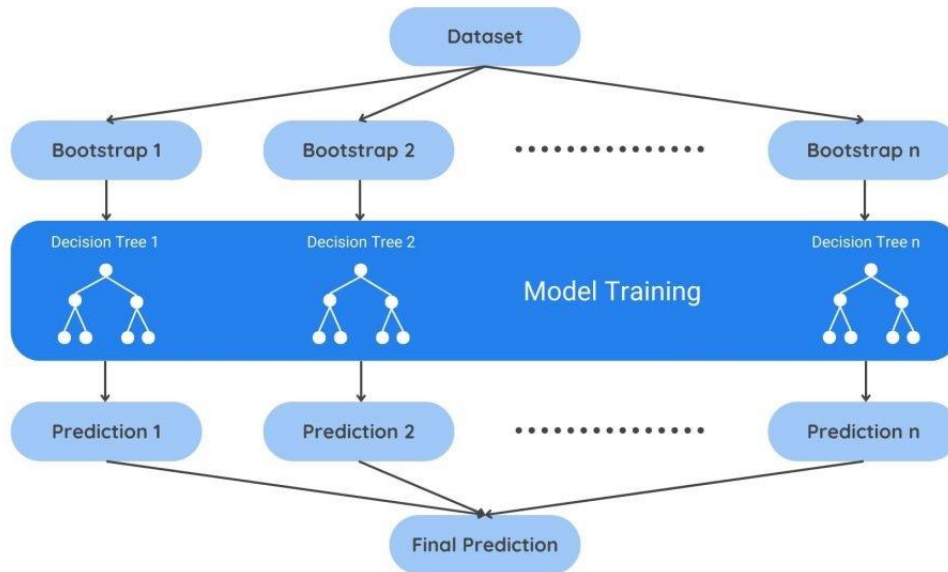
4.2.3 Types of Ensemble Learning Methods

Ensemble methods can be broadly classified into several categories, each with unique mechanisms and advantages.

1. Bagging (Bootstrap Aggregating)

Bagging, or Bootstrap Aggregating, aims to reduce variance and avoid overfitting. The key idea is to train multiple instances of the same model on different subsets of the training data, which are generated through bootstrapping (random sampling with replacement). The final prediction is obtained by averaging the predictions (for regression) or taking a majority vote (for classification).

Skin Disease Detection System

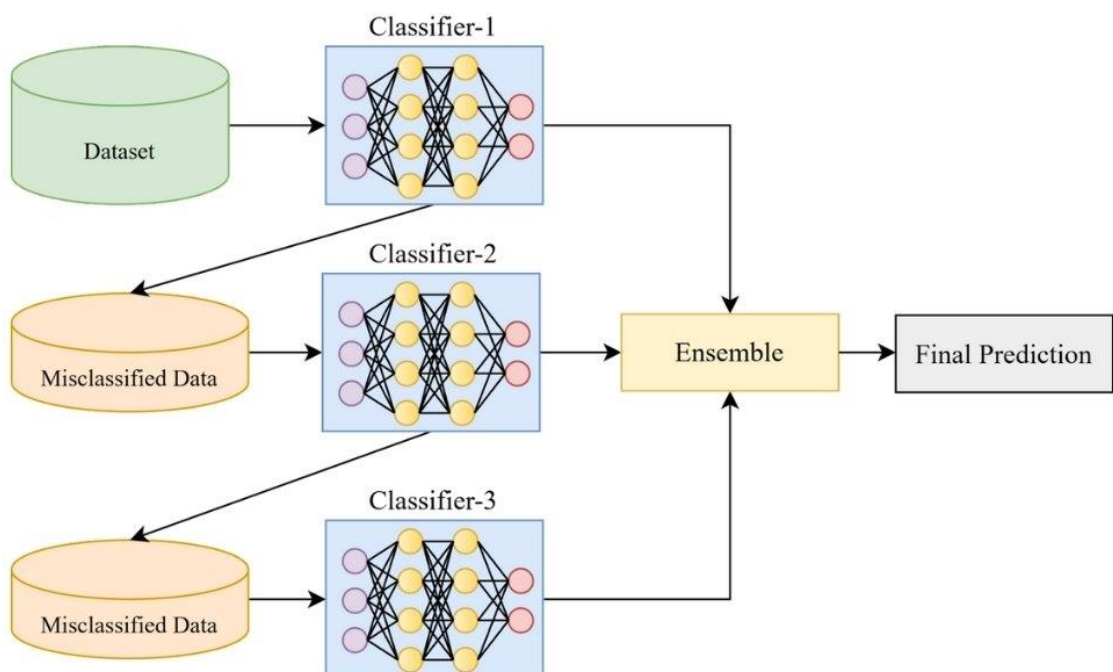


Examples:

- **Random Forest:** Random Forest is a popular bagging technique where multiple decision trees are trained on different bootstrapped subsets of the data. The final output is based on the majority vote of the individual trees. Random Forest reduces variance and improves generalization.

2. Boosting

Boosting focuses on converting weak learners into strong learners by sequentially training models. Each subsequent model attempts to correct the errors of the previous models, with a focus on the hardest cases. Boosting assigns higher weights to misclassified instances, making them more influential in subsequent models.

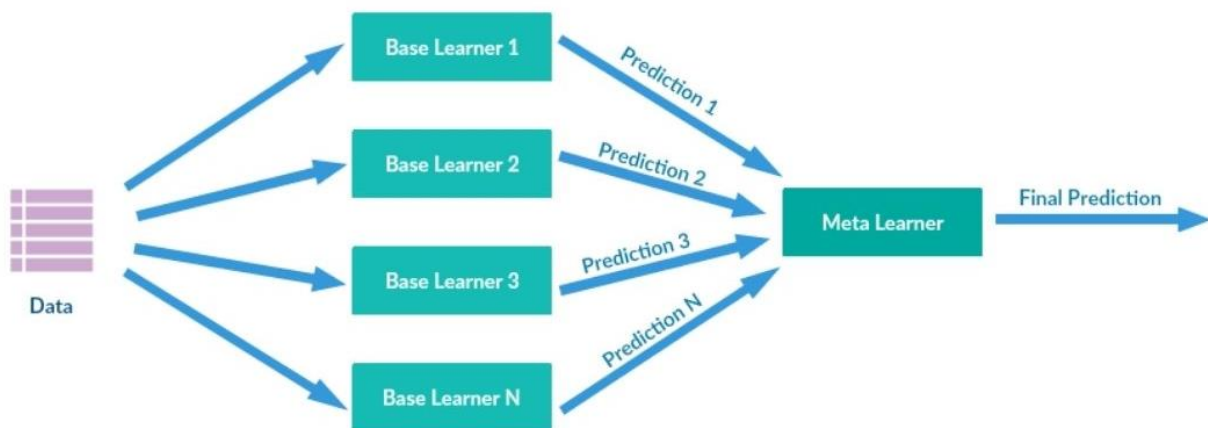


Examples:

- **AdaBoost (Adaptive Boosting):** Adjusts the weights of incorrectly classified instances and combines weak classifiers with a weighted majority vote.

3. Stacking (Stacked Generalization):

Stacking involves training multiple base models and then using a meta-model to combine their predictions. The meta-model is trained on the predictions of the base models, aiming to learn the best way to combine these predictions.



