A COMPARATIVE STUDY ON SKIN LESION DETECTION AND CLASSIFICATION USING TRANSFER LEARNING MODELS

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

JADA VISWA TEJA	**21A51A05F9**
MURAKALA PRASANTHI	**21A51A05D6**
BANAPURAM HEMASUNDARA RAO	**21A51A05G9**
ARANGI VIKAS	**21A51A05J2**
CHELLURU PAVAN KUMAR	**21A51A05I4**

Under the Guidance of

Sri. G. SURYA PAVAN KUMAR M. Tech, (Ph.D.)
Assistant Professor, Dept. of CSE.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ADITYA INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(An Autonomous Institution)
Approved by AICTE, Permanently Affiliated to JNTUGV – Vizianagaram
Accredited by NBA & NAAC with A+, Recognized Under 2(f) 12(B) of UGC
K.Kotturu, Tekkali, Srikakulam - 532201, Andhra Pradesh.

May – 2025

DECLARATION

"We hereby declare that the project entitled ***"A Comparative Study on Skin Lesion Detection and Classification Using Transfer Learning Models."*** *Submitted for the award of the degree of Bachelor of Technology in* ***Computer Science and Engineering*** *is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the completion for the award of any other degree, associate ship, fellowship or any other similar titles.*

PLACE: Tekkali	**JADA VISWA TEJA**	**21A51A05F9**
DATE :	**MURAKALA PRASANTHI**	**21A51A05D6**
	BANAPURAM HEMASUNDARA RAO	**21A51A05G9**
	ARANGI VIKAS	**21A51A05J2**
	CHELLURU PAVAN KUMAR	**21A51A05I4**

ADITYA INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(An Autonomous Institution)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project report entitled "A COMPARATIVE STUDY ON SKIN LESION DETECTION AND CLASSIFICATION USING TRANSFER LEARNING MODELS" being submitted by **Jada Viswa Teja 21A51A05F9, Murakala Prasanthi 21A51A05D6, Banapuram HemaSundara Rao 21A51A05G9, Arangi Vikas 21A51A05J2, Chelluru Pavan Kumar 21A51A05I4** in partial fulfillment for the award of the Degree of Bachelor of Technology in **Computer Science and Engineering** to the Jawaharlal Nehru Technological University, Gurajada, Vizianagaram is a record of bonafied work carried out under my guidance and supervision.

Dr. Y. Ramesh, M .Tech, Ph.D.
Head of the Department CSE

Sri. G. Surya Pavan Kumar M.Tech,(Ph.D.)
Assistant Professor, Dept. of CSE.

ACKNOWLEDGEMENT

We wish to thank **Sri G. Surya Pavan Kumar** for his kind support and his valuable suggestions and encouragement helped us a lot in carrying out this project work as well as in bringing this project to this form.

We take this opportunity to express our sincere gratitude to our Director **Prof. V. V. Nageswara Rao** for his encouragement in all respect.

We take the privilege to thank our principal **Dr. A. S. Srinivasa Rao** for his encouragement and support.

We are also very much thankful to **Dr. Y.Ramesh**, Head of Computer Science & Engineering for his help and valuable support in completing the project.

We are also thankful to all staff members in the Department of Computer Science and Engineering, for their feedback in the reviews and kind help throughout our project.

Last but not the least, we thank all our classmates for their encouragement and help in making this project a success.

It is their help and support, due to which we became able to complete the design and technical report.

Jada Viswa Teja	**21A51A05F9**
Murakala Prasanthi	**21A51A05D6**
Banapuram HemaSundara Rao	**21A51A05G9**
Arangi Vikas	**21A51A05J2**
Chelluru Pavan Kumar	**21A51A05I4**

Project Details

Batch No : 36

Project Title : A Comparative Study on Skin Lesion Detection and Classification Using Transfer Learning Models.

