Skin Disease Identification Using Deep Learning Techniques

[1]R. BHANU SANKAR ,[2]POTNURU MANIKANTA
[1]Assistant Professor, Department Of MCA, MCA Final Semester,
[1]Master of Computer Applications,
[1]Sanketika Vidya Parishad Engineering College, Vishakhapatnam, Andhra Pradesh, India.

Abstract:

Skin diseases are a common global health issue, especially in rural areas with limited access to dermatological care. Traditional diagnostic methods are often time-consuming and rely heavily on expert interpretation. This project proposes an automated skin disease detection system using Convolutional Neural Networks (CNNs) for accurate and efficient diagnosis. The system uses image processing techniques to analyze skin images and identify patterns associated with various diseases. A deep learning model is trained using labeled datasets and implemented with Keras. The final model is deployed through a Flask web application, offering a user-friendly interface for real-time predictions. Users can upload an image of the affected area and receive instant classification results. The system improves accessibility and reduces dependency on specialists for initial screening. It also enables early detection, which is crucial for effective treatment. The accuracy of the model enhances trust in automated systems. This approach supports the growing need for AI in healthcare, especially in under-resourced areas.

Index Terms: Skin Disease Detection, Convolutional Neural Networks (CNN), Deep Learning, Image Processing, Keras, Flask, Medical Image Classification, Artificial Intelligence in Healthcare, Automated Diagnosis, Dermatology, Web Application.

1.Introduction:

Skin diseases are among the most common health conditions worldwide, affecting millions of people across all age groups. They are particularly widespread in rural and underserved regions, where access to specialized dermatological care is limited. Early and accurate diagnosis of skin conditions is crucial to prevent complications, but conventional diagnosis often involves long waiting times and limited specialist availability. Moreover, visual examination by dermatologists can be subjective and prone to human error, especially in early or mild cases. These challenges highlight the need for automated and accurate diagnostic tools. Advances in artificial intelligence and image processing provide promising solutions to this problem.

Deep learning, particularly Convolutional Neural Networks (CNNs), has shown exceptional performance in the field of image classification and medical diagnostics. CNNs are capable of learning complex patterns and textures from medical images, making them ideal for identifying and classifying skin diseases. Unlike traditional machine learning, CNNs do not require manual feature extraction, which simplifies the pipeline and improves accuracy. This project utilizes CNNs built with the Keras deep learning library, enabling fast prototyping and training. By feeding the network thousands of labeled skin images, the model learns to differentiate between various skin conditions. This deep learning model forms the core of our diagnostic system.

To make this system accessible to end-users, the trained CNN model is integrated into a Flask-based web application. This application provides a simple interface where users can upload skin images and receive predictions instantly. The backend processes the image, passes it through the model, and returns the most likely disease class along with confidence scores. This approach not only speeds up the diagnosis but also makes early detection possible even in remote locations. By combining image processing, deep learning, and web development, this system aims to reduce the burden on healthcare facilities and improve the quality of dermatological care. It represents a step forward in the use of AI for public health and telemedicine.

1.1. Existing system

In many current systems, the diagnosis of skin diseases relies on traditional methods such as visual inspection by dermatologists and basic dermatoscopic imaging. These methods are often limited by the availability of specialists, particularly in rural and remote areas[1]. Manual diagnosis can be time-consuming and subjective, depending on the expertise of the doctor. Most existing automated systems use Feedforward Neural Networks (FFNN), which struggle with complex patterns and high-dimensional data. FFNNs typically use fixed features and do not handle image variability effectively[2]. They also lack the capability to reuse information, limiting their predictive accuracy. These limitations make traditional and older AI-based systems less effective for accurate skin disease classification.
Some systems attempt to use Hidden Layer Neural Networks (HLNN), which offer slight improvements by applying abstraction in processing. However, even HLNN-based models face issues with data generalization and scalability[5]. These networks are limited in their ability to handle non-linear data transformations and are prone to overfitting when exposed to large image datasets. Additionally, manual feature extraction is still required in many of these systems, increasing the complexity and reducing reliability. Lack of real-time diagnosis support is another drawback, as most systems are not integrated with accessible platforms for user interaction[7]. As a result, they fail to provide efficient and user-friendly diagnostic assistance. The need for more accurate, scalable, and accessible systems has led to the exploration of deep learning approaches like CNN.

1.1.1.Challenges

Data Availability and Quality

❖ High-quality, labeled datasets of diverse skin tones, lighting conditions, and disease types are limited. Inconsistent image quality and unbalanced classes affect model performance.

Class Imbalance

❖ Some skin diseases have many image samples, while others have very few, causing the model to be biased toward frequently occurring classes.

Variability in Skin Conditions

❖ The same disease can appear differently across individuals due to skin color, stage of infection, and lighting, making classification more complex.

Overfitting

❖ With small or limited datasets, the model may perform well during training but poorly on unseen data, reducing generalizability.

Model Interpretability

❖ Deep learning models, especially CNNs, function as “black boxes,” making it difficult for users and doctors to understand why a particular prediction was made.

Real-Time Performance

❖ Ensuring fast and accurate predictions in real-time, especially when deploying on low-resource environments or mobile devices, can be challenging.