Examples:

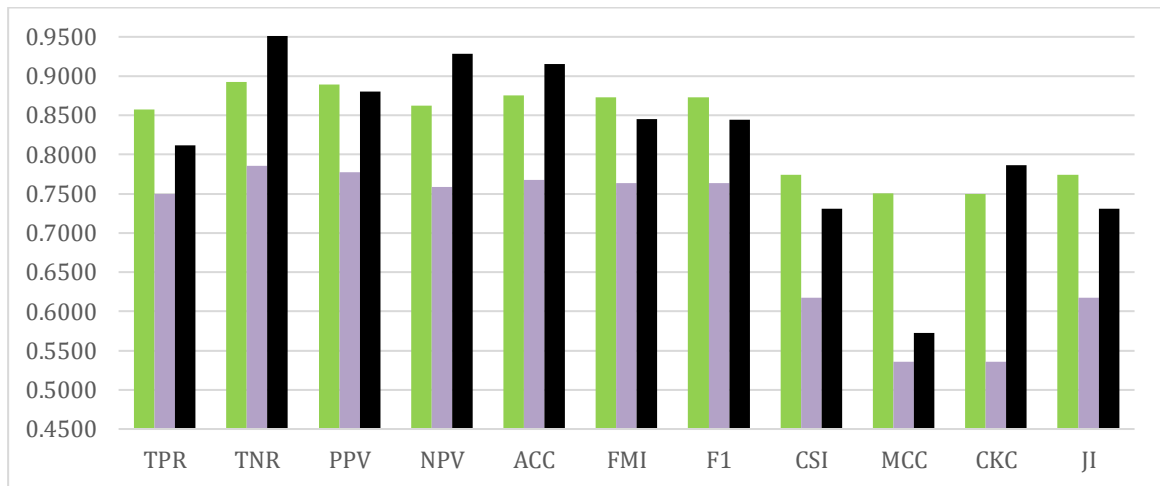
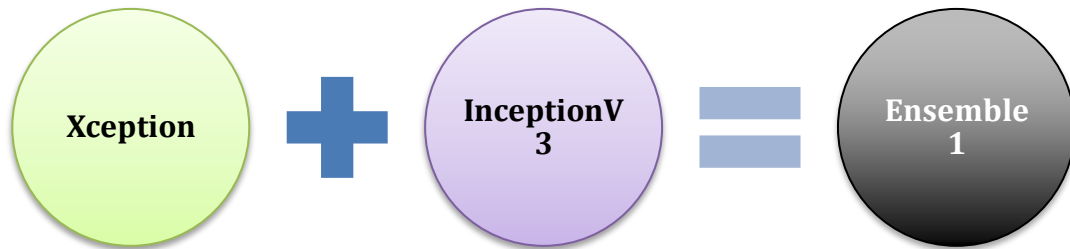
- **Super Learner:** Uses cross-validation to determine the optimal combination of base model predictions.

4. Other Ensemble Methods: -

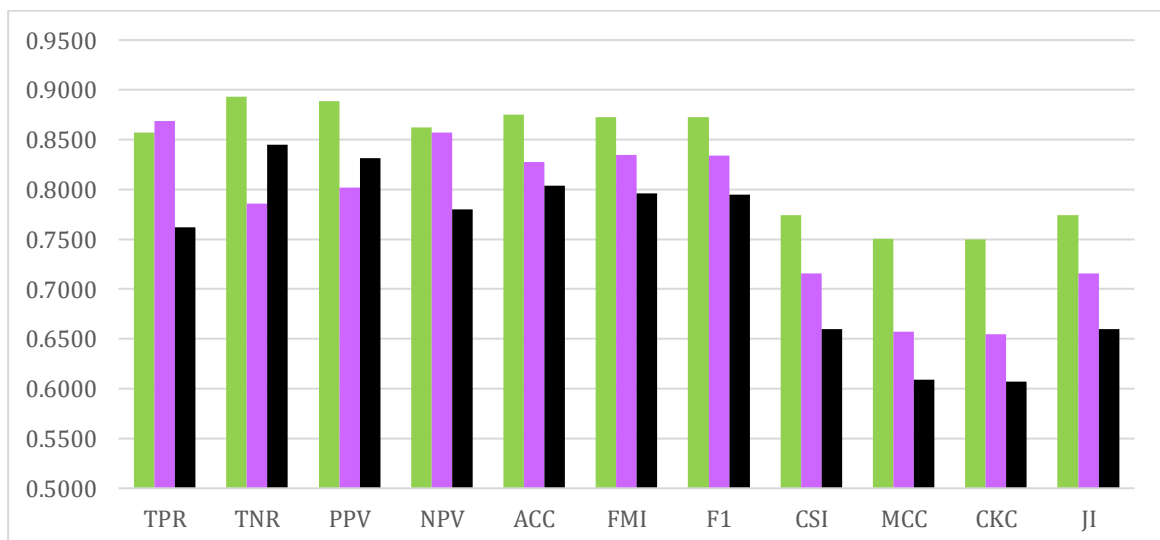
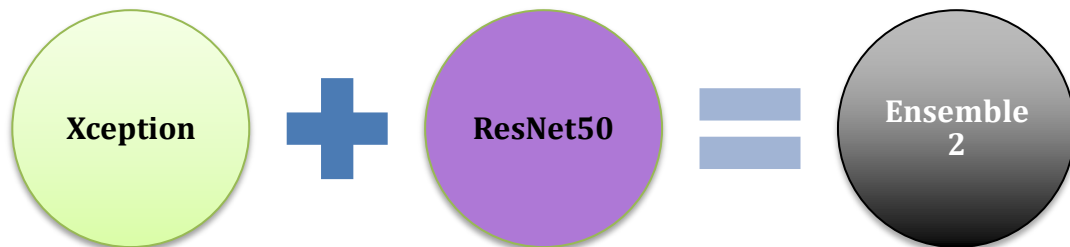
- **Voting:** Combines the predictions of multiple models by averaging (for regression) or majority voting (for classification). There are two types of voting: hard voting (majority rule) and soft voting (weighted average of probabilities).
- **Cascading:** Trains models sequentially where each model in the sequence uses the output of the previous model as input. This method is often used in complex pipelines where different stages of models contribute to the final prediction.

We choose Stacking Ensemble Learning Methods,

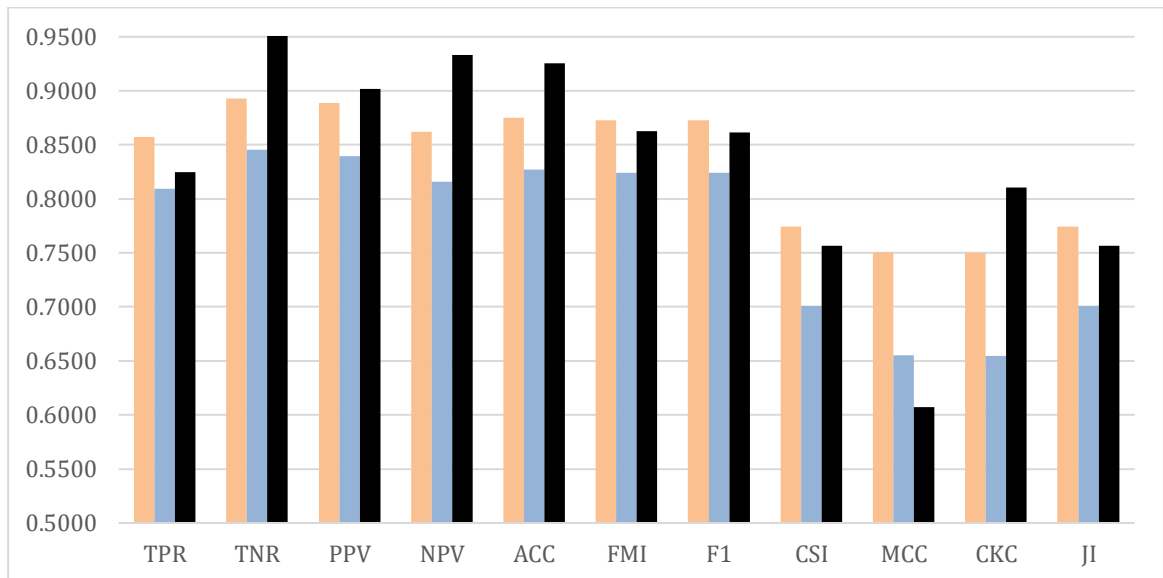
4.2.4 Ensemble 1:



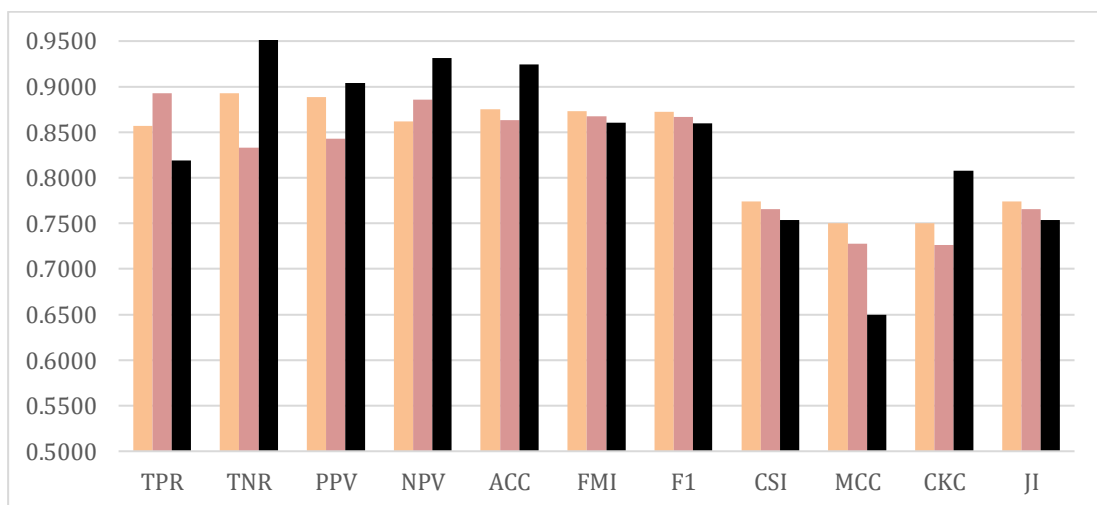
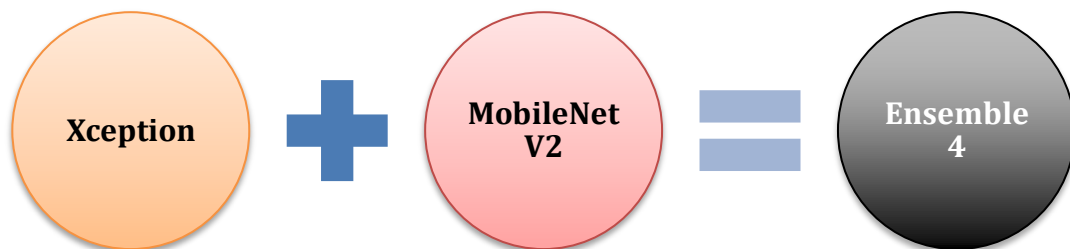
4.2.5 Ensemble 2:



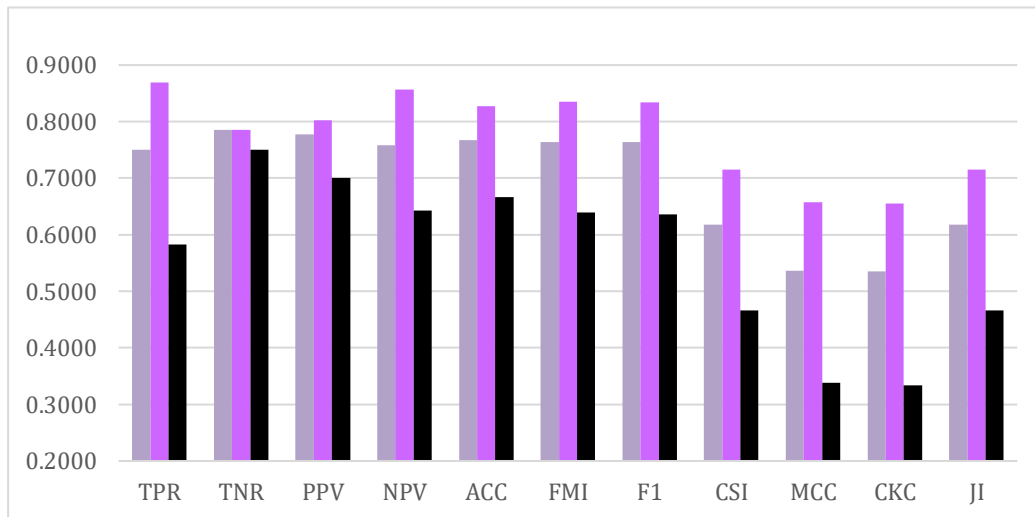
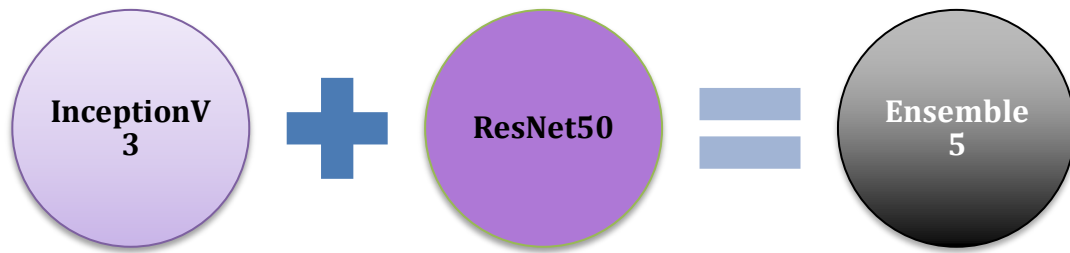
4.2.6 Ensemble 3:



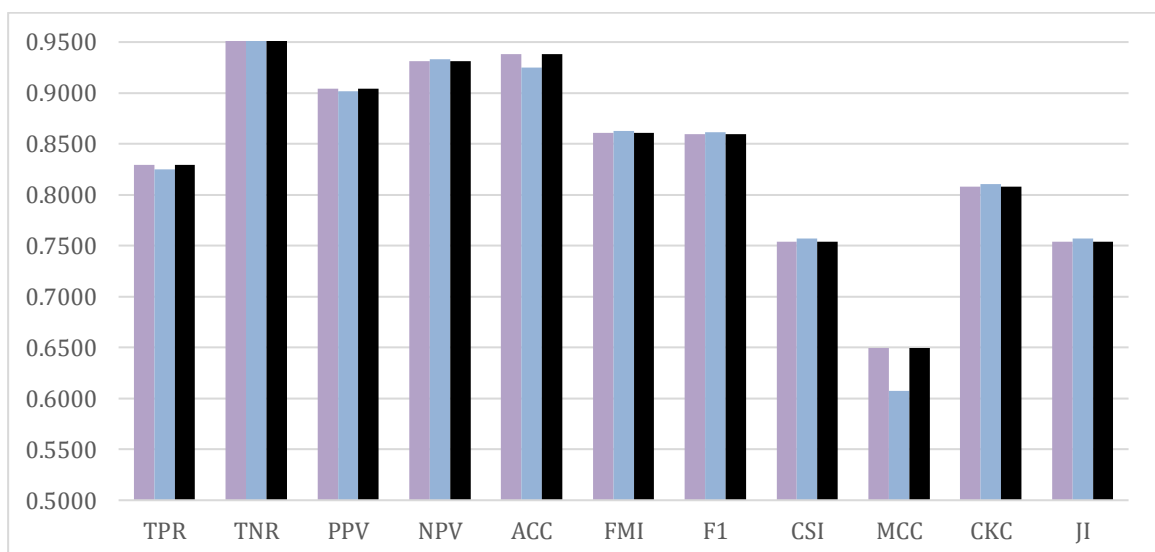
4.2.7 Ensemble 4:



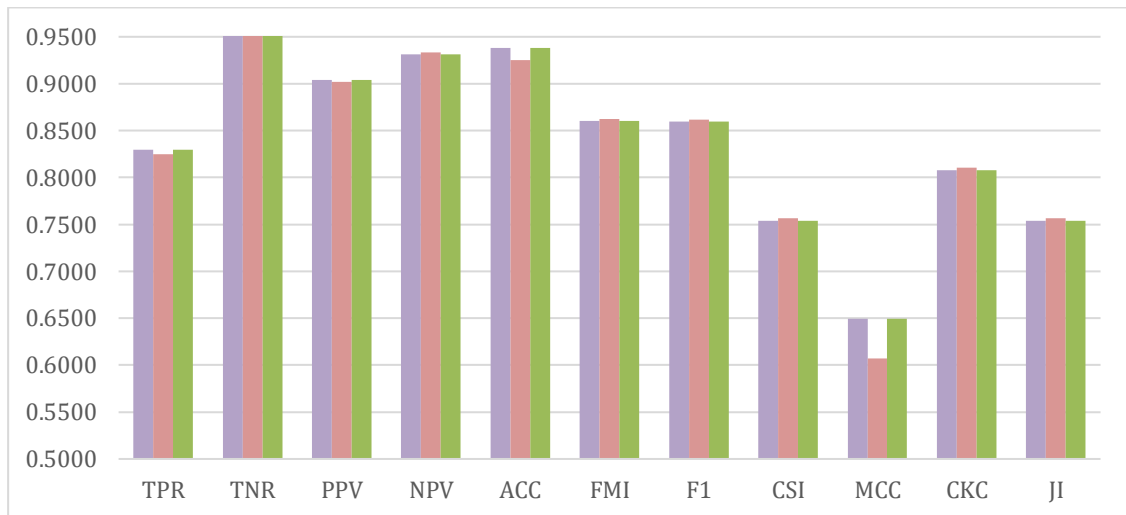
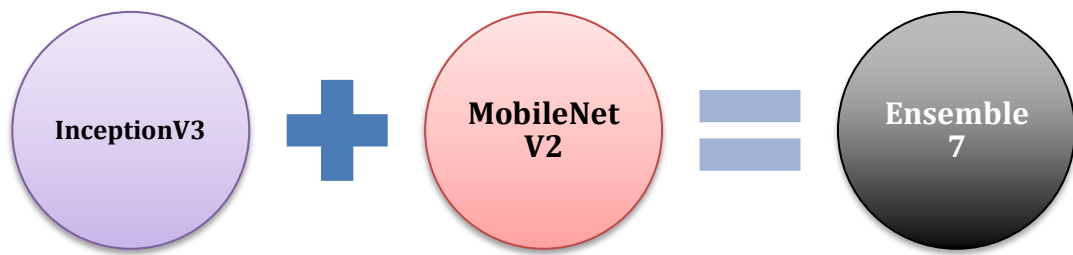
4.2.8 Ensemble 5:



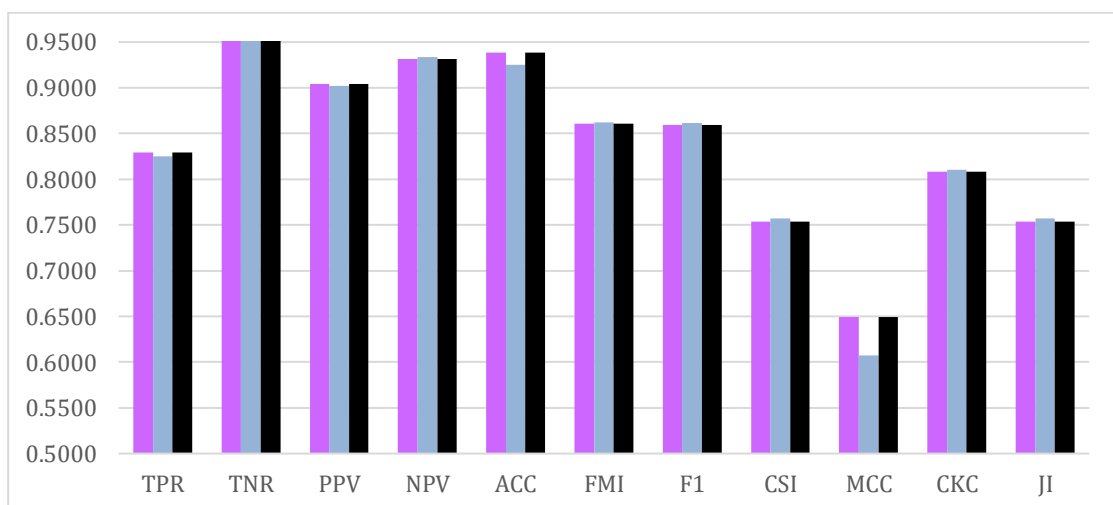
4.2.9 Ensemble 6:



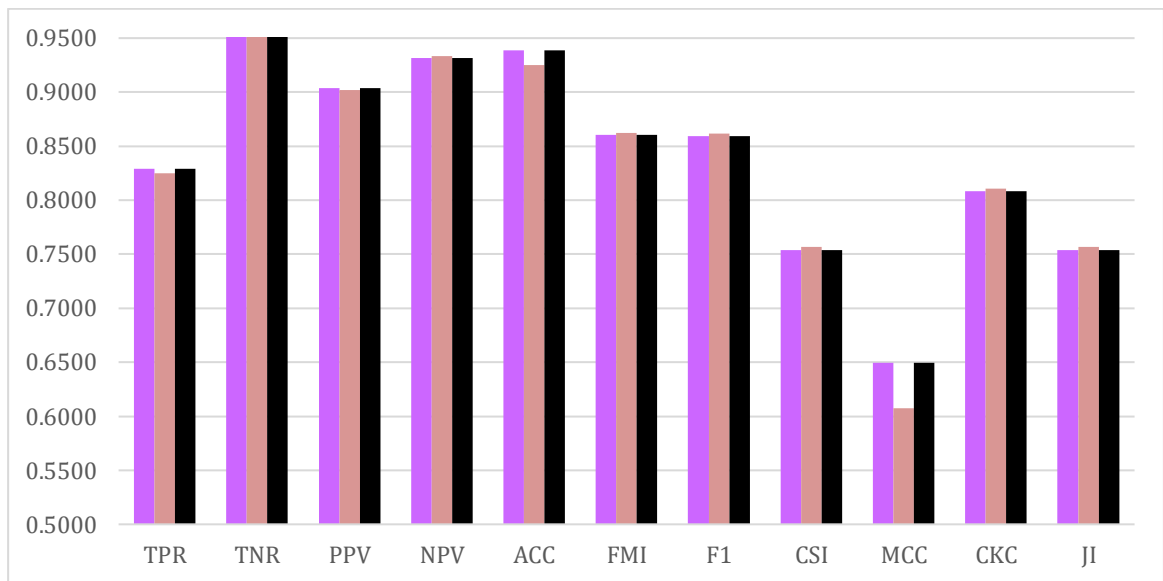
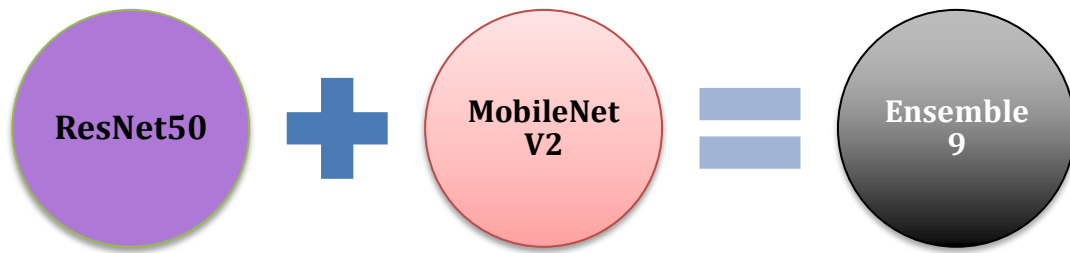
4.2.10 Ensemble 7:



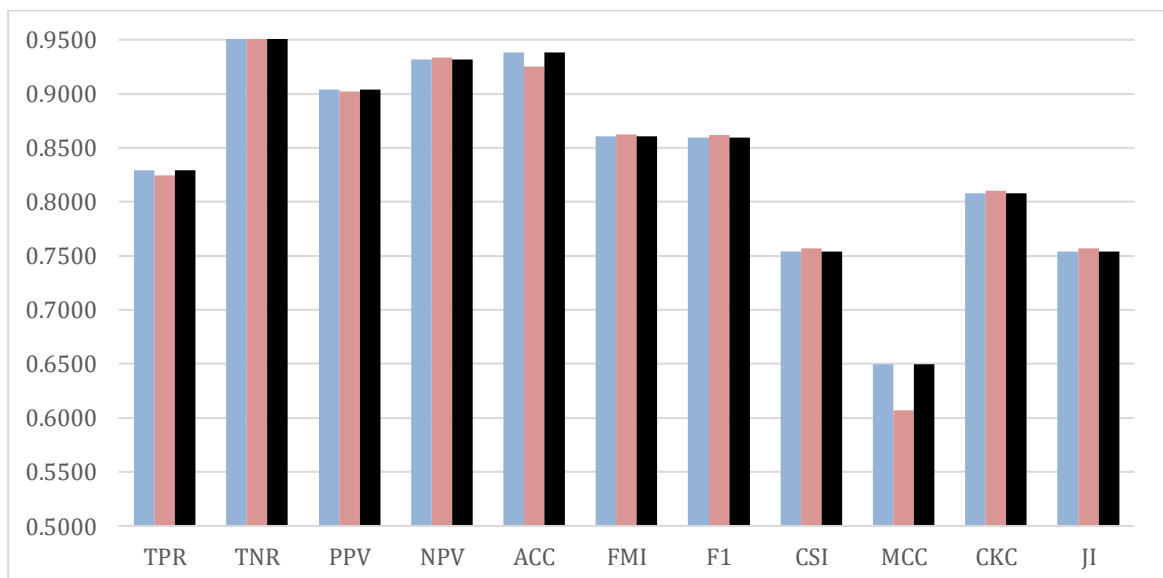
4.2.11 Ensemble 8:



4.2.12 Ensemble 9:



4.2.13 Ensemble 10:



4.3 Phase-3, Optimization Technique

The optimization technique used in the third phase involved a weighted sum approach. This method assigns different weights to the performance metrics of each model based on their relative importance. By calculating a weighted average of these metrics, we were able to comprehensively evaluate the models and identify the one that offers the best overall performance. This approach ensures that the selected model not only excels in accuracy but also maintains a balanced performance across other crucial evaluation parameters, thereby optimizing the system's effectiveness in detecting skin diseases.

Weighted sum optimization is a method used in multi-objective optimization where multiple objectives are combined into a single objective function by assigning weights to each objective. This approach allows for a straightforward optimization process using standard optimization techniques.

Here's a step-by-step outline of how to perform weighted sum optimization:

1. **Define Objectives:** Identify the multiple objectives $f_1(x), f_2(x), \dots, f_n(x)$ that need to be optimized.
2. **Assign Weights:** Assign a weight w_i to each objective $f_i(x)$. These weights should be non-negative and typically sum up to 1. The weights represent the relative importance of each objective.
3. **Formulate the Weighted Sum Objective Function:**

$$F(x) = w_1 f_1(x) + w_2 f_2(x) + \dots + w_n f_n(x)$$

This creates a single scalar objective function $F(x)$ from the multiple objectives.

4. **Optimize the Weighted Sum Objective:** Use an appropriate optimization algorithm to find the solution x^* that minimizes (or maximizes, depending on the problem) $F(x)$.
5. **Analyze the Results:** Evaluate the optimal solution to ensure that it is satisfactory for all objectives. Adjust the weights and re-optimize if necessary to explore different trade-offs between objectives.

Example

Consider a problem with two objectives: minimizing cost ($f_1(x)$) and maximizing quality ($f_2(x)$). Let's assign weights $w_1 = 0.6$ for cost and $w_2 = 0.4$ for quality.

1. **Define Objectives:**
 - $f_1(x)$: Cost
 - $f_2(x)$: Quality (note that if higher quality is better, you might need to minimize $-f_2(x)$ instead).
2. **Assign Weights:**
 - $w_1 = 0.6$
 - $w_2 = 0.4$
3. **Formulate the Weighted Sum Objective Function:**

$$F(x) = 0.6 \cdot f_1(x) + 0.4 \cdot f_2(x)$$

(Assuming $f_2(x)f_2(x)f_2(x)$ is to be maximized, and we've adjusted it appropriately if necessary.)

4. Optimize the Weighted Sum Objective:

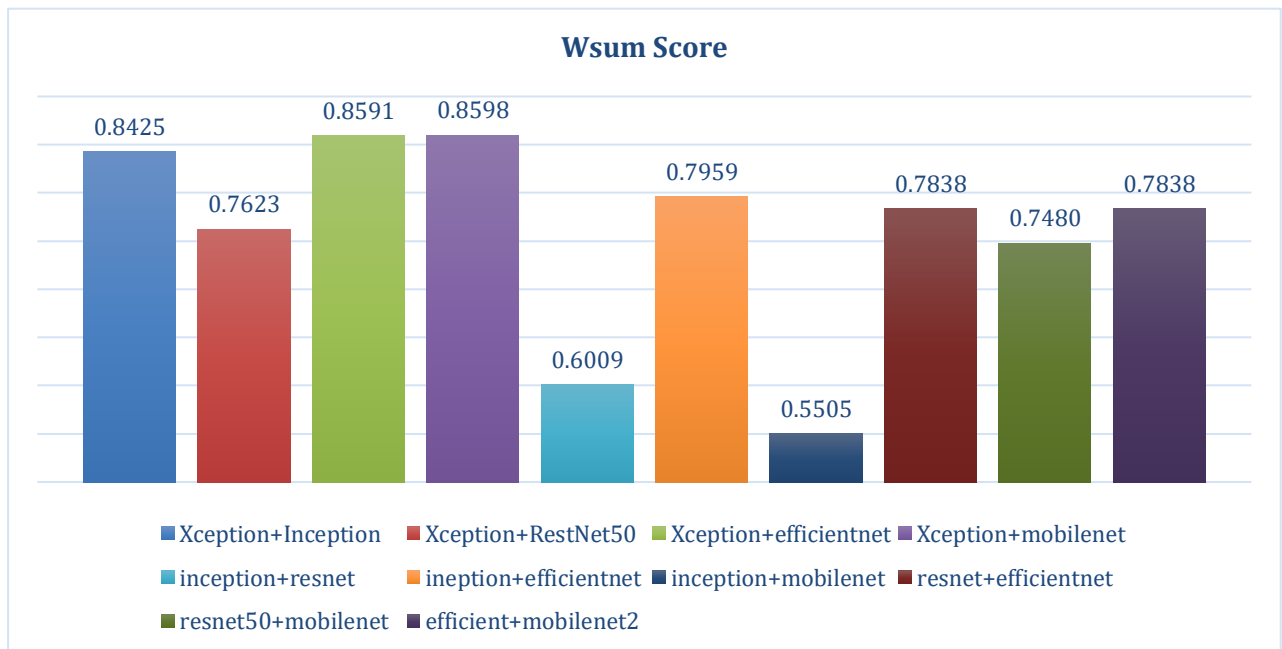
Use an optimization algorithm such as gradient descent, genetic algorithms, or any suitable method to find x^* that minimizes $F(x)$.

5. Analyze the Results:

Check the values of $f_1(x^*)$ and $f_2(x^*)$. If the solution is not satisfactory, adjust w_1 and w_2 and repeat the process.

Weighted sum optimization is a powerful and intuitive method for handling multi-objective optimization problems, allowing for a flexible trade-off between competing objectives.

4.3.1 Our result of Weighted sum technique

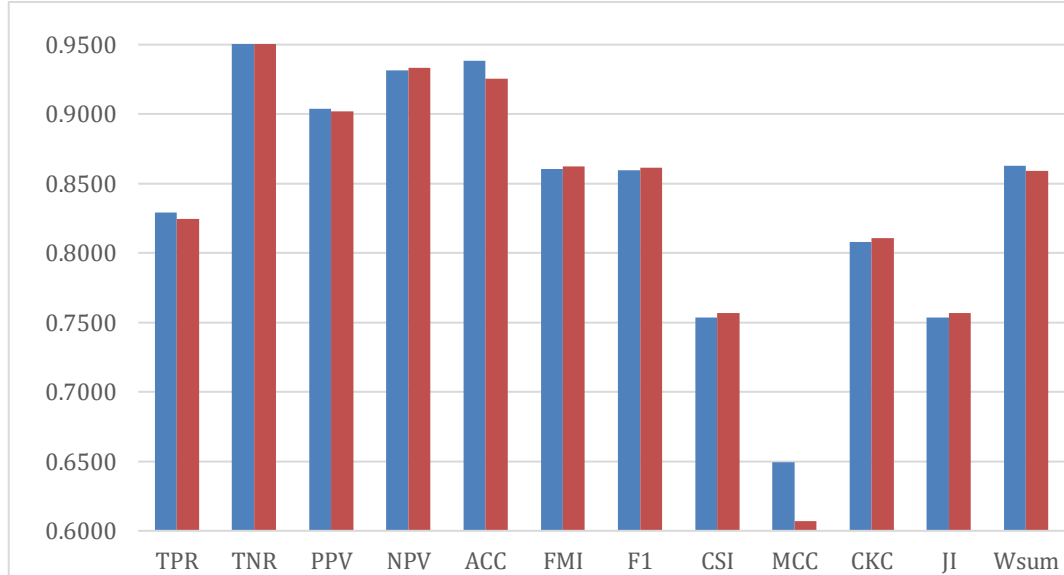


4.1.1 Comparison of best of two models

Features	Xception+mobilenet	Xception+efficientnet
TPR	0.8293	0.8247
TNR	0.9658	0.9648
PPV	0.9040	0.9019
NPV	0.9315	0.9334
ACC	0.9384	0.9253
FMI	0.8606	0.8625
F1	0.8596	0.8616
FNR	0.1707	0.1753
FPR	0.0342	0.0352
FDR	0.0960	0.0981
FOR	0.0685	0.0666
CSI	0.7537	0.7568