Area /Domain : Deep Learning (Transfer learning)

Application : Health care

Description of the Project : This project employs deep learning and transfer learning such as VGG19, InceptionV3, EfficientNetB0,DenseNet201, MobileNetV2, and ResNet50 to detect and classify skin lesions. In order to overcome class imbalance, it incorporates hybrid models with classifiers like XGBoost, Random Forest, and SVM while using preprocessing methods like data augmentation and undersampling. MobileNetV2 surpasses hybrid models with the maximum accuracy of 91.11%, according to the study. The objective is to improve early identification of skin cancer and support healthcareprofessionals with AI-powered diagnostic tools.

Objectives : The objective is to improving the skin cancer early detection and categori--zation.Additionally addressing challenges such as class imbalance through data preprocessing techniques ensures better model performance and robustness.

Project Outcome : This study investigated the efficacy of transfer learning **models** for skin lesion classification Using architectures like VGG19, InceptionV3, Efficient-NetB0, DenseNet201, MobileNetV2, and ResNet50, as well as hybrid approaches combining classifiers like XGBoost, Random Forest, and SVM.The results showed significant insights into model performance on the HAM10000 dataset. MobileNetV2, a strong standalone model, outperformed all other models, including VGG 19, DensNet 201, and Resnet 50,Inception V3,Efficient Net B0 and both hybrid models with classifiers, with an exceptional test accuracy of 91.11%. The performance of MobileNetV2 and EfficientNetB0 are closely identical, demonstrating the potential of how well they classify medical images.

Batch Details

Project Guide

Surya Pavan Kumar Gudla

Assistant Professor

Ph: 9490346746

Pavan1980.mca@gmail.com

Jada Viswateja
Roll No. :21A51A05F9
Viswatejajada4@gmail.com
Contact No: 9704046877

Murakala Prasanthi
Roll No. :21A51A05D6
murakalaprasanthi@gmail.com
Contact No:6305052737

Banapuram Hemasundar Rao
Roll No. :21A51A05G9
Hemasundar241@gmail.com
Contact No:9573498405

Arangi Vikas
Roll No. :21A51A05J2
Vikasarangi333@gmail.com
Contact No:8688628141

Chelluru Pavan Kumar
Roll No. :21A51A05I4
Kumarpavan52828@gmail.com
Contact No:8179713625

Program Outcomes (PO)

1. **ENGINEERING KNOWLEDGE:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **PROBLEM ANALYSIS:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **DESIGN/DEVELOPMENT OF SOLUTIONS:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **CONDUCT INVESTIGATIONS OF COMPLEX PROBLEMS:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **MODERN TOOL USAGE:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding ofthe limitations.
6. **THE ENGINEER AND SOCIETY:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevantto the professional engineering practice.
7. **ENVIRONMENT AND SUSTAINABILITY:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **ETHICS:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **INDIVIDUAL AND TEAM WORK:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **COMMUNICATION:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able tocomprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **PROJECT MANAGEMENT AND FINANCE:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **LIFE-LONG LEARNING:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes

- **PSO1:-** Apply mathematical foundations, algorithmic principles, and theoretical computer science in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- **PSO2:-** Demonstrate understanding of the principles and working of the hardware and software aspects of computer systems.
- **PSO3:-** Use knowledge in various domains to identify research gaps and hence to provide solution to new ideas and innovations

PO-PSO Mapping

PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
3	3	3	3	3	2	2	1	2	2	2	3	3	2	3

ABSTRACT

Early and accurate skin lesion classification is essential for detecting melanoma and other skin cancers. When it comes to medical image analysis, it is crucial for early melanoma diagnosis, which lowers death rates. This study boosts feature extraction and classification by examining deep learning models such as VGG19, InceptionV3, EfficientNetB0, DenseNet201, MobileNetV2 and ResNet50, along with transfer learning hybrid models that incorporate classifiers like XGBoost and Random Forest, and SVM. The HAM10000 dataset is utilized, and preprocessing methods like undersampling, and data augmentation are employed to address class imbalance. Remarkably, the results demonstrated that MobileNetV2 standalone performed better than its hybrid transfer learning models, with the highest train and test accuracies of 98.22% and 91.11% respectively.