1.2 Proposed system:

The proposed system uses Convolutional Neural Networks (CNNs) to automatically identify and classify various skin diseases from image data. It replaces traditional manual diagnosis with a more accurate, faster, and scalable solution[9]. The model is trained on a dataset of labeled skin images using Keras, enabling it to learn complex visual patterns. Once trained, the model is deployed through a Flask web application, allowing users to upload images and receive instant predictions. The system is designed to be user-friendly and accessible, even in remote or rural areas. By automating the initial diagnosis process, it reduces the burden on healthcare professionals[11]. This approach improves early detection, helping users seek timely medical attention.

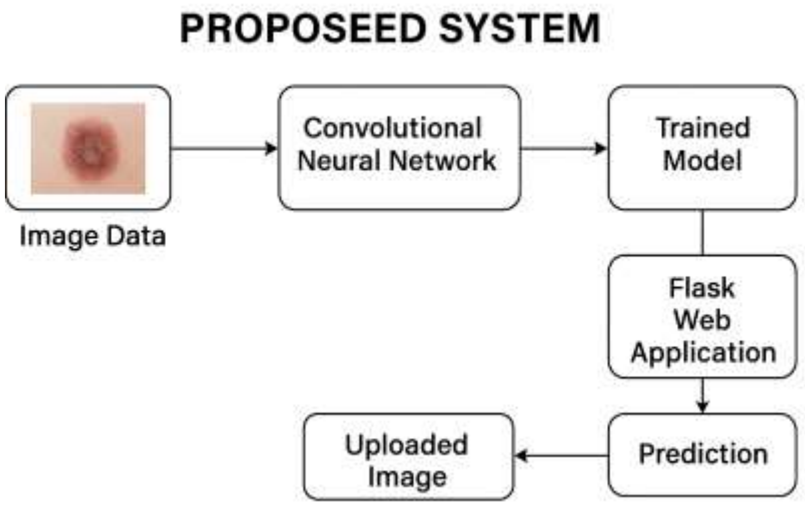


Fig: 1 Proposed Diagram

1.1.1 Advantages:

1. Fast and Automated Diagnosis

❖ The system provides instant results, reducing the time needed for initial screening compared to manual consultations.

2. High Accuracy

❖ CNNs can detect complex patterns in images, leading to better prediction accuracy than traditional models.

3. User-Friendly Interface

❖ The Flask-based web application makes it easy for users to upload images and receive results without technical knowledge.

4. Accessibility in Remote Areas

❖ The system can be accessed via internet, making it especially useful in rural or underserved regions with limited dermatology services.

5. Scalable and Adaptable

❖ New diseases or more data can be added over time, improving the model's performance continuously.

6. Cost-Effective

❖ Reduces the need for multiple in-person visits and costly consultations for basic diagnosis.

7. Supports Early Detection

❖ Early identification of skin diseases allows users to seek timely medical treatment, potentially preventing complications.

2.1 Architecture:

The architecture of the proposed system begins with a user interface built using Flask, where users can upload images of skin conditions. These images are saved temporarily and passed to a preprocessing module, which resizes and normalizes them to match the model's input format[13]. The preprocessed images are then fed into a Convolutional Neural Network (CNN) trained using Keras, which analyzes the visual patterns to classify the disease. The model outputs the predicted class along with a confidence score. The top predictions are displayed back to the user on the web interface. All predictions are handled in real time, ensuring fast responses. The system also includes error handling for invalid inputs or failed predictions. This modular and scalable architecture ensures high performance, ease of use, and efficient

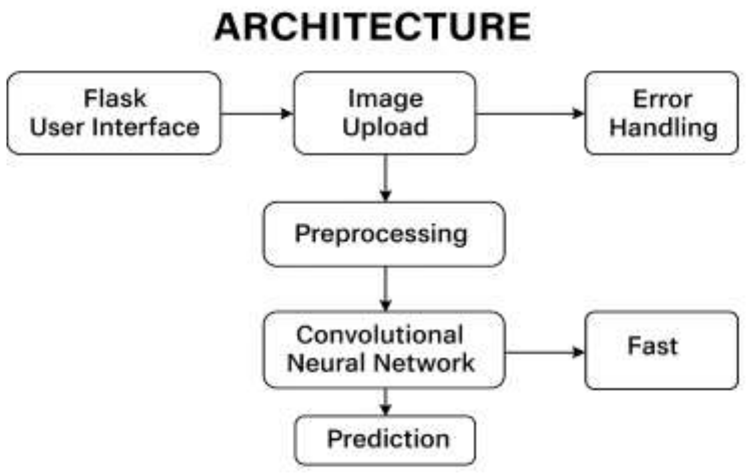


Fig:2 Architecture

2.2 Algorithm:

The algorithm begins when a user uploads an image of a skin condition through the Flask-based web interface. The system first verifies the file format to ensure it's an accepted type such as .jpg, .jpeg, or .png. Once validated, the image is saved to a temporary folder on the server. The preprocessing stage then resizes the image to the required input shape (e.g., 192x192), converts it to RGB, and normalizes pixel values between 0 and 1. The image is reshaped into a format suitable for model input using NumPy arrays[15]. This processed image is then passed into the trained CNN model built using Keras. The CNN model processes the image and returns a probability distribution across all known skin disease classes.