Skin Disease Detection System

MCC	0.6495	0.6072
CKC	0.8080	0.8105
JI	0.7537	0.7568
Wsum	0.8628	0.8591



4.1.1 Final Selection: Ensemble of Xception and MobileNetv2 models

The optimization process revealed that the combination of Xception and MobileNetv2 models yielded the best results. By leveraging the strengths of both models, the ensemble achieved superior accuracy and overall performance in detecting skin diseases. This optimized approach ensures a robust and reliable diagnostic tool, capable of providing precise and consistent results.

Chapter – 5

5. Front-end design

The integration of machine learning and Django offers a powerful tool for improving the diagnosis and management of psoriasis. By providing an automated, accurate, and scalable solution, our system aims to support healthcare providers in making more informed decisions and offer patients an accessible means to monitor their skin health.

5.1 Django: -

Django is a high-level Python web framework designed to help developers build robust, scalable, and maintainable web applications quickly and with minimal code. It follows the "batteries-included" philosophy, providing a wide range of built-in features that facilitate common web development tasks. Here are some key aspects of Django:

MVC Framework: Django follows the Model-View-Controller (MVC) architectural pattern, though it often uses the term Model-View-Template (MVT):

- **Model:** Defines the data structure, usually mapped to a database table.
- **View:** Handles the logic and control flow, often responding to user inputs and querying the model to return a response.
- **Template:** Renders the HTML and presentation layer for the end user.

ORM (Object-Relational Mapping): Django includes an ORM that allows developers to interact with the database using Python code instead of writing raw SQL queries. This promotes the use of clean, readable, and maintainable code.

Admin Interface: One of Django's standout features is its automatically generated admin interface. Once you define your models, Django can generate a fully functional admin panel for managing your application data.

Authentication: Django provides a robust authentication system out of the box, including user registration, login, logout, password management, and permissions.

URL Routing: Django includes a powerful URL dispatcher that maps URL patterns to views, allowing for clean and readable URL structures.

Templating Engine: Django comes with its own templating engine, which helps in separating the HTML presentation from the Python code logic.

Form Handling: Django simplifies form handling with its forms library, which supports form creation, validation, and processing.

Security: Django addresses common security issues like SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and clickjacking, providing built-in protections to make applications secure by default.

Scalability: Django is designed to handle high traffic and can be scaled efficiently to meet the demands of large-scale applications.

Django is particularly suitable for developers who prefer convention over configuration and appreciate having many tools readily available for common web development tasks. It's widely used in various industries, including content management, social media, scientific computing, and e-commerce.

5.2 Working of Django: -

Django works by providing a structured framework that helps developers build web applications efficiently. Here's a breakdown of how Django operates, highlighting its main components and their interactions:

I. Project and App Structure

- **Project:** A Django project is a collection of settings for an instance of Django, including database configuration, Django-specific options, and application-specific settings.
- **App:** A project contains multiple applications (apps), each of which is a web application that does something, such as a blog, a forum, or a customer management system.

II. URL Routing

- Django uses a URL dispatcher to map URL patterns to views.
- **urls.py:** Each app can have a urls.py file where URL patterns are defined. The main urls.py file in the project routes the top-level URLs to the appropriate app-specific URLs.

III. Views

- Views handle the logic of the application.
- A view is a Python function or class that receives a web request and returns a web response.
- **views.py:** This file contains the view functions or classes. A view can retrieve data from the database, process it, and render a template.

1. Models

- Models define the structure of the application's data and are typically mapped to database tables.
- **models.py:** This file contains model classes, which are Python classes that inherit from `django.db.models.Model`. Each attribute of the class corresponds to a database field.

2. ORM (Object-Relational Mapping)

- Django's ORM allows interaction with the database using Python code.
- Queries are written using the model classes, which Django translates into SQL.

3. Templates

- Templates are responsible for rendering HTML content.
- **templates/:** Templates are stored in this directory within each app. They use the Django template language to embed dynamic content in HTML.

4. Forms

- Forms handle user input, validation, and processing.
- **forms.py:** This file contains form classes, which define the fields and validation logic.

5. Admin Interface

- Django provides an automatic admin interface for managing application data.
- **admin.py:** This file registers models to be included in the admin interface, allowing them to be managed through a web interface without additional code.

6. Settings and Configuration

- **settings.py:** This file contains configuration settings for the Django project, such as database configurations, installed apps, middleware, templates, static files, and more.

Django's structure and built-in components streamline web application development, making it a powerful framework for building modern web applications.

5.3 Advantages of Using Django: -

Django offers numerous advantages for building web applications, making it a popular choice among developers. Here are some key benefits:

1. Rapid Development

- **Batteries-Included Framework:** Django comes with many built-in features such as authentication, URL routing, ORM, and an admin interface, reducing the need for third-party libraries and accelerating the development process.
- **Reusable Code:** Django encourages reusable code with its modular structure, allowing developers to use pre-built modules for common tasks.

2. Scalability

- **Modular Architecture:** Django's decoupled components allow applications to scale easily. You can scale different parts of your application independently.
- **Handling Traffic:** Django can handle high traffic loads and is used by many high-traffic websites, such as Instagram and Disqus.

3. Security

- **Built-in Protections:** Django includes built-in protections against many common security threats, including SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and clickjacking.
- **Strong Authentication:** Django provides a robust authentication system, managing user accounts, passwords, and permissions securely.

4. Versatility

- **Broad Application Range:** Django is suitable for a wide range of applications, from simple websites to complex data-driven platforms.
- **Third-Party Packages:** A rich ecosystem of third-party packages allows you to extend Django's capabilities to meet specific needs.

5. ORM (Object-Relational Mapping)

- **Database Interaction:** Django's ORM allows developers to interact with the database using Python code rather than SQL, which improves productivity and code readability.
- **Database Migrations:** Django's migration system keeps the database schema in sync with your models, simplifying schema changes over time.

6. Admin Interface

- **Automatic Admin Panel:** Django provides an automatically generated admin interface for managing application data, reducing the need for custom admin tools.

7. Community and Support

- **Active Community:** Django has a large and active community of developers who contribute to its continuous improvement and provide support through forums, mailing lists, and third-party packages.
- **Extensive Documentation:** Django's documentation is comprehensive and well-maintained, making it easier for developers to learn and use the framework effectively.

8. Template Engine

- **Separation of Concerns:** Django's templating engine encourages the separation of business logic and presentation, promoting cleaner and more maintainable code.
- **Reusable Templates:** Templates can be reused across different parts of the application, reducing redundancy.

9. Internationalization

- **Built-In Support:** Django includes built-in support for translating text into different languages, making it easier to develop applications for a global audience.