Keywords— Convolutional Neural Networks, Data Augmentation, Class imbalance, Transfer Learning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1. **AI** – Artificial Intelligence
2. **ML** – Machine Learning
3. **DL** – Deep Learning
4. **CNN** – Convolutional Neural Network
5. **SVM** – Support Vector Machine
6. **RF** – Random Forest
7. **XGBoost** – extreme Gradient Boosting
8. **MLP** – Multi-Layer Perceptron
9. **VGG** – Visual Geometry Group (e.g., VGG16, VGG19)
10. **ResNet** – Residual Network (e.g., ResNet50)
11. **DenseNet** – Densely Connected Convolutional Network (e.g., DenseNet201)
12. **InceptionV3** – Inception Version 3
13. **EfficientNet** – Efficient Neural Network (e.g., EfficientNet B0)
14. **MobileNetV2** – Mobile Convolutional Neural Network Version 2
15. **HAM10000** – Human Against Machine with 10,000 Training Images Dataset
16. **ISIC** – International Skin Imaging Collaboration
17. **SMOTE** – Synthetic Minority Over-sampling Technique (for handling class imbalance)
18. **RGB** – Red Green Blue (color model for images)
19. **GLCM** – Gray Level Co-occurrence Matrix (texture feature extraction**)**
20. **ROC** – Receiver Operating Characteristic Curve
21. **AUC** – Area Under the
22. **TL** – Transfer Learning
23. **ReLU** – Rectified Linear Unit (activation function)
24. **SGD** – Stochastic Gradient Descent
25. **Adam** – Adaptive Moment Estimation (optimizer
26. **TPU** – Tensor Processing Unit
27. **GPU** – Graphics Processing Unit

Chapter – 1

INTRODUCTION

1.1 INTRODUCTION

The rise in skin malignancies like melanoma worldwide, skin lesion identification and classification have become crucial areas in medical image analysis. Melanoma is a serious worldwide health issue and one of the most severe and deadliest forms of skin cancer. One of the main causes of high death rates from these cancers is their late identification, especially in areas with little access to cutting-edge healthcare. Therefore, improving treatment outcomes, lowering mortality, and increasing patient survival rates all depend on early and precise skin problem diagnosis.

Several things such as infections, inflammation, exposure to the environment, or genetic susceptibility, can result in skin lesions, which show up as abnormal growths or changes in the texture and colour of the skin. It's difficult and demanding to distinguish between benign and malignant lesions, particularly melanoma, and it calls for extreme precision. Conventional diagnostic techniques, such as dermatologists eye examination and biopsy-based confirmation, are frequently laborious, arbitrary, and resource-intensive. As a result, medical image analysis has accelerated the use of sophisticated computational methods, especially deep learning.

Specifically Convolutional Neural Networks (CNNs), in deep learning have notably advanced the field of automated medical diagnosis by allowing for the rapid and reliable extraction of detailed data from medical images. Reliable classification is made possible by CNN-based architectures, which have shown impressive abilities in identifying patterns, textures, and changes in skin lesions. In the classifying of skin lesions, the most advanced CNN models—including VGG19, Inception, ResNet50, and DenseNet201—have shown remarkable efficacy. To improve their performance on more specialized, smaller datasets, these models use transfer learning after being trained on large-scale datasets like HAM10000.

The objective is to improving the skin cancer early detection and categorization. Additionally, addressing challenges such as class imbalance through data preprocessing techniques ensures better model performance and robustness.

1.2 PROBLEM STATEMENT

Early detection and accurate classification of skin lesions are critical for effective treatment and patient outcomes. Traditional diagnostic methods are time- consuming, heavily reliant on the expertise of dermatologists, and susceptible to human error.

1.3 OBJECTIVE

The goal of this project is to develop a deep learning model using CNN and its variants to accurately detect and classify skin lesions, contributing to the early detection of skin cancers.Additionally, addressing challenges such as class imbalance through data preprocessing techniques ensures better model performance and robustness.