From the output probabilities, the class with the highest confidence score is selected as the predicted disease. The system also extracts the top three classes with their corresponding probabilities for more detailed feedback. These results are formatted and displayed back to the user on the result page of the Flask app. A confidence percentage is shown next to the predicted class to help users understand the model's certainty[16]. If the model or image processing fails, the system shows an error message with guidance. This algorithm ensures quick, accurate, and user-friendly disease detection. It combines efficient image handling, deep learning inference, and real-time web integration.

2.3 Techniques:

The first key technique used in this project is image preprocessing, which ensures that the input data is clean and suitable for analysis. Images uploaded by users are resized to a fixed dimension (192x192), converted from BGR to RGB format, and normalized to improve model performance[18]. These steps are essential for standardizing inputs and ensuring compatibility with the CNN model. Additionally, pixel values are scaled to a range of 0 to 1, which helps the model converge faster during training and improves accuracy during prediction[19]. Image augmentation techniques like rotation, zooming, and flipping can also be applied to expand the dataset and reduce overfitting. Proper preprocessing directly impacts the quality of model learning. This ensures that each input is interpreted consistently by the deep learning algorithm.

The second core technique is the implementation of a Convolutional Neural Network (CNN) using the Keras deep learning library. CNNs are highly effective in image classification tasks due to their ability to capture spatial hierarchies and textures in images. The model consists of multiple layers such as convolutional layers, activation functions (ReLU), max pooling layers, and dense layers for classification. These layers help in extracting features like edges, patterns, and shapes that are common across different skin diseases[20]. The network is trained using a labeled dataset of skin disease images, with a loss function such as categorical cross-entropy. Optimization is carried out using techniques like Stochastic Gradient Descent or Adam Optimizer. Dropout layers are used to prevent overfitting during training and improve generalization.

The third important technique is web application integration using Flask, which allows the trained model to be used in a real-time setting. The Flask framework handles image uploads, server requests, and routing, providing an easy-to-use interface for end users. Once an image is uploaded, it is sent through the preprocessing pipeline and passed to the trained model for prediction. The predicted result is then displayed on the web interface along with confidence scores and top predictions. This enables a seamless and interactive experience for users, even in non-clinical environments. Flask also makes it possible to deploy the system locally or on cloud platforms. This integration of deep learning with web development bridges the gap between complex models and real-world usability.

2.4 Tools:

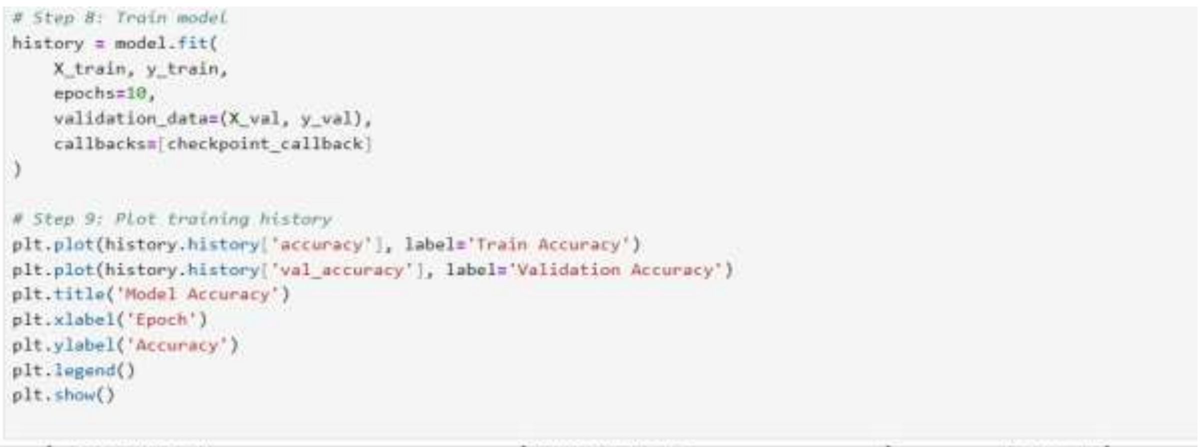
The development of this skin disease detection system involves several powerful tools and technologies. Python is used as the primary programming language due to its simplicity and strong support for machine learning. Keras and TensorFlow are used to build and train the Convolutional Neural Network (CNN) model for image classification. NumPy and OpenCV handle image processing tasks such as resizing, normalization, and color space conversion. For the user interface, the system uses Flask, a lightweight Python web framework that enables real-time interaction between users and the model. HTML and CSS are used for designing the front-end web interface. Jupyter Notebook is used during the development and testing phase for writing and debugging code. Matplotlib and Seaborn help visualize training metrics and results. The model is saved using Joblib or Pickle for later use in prediction. All these tools together create an efficient, interactive, and scalable deep learning application for healthcare use.

2.5 Methods:

The skin disease detection system follows a structured methodology combining deep learning and web development. It begins with data collection, where a labeled dataset of skin disease images is gathered for training. Next, image preprocessing is performed, which includes resizing images to a fixed size, converting them to RGB format, and normalizing pixel values. These preprocessed images are then used to train a Convolutional Neural Network (CNN) built using Keras and TensorFlow. The CNN model includes multiple layers such as convolution, pooling, and dense layers that learn to extract features and classify diseases. During training, the model uses a loss function and an optimizer (like Adam) to minimize errors. Model evaluation is done using metrics like accuracy and loss, along with validation datasets to avoid overfitting. After achieving good performance, the model is saved and integrated into a Flask web application. The Flask app handles user interactions, allowing users to upload images through a browser interface. The uploaded image is passed through the same preprocessing steps before being sent to the CNN model for prediction. The model then outputs the predicted disease along with the confidence score. The result is displayed back to the user through a user-friendly interface. This method ensures a seamless flow from input to output, enabling real-time and reliable skin disease diagnosis.