10. Customizable

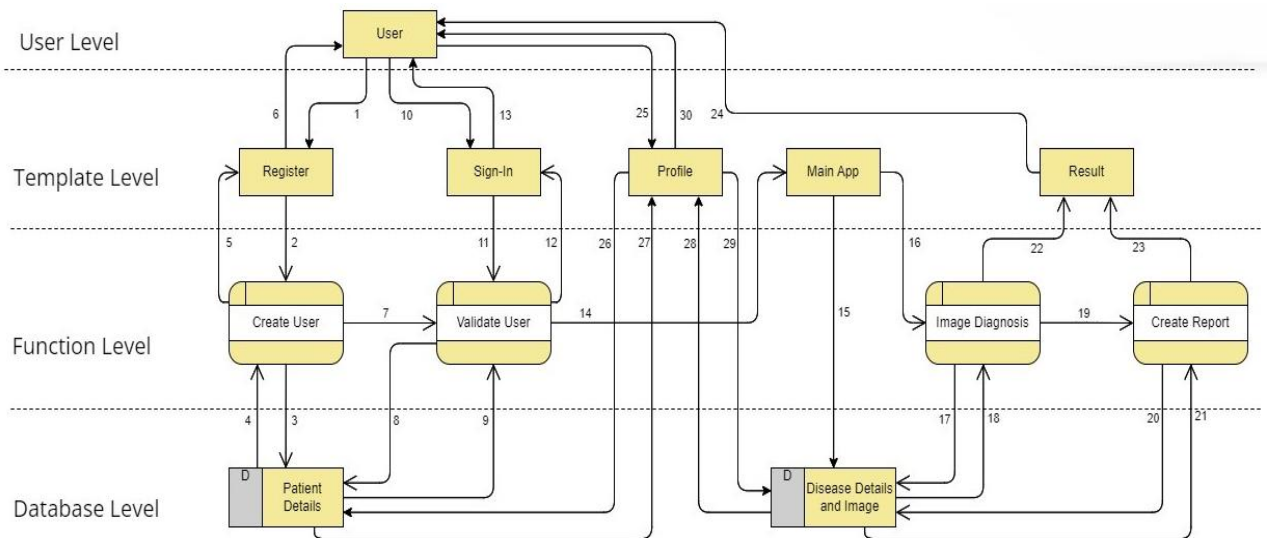
- **Flexible Framework:** While Django provides many built-in features, it is also highly customizable, allowing developers to override defaults and integrate custom solutions as needed.
- **Middleware:** Django allows for custom middleware components to process requests and responses, enabling customization of request handling.

11. Performance Optimization

- **Caching Framework:** Django includes a caching framework that can significantly improve application performance by reducing database access and rendering times.
- **Asynchronous Support:** Recent versions of Django support asynchronous views and middleware, which can improve the performance of I/O-bound applications.

Django is a comprehensive and versatile web framework that offers a range of features to facilitate rapid development, scalability, security, and maintainability. Its built-in tools and strong community support make it a robust choice for both small and large-scale web applications.

5.4 Data-Flow Diagram for Psoriasis Detection System: -



5.5 Steps in the DFD are explained hereby:

1. The 'User' opts to create a new account. For this, they will have to access the 'Register' entity.
2. The 'Register' entity will then call the function 'Create User' after it has taken all the required data from the 'User'.
3. The 'Create User' function will check with the database if the details are correct. It will also check whether any user with the same credentials already exists in the system. If everything is correct, then the details will be stored in the 'Patient Details' database and the new account will be created.
4. If there is already another account with the same credentials, an error message will be returned from the database.
5. This error message is relayed back to the 'Register' entity.
6. The error message reaches the 'User' from the 'Register' entity.
7. After successful creation of the new account, the 'Create User' function will call the 'Validate User' function, which will then automatically log the 'User' into the system.
8. The 'Validate User' function verifies the data received by it from the 'Patient Details' database.
9. The database then confirms to the function whether the credentials passed by it are valid or not.
10. The 'User' also has the option to directly login to his previously created account. For this, he/she uses the 'Sign-In' entity.
11. The 'Sign-In' entity will take the login credentials from the 'User' and then relay it to the 'Validate User' function. Then steps '8' and '9' will take place.
12. If the 'User' cannot be validated, an error message will be relayed from the 'Validate User' function to the 'Sign-In' entity.
13. The 'User' gets the error message from the 'Sign-In' entity.

14. If the 'User' is validated successfully, they will be granted access to the 'Main-App' entity, where they can upload all the information about their ailment.
15. The data that is given by the 'User' is then passed from the 'Main-App' to the 'Disease_Details_And_Image' database, and then stored there for further processing.
16. The 'Image Diagnosis' function is then called upon by the 'Main-App'
17. The 'Image Diagnosis' function then accesses the 'Disease_Details_And_Image' database for the 'Image' that was uploaded by the 'User'.
18. The image is then returned to the 'Image Diagnosis' function to then use it for predicting whether the skin issue is 'Psoriasis' or not.
19. After the diagnosis is done by 'Image Diagnosis' function, it then calls the 'Create Report' function.
20. The 'Create Report' function then accesses the 'Disease_Details_And_Image' database to get the information about the patient and the disease details.
21. The information returned by the database is then used by the 'Create Report' function to create a report which can be downloaded by the 'User' for future references.
22. The diagnosis returned by the 'Image-Diagnosis' function is returned to the 'Result' entity.
23. The report returned by the 'Create-Report' function is returned to the 'Result' entity.
24. The 'User' can view the diagnosis and the report from the 'Result entity'.
25. The 'User' can view his account details, medical history and previous reports via the 'Profile' entity.
26. The 'Profile' entity then inquires the 'Patient Details' database for information about the patient.
27. The database returns the inquired data back to the 'Profile' entity.
28. The 'Profile' entity then inquires the 'Disease_Details_And_Image' database for information about the patient's medical history.
29. The database returns the inquired data back to the 'Profile' entity.
30. The 'User' can view all the details from the 'Profile' entity.

Our current skin disease detection system focuses primarily on accurately identifying psoriasis. For future work, we aim to significantly expand the system's capabilities to detect a variety of skin diseases. This enhancement will involve classifying conditions into three distinct categories: psoriasis, other skin diseases (non-psoriasis), and healthy skin.

To achieve this, we will undertake the task of collecting a comprehensive and diverse database that encompasses a wide spectrum of skin conditions. This enriched dataset will then be meticulously trained and tested using state-of-the-art machine learning models. Should the initial outcomes fall short of expectations, we will employ advanced ensemble methods to synergize the strengths of multiple models, thereby boosting the system's overall performance.

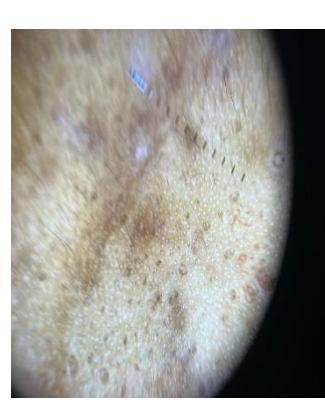
Additionally, we plan to incorporate multi-modal data, integrating patient history and symptom descriptions, to create a more holistic diagnostic tool. This approach aims to develop a more robust, reliable, and versatile detection system capable of accurately identifying and classifying multiple skin diseases, ultimately improving patient outcomes and advancing the field of dermatological diagnostics.

Chapter – 6

6. Final Testing

6.1. Sample Images

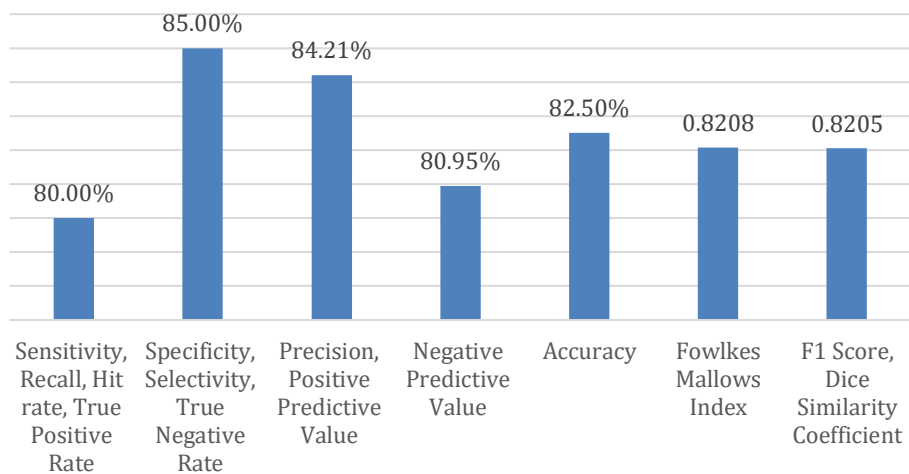
We test our system with some sample. Here are some examples of these