1.4 SKIN LESSION

Skin lesions can occur in any individual, in varying forms, and ranging from a mild rash to conditions which require a medical evaluation. There are skin lesions or diseases that are caused by infections, for example, impetigo is a skin infection that causes crusty sores and herpes simplex can cause painful, blisters and sores. Other skin lesions or diseases are due to inflammation, such as eczema and psoriasis, which cause red, itchy and scaling areas causing discomfort, pruritus and sometimes chronic symptoms. Skin allergies can also be a source of skin problems, i.e., contact dermatitis will create redness, swelling and itching after the exposure to an irritant skin care product. Skin diseases or disorders can also occur due to an auto immune problem (such as lupus or pemphigus vulgaris) that are associated with long-term blistering or ulcers.

In some cases, skin lesions can be a symptom of a more serious condition, skin cancer. Melanoma, basal cell carcinoma, and squamous cell carcinoma in most cases will begin as abnormal mole or sore, which will change in appearance over time. Diabetic ulcers or pressure sores, if not treated appropriately, can worsen and lead to an infection or long-term consequences. Early diagnosis is very important- skin exams, biopsies or lab tests can be helpful to distinguish if the source of the skin problem is benign or a concerning issue. Treatments options can range from antibiotics and medicated creams to more complex medications such as surgery- related treatments. No matter the cause, paying attention to changes in the skin and seeking medical advice when needed can make a big difference in managing and treating skin lesions effectively.

1.4.1 Malignant Skin Lesion (Type 1):

Melanoma is a serious and potentially life-threatening form of skin cancer that develops from melanocytes, the pigment-producing cells in the skin. It often appears as an irregularly shaped mole or dark spot that changes over time. Melanomas can vary in color, ranging from black, brown, or even blue, red, or white. Unlike benign moles, they tend to have uneven borders, asymmetry, and a diameter greater than 6 mm. Other warning signs include rapid growth, itching, bleeding, or ulceration. Early detection is crucial, as melanoma can spread to other parts of the body if not treated promptly.

1.4.2 Benign Skin Lesion (Type 2):

Benign skin lesions, such as regular moles, freckles, or seborrheic keratoses, are non-cancerous and typically harmless. They are usually symmetrical, have smooth and well-defined edges, and remain consistent in size, shape, and color over time. Moles (nevi) can be tan, brown, or black, and they often appear during childhood or adolescence. Other common benign lesions include cherry angiomas, which are small red bumps, and dermatofibromas, which feel firm under the skin. While benign lesions do not pose a health risk, any sudden changes in size, shape, or color should be evaluated by a dermatologist to rule out malignancy.

1.5 APPROACH TO THE PROBLEM

To tackle this problem effectively, we begin by understanding the core requirements and constraints. First, we break down the problem into smaller, manageable components and identify key variables that influence the solution.

Next, we explore different methodologies or algorithms that can be applied, considering factors such as efficiency, accuracy, and scalability. We evaluate potential edge cases and limitations to ensure robustness.

Once a suitable approach is identified, we implement the solution step by step, optimizing where necessary. Finally, we test the solution thoroughly to validate its correctness and refine it based on results.This structured approach ensures a clear, logical, and efficient resolution to the problem.

1.6 ADVANTAGES

- **Improved Diagnostic Accuracy**

The primary advantage of this project is the ability to detect skin lesions early, particularly melanoma, which is one of the deadliest forms of skin cancer. Early detection is crucial for improving survival rates, as melanoma treatment is most effective when caught in the early stages.

Traditional methods rely heavily on the expertise of dermatologists, who can be subject to fatigue or oversight. By automating the detection process, the system can help minimize human errors, ensuring more consistent and accurate results.

- **Reduced Time and Cost**

The deep learning model can process skin lesion images significantly faster than manual methods. This can reduce the time required to diagnose and triage patients, allowing healthcare professionals to focus on treatment and follow-up care.

Once trained, the deep learning model can be deployed with minimal computational overhead, offering a cost-effective solution to healthcare systems, especially in regions with limited access to specialized dermatology professionals.

- **Scalability**

The system can be easily scaled to handle large datasets ofmedical images, making it applicable in different clinical settings. With minimal adjustments, it can be integrated into various healthcare systems worldwide, even in areas with limited access to dermatologists. The potential to deploy this system globally, especially in regions with high skin cancer incidence but a shortage of dermatologists (e.g., rural areas or developing countries), can significantly impact global health outcomes.