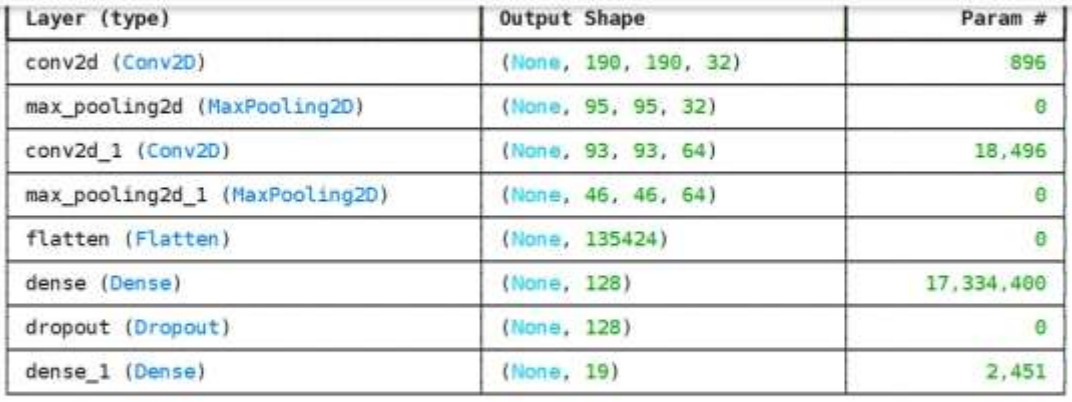
III. METHODOLOGY

3.1 Input: The input to the skin disease detection system is a digital image of the affected skin area, uploaded by the user through the web interface. This image can be in formats such as .jpg, .jpeg, or .png. Once uploaded, the image is passed through a preprocessing pipeline where it is resized to a fixed dimension (e.g., 192x192 pixels) and converted to RGB color format to match the model's expected input. The pixel values are normalized to a range of 0 to 1 for better model performance. The system expects clear images taken in good lighting conditions to minimize noise and maximize detection accuracy. Inputs can vary widely in terms of skin tone, background, and disease type, so proper preprocessing is essential. Each image should focus closely on the skin lesion or affected area to avoid irrelevant features. The input image is then reshaped into a format suitable for the CNN model as a NumPy array. This standardized input helps the model identify patterns and classify the disease effectively. The quality, clarity, and correctness of the input image directly affect the accuracy of the prediction. Therefore, the input plays a crucial role in the performance of the overall system.



```
# Step 8: Train model
history = model.fit(
    X_train, y_train,
    epochs=10,
    validation_data=(X_val, y_val),
    callbacks=[checkpoint_callback]
)

# Step 9: Plot training history
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 190, 190, 32)	896
max_pooling2d (MaxPooling2D)	(None, 95, 95, 32)	0
conv2d_1 (Conv2D)	(None, 93, 93, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 46, 46, 64)	0
flatten (Flatten)	(None, 135424)	0
dense (Dense)	(None, 128)	17,334,400
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 19)	2,451

Fig 1: Custom CNN for Multi-Class Skin Disease Classification

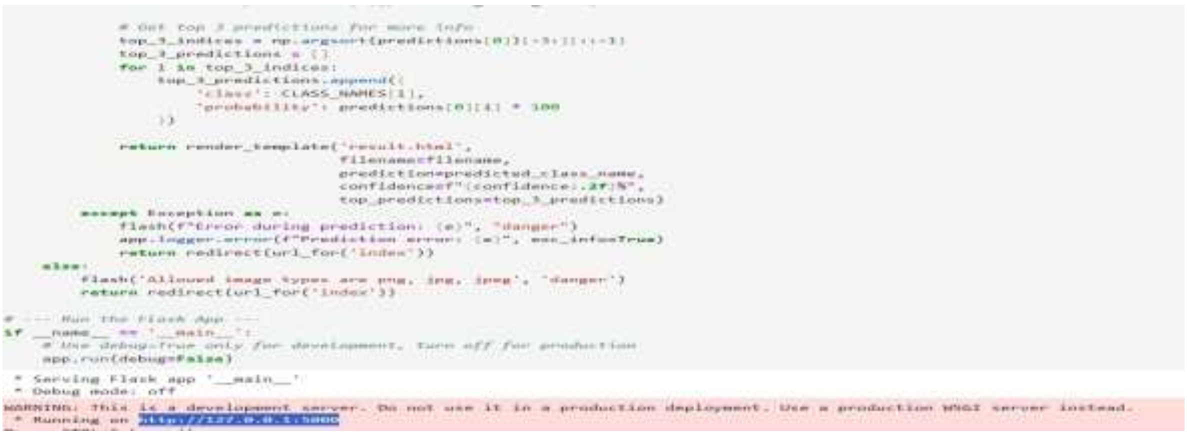
The input for this system is not limited to just the image itself but also includes several implicit factors that affect prediction accuracy. For instance, the camera quality, lighting conditions, and clarity of the skin lesion are important elements that influence model performance. A well-focused image with the affected area centered increases the chances of a correct prediction. Users are encouraged to avoid images with distractions like jewelry, clothing, or blurred backgrounds. Additionally, the system assumes that the uploaded image corresponds to a single skin condition and not multiple overlapping issues. Images should not include unnecessary annotations, filters, or edits that may distort original features. Proper guidance on input image standards ensures that the CNN model receives consistent and meaningful data for reliable predictions.

3.2 Method of Process:

The process begins when a user accesses the web interface and uploads an image of the affected skin area. The system first checks the file type to ensure it is a valid image format such as JPG or PNG. Once validated, the image is saved temporarily and passed through a preprocessing stage, where it is resized, normalized, and converted to RGB format. This preprocessed image is then converted into a NumPy array suitable for model input. The image is forwarded to the Convolutional Neural Network (CNN) model built using Keras, which analyzes the image and predicts the probabilities for each disease class. The model identifies the most likely skin condition based on the highest probability score. The system also extracts the top three predictions for better insight. These results, along with their confidence levels, are sent to the front-end via Flask. The user sees the prediction results instantly on the web page. The entire process takes only a few seconds, enabling real-time diagnosis. In case of any errors during upload or prediction, the system displays helpful messages. This method ensures a fast, accurate, and user-friendly way to detect skin diseases through image analysis.

3.3 Output:

The output of the skin disease detection system is a predicted classification of the uploaded skin image, identifying the most probable disease from a predefined set of classes. Once the image is processed and analyzed by the CNN model, the system returns the predicted disease name along with a confidence percentage indicating how certain the model is about its prediction. This result is displayed clearly on the result page of the Flask web application. To enhance user understanding, the system also provides the top three predictions with their respective probabilities, offering a broader view of possible conditions. The prediction is shown along with the uploaded image, allowing users to verify that the correct file was used. The results are rendered in real-time, making the process fast and efficient. This instant output is especially valuable in early-stage detection and in areas lacking access to dermatologists. Additionally, the output interface is designed to be clean and user-friendly, even for non-technical users. The result can serve as a preliminary screening tool, encouraging timely medical consultation. If the model fails or the input is invalid, an appropriate error message is shown. The system ensures transparency and reliability by clearly displaying model feedback. Ultimately, the output empowers users with quick, accessible, and informative skin health insights.

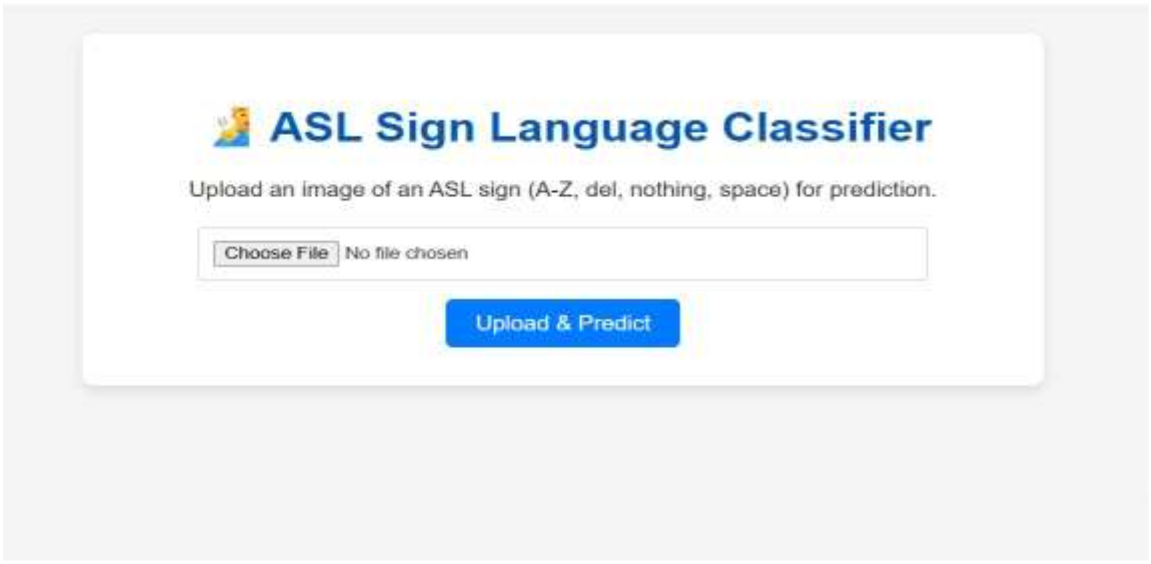


```
            # Get top 3 predictions for more info
            top_3_indices = np.argsort(predictions[0])[-3:][::-1]
            top_3_predictions = []
            for i in top_3_indices:
                top_3_predictions.append({
                    'class': CLASS_NAMES[i],
                    'probability': predictions[0][i] * 100
                })

            return render_template('result.html',
                                   filename=filename,
                                   prediction=predicted_class_name,
                                   confidence=f"{confidence:.2f}%",
                                   top_predictions=top_3_predictions)
        except Exception as e:
            flash(f"Error during prediction: {e}", "danger")
            app.logger.error(f"Prediction error: {e}", exc_info=True)
            return redirect(url_for('index'))
    else:
        flash('Allowed image types are png, jpg, jpeg', 'danger')
        return redirect(url_for('index'))