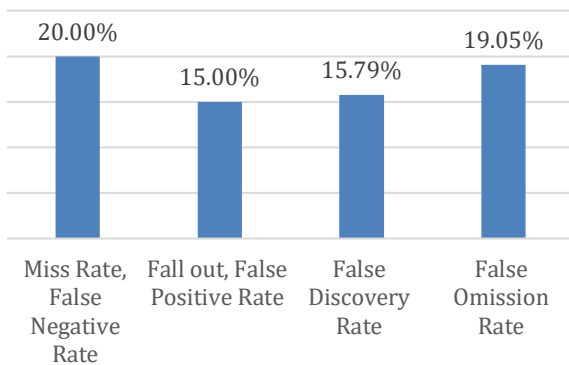
6.2. Final result

Training Set		
TARGET \ OUTPUT	Psoriasis	Non-Psoriasis
Psoriasis	16 40.00%	3 7.50%
Non-Psoriasis	4 10.00%	17 42.50%

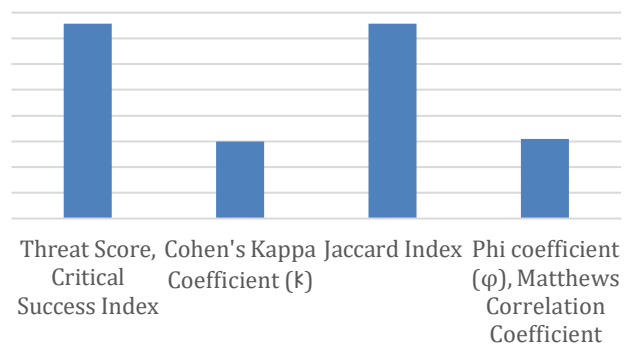
Final Result:1



Final Result:2



Final Result:3



Chapter – 7

7.1 Future Scope

Our current skin disease detection system focuses primarily on accurately identifying psoriasis. For future work, we aim to significantly expand the system's capabilities to detect a variety of skin diseases. This enhancement will involve classifying conditions into three distinct categories: psoriasis, other skin diseases (non-psoriasis), and healthy skin.

To achieve this, we will undertake the task of collecting a comprehensive and diverse database that encompasses a wide spectrum of skin conditions. This enriched dataset will then be meticulously trained and tested using state-of-the-art machine learning models. Should the initial outcomes fall short of expectations, we will employ advanced ensemble methods to synergize the strengths of multiple models, thereby boosting the system's overall performance.

Additionally, we plan to incorporate multi-modal data, integrating patient history and symptom descriptions, to create a more holistic diagnostic tool. This approach aims to develop a more robust, reliable, and versatile detection system capable of accurately identifying and classifying multiple skin diseases, ultimately improving patient outcomes and advancing the field of dermatological diagnostics.

7.2 Conclusion

There are several challenges associated with skin disease detection in India. One of the biggest challenges is the lack of awareness and education about skin diseases among the general public, which can result in delayed diagnosis and treatment. Additionally, there is a shortage of dermatologists in India, particularly in rural areas, which can limit access to specialized care.

Another challenge is the variability in skin types and conditions across different regions of India, which can make it difficult to develop algorithms that are accurate for all populations. Finally, the quality of images used for analysis can be a challenge, particularly in areas where there is limited access to high-quality imaging equipment.

Despite these challenges, we are trying to improve the accessibility and accuracy of dermatological care in the country using our research work.

7.3 References:

- [1] Noel C. F. Codella, David Gutman, M. Emre Celebi, Brian Helba, Michael A. Marchetti, Stephen W. Dusza, Aadi Kallou, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, and Allan Halpern. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 168–172, 2018.
- [2] A. Kalaivani and S. Karpagavalli. Detection and classification of skin diseases with ensembles of deep learning networks in medical imaging. *International Journal of Health Sciences*, (I), 6 2022.
- [3] Rashika Mishra and Ovidiu Daescu. Deep learning for skin lesion segmentation. *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1189–1194, 2017.
- [4] Md. Khairul Islam, Md Shahin Ali, Md Mosa-hak Ali, Mst. Farija Haque, Abhilash Arjan Das, Md.

- Maruf Hossain, D S Duranta, and Md Afifur Rahman. Melanoma skin lesions classification using deep convolutional neural network with transfer learning. In *2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA)*, pages 48–53, 2021.
- [5] Analysis of dermoscopy images by using abcd rule for early detection of skin cancer. *Global Transitions Proceedings*, 2(1):1–7, 2021.
- [6] Giuseppe Argenziano, Susana Puig, Iris Zalaudek, Francesco Sera, Rosamaria Corona, Mercè Alsina, Filomena Barbato, Cristina Carrera, Gerardo Ferrara, Antonio Guilabert, Daniela Massi, Juan A. Moreno-Romero, Carlos Muñoz Santos, Gianluca Petrillo, Sonia Segura, H. Peter Soyer, Renato Zanchini, and Josep Malvehy. Dermoscopy improves accuracy of primary care physicians to triage lesions suggestive of skin cancer. *Journal of Clinical Oncology*, 24(12):1877–1882, 2006. PMID: 16622262.
- [7] M. E. Celebi, Quan Wen, Hitoshi Iyatomi, Kouhei Shimizu, Huiyu Zhou, and Gerald Schaefer. A state-of-the-art survey on lesion border detection in dermoscopy images. 2015.
- [8] Transfer learning using a multi-scale and multi-network ensemble for skin lesion classification. *Computer Methods and Programs in Biomedicine*, 193:105475, 2020.
- [9] Khalid M. Hosny, Mohamed A. Kassem, and Mohamed M. Foad. Skin cancer classification using deep learning and transfer learning. *2018 9th Cairo International Biomedical Engineering Conference (CIBEC)*, pages 90–93, 2018.
- [10] Abbas Qaisar, M. Emre Celebi, Irene Fondón Garcia, and Ahmad Waqar. Melanoma recognition framework based on expert definition of abcd for dermoscopic images. *Skin Research and Technology*, 19(1):e93–e102, 06 2012.
- [11] Reda Kasmi and Karim Mokrani. Classification of malignant melanoma and benign skin lesions: Implementation of automatic abcd rule. *IET Image Processing*, 10, 01 2016.
- [12] Abder-Rahman H. Ali, Jingpeng Li, and Guang Yang. Automating the abcd rule for melanoma detection: A survey. *IEEE Access*, 8:83333–83346, 2020.
- [13] Ihab Zaqout. Diagnosis of skin lesions based on dermoscopic images using image processing techniques. *Pattern Recognition - Selected Methods and Applications*, 2016.
- [14] Giuseppe Argenziano, Susana Puig, Iris Zalaudek, Francesco Sera, Rosamaria Corona, Merce Alsina, Filomena Barbato, Cristina Carrera, Gerardo Ferrara, Guilabert A, Daniela Massi, Juan Moreno-Romero, Carlos Muñoz-Santos, Gianluca Petrillo, Sonia Segura, Peter Soyer, Renato Zanchini, and Josep Malvehy. Dermoscopy improves accuracy of primary care physicians to triage lesions suggestive of skin cancer. *Journal of clinical oncology : official journal of the American Society of Clinical Oncology*, 24:1877–82, 05 2006.
- [15] Yali Nie, Paolo Sommella, Marco Carratù, Matteo Ferro, Mattias O’Nils, and Jan Lundgren. Recent advances in diagnosis of skin lesions using dermoscopic images based on deep learning. *IEEE Access*, 10:95716–95747, 2022.
- [16] Adria Romero Lopez, Xavier Giro-i Nieto, Jack Burdick, and Oge Marques. Skin lesion classification from dermoscopic images using deep learning techniques. In *2017 13th IASTED International Conference on Biomedical Engineering (BioMed)*, pages 49–54, 2017.
- [17] Pengcheng Jiang. Cnn-based diagnosis system on skin cancer using ensemble method weighted by cubic precision. In *2021 2nd International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, pages 145–152, 2021.

- [18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for largescale image recognition. arXiv preprint arXiv:1409.1556, 2014.
- [19] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.
- [20] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.
- [21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2014.
- [22] Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications, 2023.