# --- Run the Flask App ---
if __name__ == '__main__':
    # Use debug=True only for development, turn off for production
    app.run(debug=False)
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
```

Fig: Skin Disease Classifier Web App



ASL Sign Language Classifier

Upload an image of an ASL sign (A-Z, del, nothing, space) for prediction.

Fig: ASL Sign Language Classifier

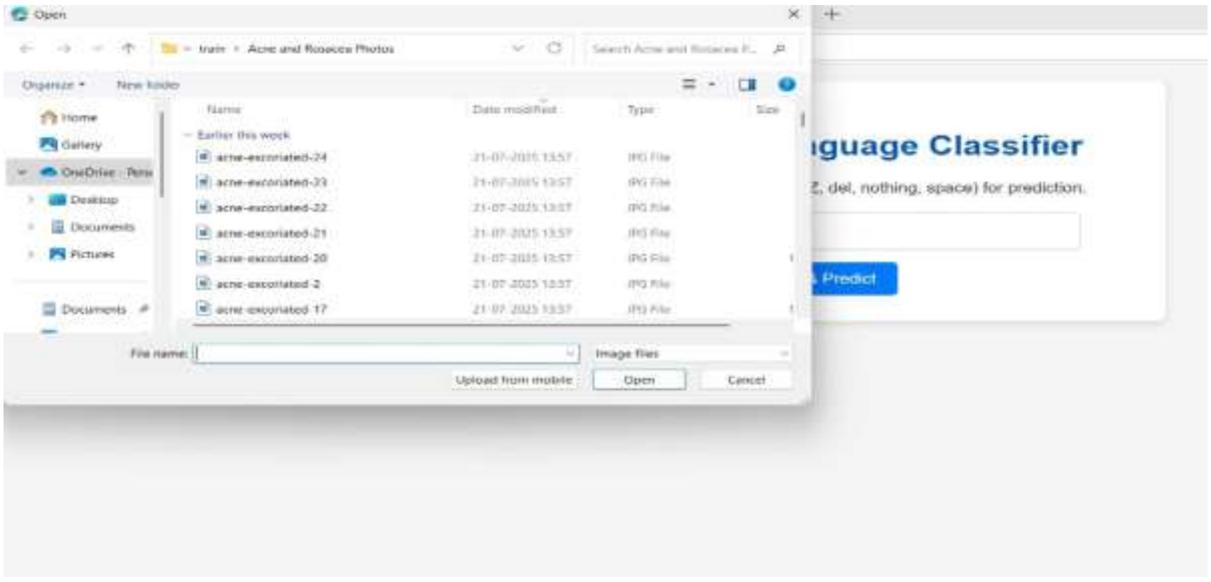


Fig: Skin Disease Dataset – Acne Excoriated Subset

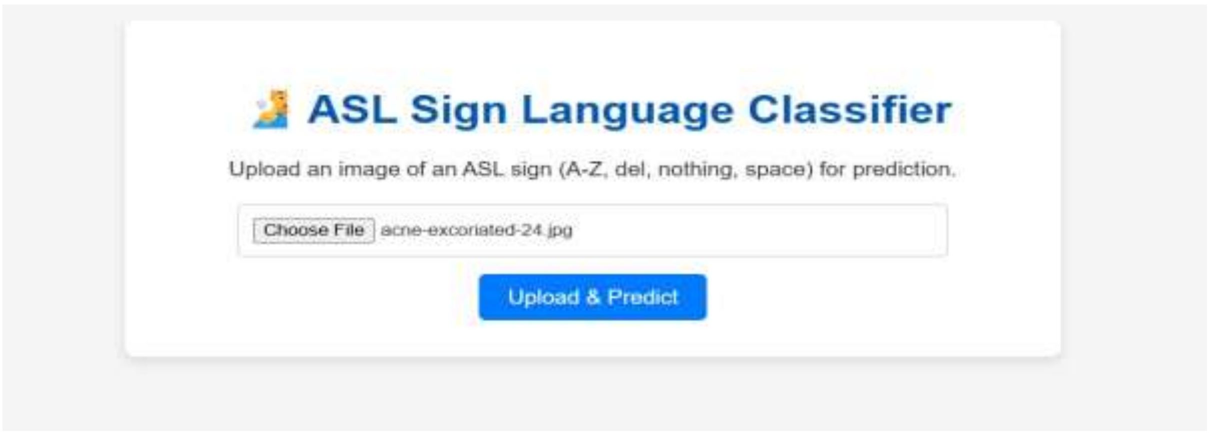

ASL Sign Language Classifier
Upload an image of an ASL sign (A-Z, del, nothing, space) for prediction.
Choose File acne-excoriated-24.jpg
Upload & Predict

Fig: ASL Sign Language Classifier Upload image

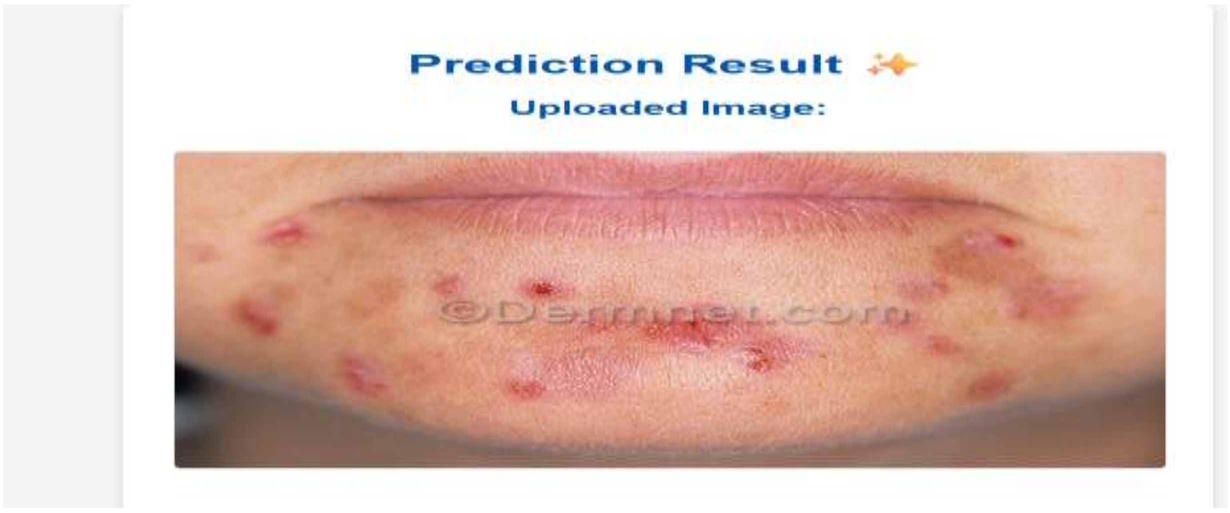
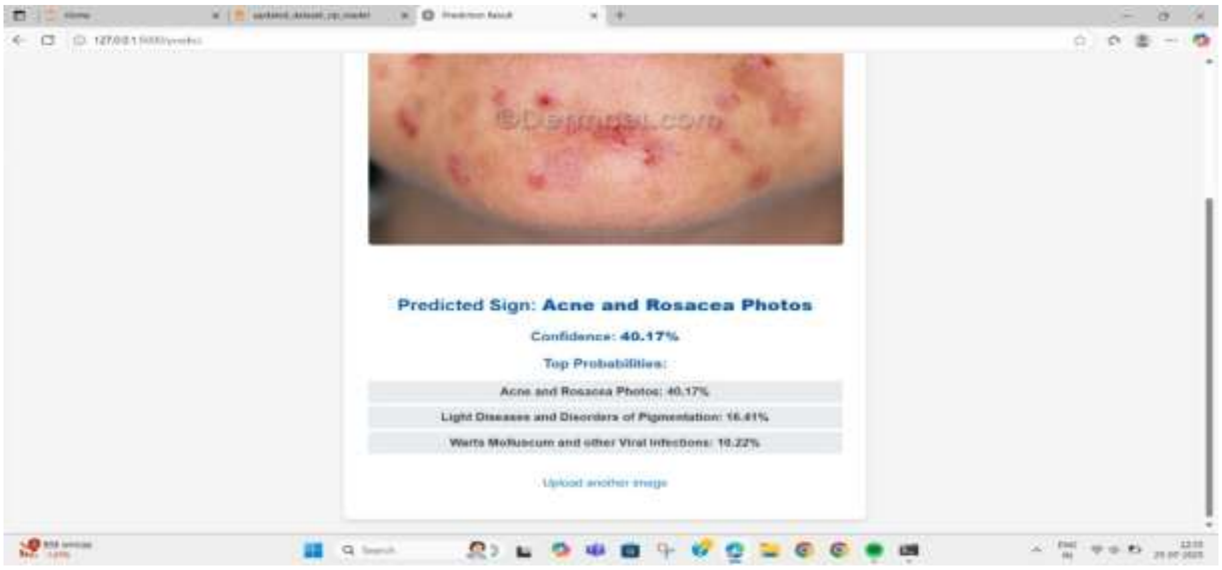


Fig: Prediction Result skin disease



Predicted Sign: Acne and Rosacea Photos

Confidence: 40.17%

Top Probabilities:

Acne and Rosacea Photos: 40.17%

Light Diseases and Disorders of Pigmentation: 16.41%

Warts Molluscum and other Viral Infections: 10.22%

Fig: Acne and Rosacea Photos

IV. RESULTS:

The result of the system is a real-time prediction of the skin condition based on the image uploaded by the user. After processing the image through the trained CNN model, the system displays the most likely skin disease along with a confidence score, indicating how certain the model is about its classification. Additionally, the top three predictions with their respective probabilities are shown to provide further insight. This allows users to understand alternative possibilities and the relative confidence between them. The output is delivered through a clean and intuitive web interface built using Flask. Users can view the uploaded image alongside the result to verify the accuracy of the submission. The model's fast response time ensures immediate feedback, making it suitable for quick diagnosis. The result serves as a preliminary assessment and encourages users to consult medical professionals if needed. In cases where the input image is unclear or invalid, the system prompts the user with an error message. Overall, the result is designed to be accurate, informative, and accessible for users at all levels.

V. DISCUSSIONS:

The skin disease detection system effectively demonstrates the use of deep learning in medical image analysis. By leveraging Convolutional Neural Networks, the model can identify skin conditions with a good level of accuracy. The integration of the model with a Flask-based web application makes it accessible and easy to use for non-technical users. The system provides real-time predictions, which is valuable for early detection, especially in remote or underserved areas. During testing, the model performed well on clear, focused images but struggled slightly with blurry or low-quality inputs. This highlights the importance of proper image preprocessing and dataset diversity. The confidence scores and top predictions enhance the reliability of the results. However, it is important to note that the system serves as a preliminary diagnostic tool, not a replacement for professional medical advice. Expanding the dataset and adding more disease categories could further improve performance. Overall, the system offers a promising solution for supporting dermatological care through AI.

VI. CONCLUSION:

In conclusion, the proposed system successfully applies Convolutional Neural Networks to detect and classify various skin diseases from images. The integration with a Flask web application allows users to receive real-time predictions in a simple and accessible way. This approach enhances early diagnosis, especially in areas with limited access to dermatological care. The system has demonstrated reliable performance on a well-preprocessed dataset. While it is not a replacement for professional diagnosis, it serves as a helpful screening tool. Future improvements can focus on expanding the dataset and enhancing model accuracy for broader medical application.

VII. FUTURE SCOPE:

In the future, this system can be enhanced by including a wider variety of skin diseases and a larger, more diverse dataset. Integration with mobile apps can improve accessibility in remote areas. Real-time camera input and multilingual support can make it more user-friendly. The system can also be extended to detect other medical conditions through image analysis. With further development, it could become a reliable preliminary diagnostic tool in telemedicine.

VIII. ACKNOWLEDGEMENT:

Rampilli Bhanu Sankar working as Assistant professor in Master of Computer Applications (MCA) at Sankethika Vidya Parishad Engineering college, Visakhapatnam, Andhra pradesh, Accredited by NAAC. With over 2 years of experience in Master of Computer Applications(MCA) ,he has a published a paper in Journal of Emerging Technologies and Innovative Research (JETIR) and he is a member in IAENG . His area of expertise include C, Data Structures, Java Programming , Python Programming .

Potnuru Manikanta is pursuing his final semester MCA in Sanketika Vidya Parishad Engineering College, accredited with A grade by NAAC, affiliated by Andhra University and approved by AICTE. With interest in Machine learning P. Manikanta has taken up his PG project on SKIN DISEASE IDENTIFICATION USING DEEP LEARNING TECHNIQUES and published the paper in connection to the project under the guidance of Rampilli Bhanu Sankar, Assistant Professor, SVPEC.

REFERENCES

1. Chollet, F. (2015). Keras.

https://keras.io

2.**TensorFlow Developers. (2023).** TensorFlow: An end-to-end open-source machine learning platform.

https://www.tensorflow.org/

3.OpenCV Team. (2023). OpenCV (Open Source Computer Vision Library).

https://opencv.org/

4. Flask Documentation. (2023). Flask web framework.

https://flask.palletsprojects.com/

5.Werkzeug. (2023). The WSGI utility library used by Flask.

https://werkzeug.palletsprojects.com/

6. **Mozilla Developer Network (MDN).** CSS Reference.

https://developer.mozilla.org/en-US/docs/Web/CSS

7. **W3C.** HTML5 Specification. *https://www.w3.org/TR/html52/*

8. **GitHub Repository**: *Skin-Disease-Identification-Using-Image-Analysis*
https://github.com/Krisshvamsi/Skin-Disease-Identification-Using-Image-Analysis

9. **Microsoft.** Configuration.xml Reference for Office Deployment Tool.
https://learn.microsoft.com/en-us/deployoffice/

10. **WHO. (2005).** Epidemiology and Management of Common Skin Diseases in Children in Developing Countries. *World Health Organization*.

11. LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning.* Nature, 521(7553), 436–444.
https://doi.org/10.1038/nature14539

12. Simonyan, K., & Zisserman, A. (2015).*Very Deep Convolutional Networks for Large-Scale Image Recognition.* arXiv preprint arXiv:1409.1556
https://arxiv.org/abs/1409.1556

13. He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep Residual Learning for Image Recognition.* CVPR.
https://doi.org/10.1109/CVPR.2016.90

14. Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks.* NeurIPS.
https://papers.nips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

15. Esteva, A., Kuprel, B., Novoa, R. A., et al. (2017). *Dermatologist-level classification of skin cancer with deep neural networks.* Nature, 542(7639), 115–118.
https://doi.org/10.1038/nature21056

16. Tschandl, P., Rosendahl, C., & Kittler, H. (2018).
The HAM10000 dataset: A large collection of multi-source dermatoscopic images. Scientific Data.
https://doi.org/10.1038/sdata.2018.161

17. Codella, N. C. F., Rotemberg, V., Tschandl, P., et al. (2018).
Skin lesion analysis toward melanoma detection: A challenge at the ISIC 2018. ISBI 2018.
https://doi.org/10.1109/ISBI.2018.8363547

18. Araújo, T., Aresta, G., Castro, E., Rouco, J., & Campilho, A. (2017).
Classification of breast cancer histology images using Convolutional Neural Networks.
Pattern Recognition, 85, 81–90.
https://doi.org/10.1016/j.patcog.2018.07.014

19. Litjens, G., Kooi, T., Bejnordi, B. E., et al. (2017).
A survey on deep learning in medical image analysis. Medical Image Analysis, 42, 60–88.
https://doi.org/10.1016/j.media.2017.07.005

20. WHO. (1997).
Improving Child Health: Integrated Management of Childhood Illnesses.
World Health Organization, Geneva. WHO/CHD/97.12.
https://www.who.int/publications/i/item/who-chd-97.12