- **Enhancement of Medical Expertise**

While the system does not replace dermatologists, it acts as a powerful decision- support tool that can assist in diagnosing skin lesions. By providing additional insights and recommendations, it enables dermatologists to make more informed decisions, particularly in complex cases.

This project can serve as an educational resource for dermatology students and professionals, providing examples of varied skin lesion cases and assisting them in understanding how a deep learning model interprets these cases.

- **Data-Driven Insights**

  The project will contribute to the field by offering insights into the kinds of features CNNs learn to classify skin lesions, which could lead to a deeper understanding of the characteristics of benign vs. malignant lesions.

  By continuously refining and improving the model with new data, the system can evolve over time, becoming more accurate and reliable. This could lead to the development of models with broader applications, including real-time surveillance and monitoring of lesions.

- **Enhanced Patient Outcomes:**

  The system can help ensure that lesions are detected and classified correctly, leading to earlier and more accurate diagnoses. Early intervention typically leads to better prognosis and treatment outcomes for patients, ultimately saving lives.

  In the long term, this project could lead to applications where patients themselves can take high-quality images of their skin lesions using a mobile device. This could facilitate remote screening and provide more people with access to early detection.

Chapter – 2

LITERATURE SURVEY

1. **Gururaj Harinahalli Lokesh, et al. "DeepSkin: a deep learning approach for skin cancer classification." IEEE Access 11 (2023): 50205-50214.**

   An improved deep learning-based method for classifying skin cancer using dermoscopic pictures is presented in the research "DeepSkin: A Deep Learning Approach for Skin Cancer Classification" by Gururaj Harinahalli Lokesh et al., which was published in IEEE Access (2023). Effective treatment of skin cancer, one of the most common and fatal types of cancer, depends on early and precise detection. Conventional diagnosis techniques mostly rely on the subjective and time-consuming expertise of dermatologists.

   These techniques improves dependability by mixing outputs from several models to further enhance predictions. The study also looks at Grad-CAM (Gradient-weighted Class Activation Mapping) to give explainability, which enables dermatologists to know which areas of a picture have the biggest influence on the model's judgement.

2. **Mahdiraji, S. A., Baleghi, Y., &Sakhaei, S. M. (2017, April). Skin lesion images classification using new color pigmented boundary descriptors. In 2017 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA) (pp. 102-107). IEEE.**

   In the 2017 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA), Mahdiraji, Baleghi, and Sakhaei's paper "Skin Lesion Images Classification Using New Colour Pigmented Boundary Descriptors" presents a novel method for classifying skin lesion images using color-pigmented boundary descriptors. By concentrating on the boundary features of skin lesions, which are essential in differentiating between benign and malignant lesions, the study seeks to increase the accuracy of skin cancer detection. This study focusses on local boundary features obtained from pigmented areas of the lesion, as opposed to traditional image classification techniques that frequently rely on global aspects like texture and colour distribution.

The study "Classification of Skin Lesion Images Using New Colour." By taking advantage of the fact that benign lesions have smoother, more uniform borders while malignant lesions usually have irregular, asymmetric, and unevenly pigmented edges, the suggested method extracts colour and shape-based descriptors from lesion boundaries. In conclusion, by integrating boundary-based colour pigmented descriptors that enhance lesion differentiation, this study offers a reliable and effective technique for skin lesion categorisation. The results of the study help to improve automated systems for detecting skin cancer, opening the door to more precise and trustworthy diagnostic instruments in medical imaging and dermatology. To improve classification performance and generalisation across various datasets, future research could investigate combining these handcrafted features with deep learning algorithms.

3. **Hegde P. R., Shenoy M. M., & Shekar, B. H. (2018, September). Comparison of machine learning algorithms for skin disease classification using color and texture features. In 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 1825-1828). IEEE.**

Hegde, Shenoy, and Shekar's paper "Comparison of Machine Learning Algorithms for Skin Disease Classification Using Colour and Texture Features" examined how well different machine learning algorithms classified skin diseases using colour and texture features taken from dermoscopic images. It was presented at the 2018 International Conference on Advances in Computing, Communications, and Informatics (ICACCI). Effective treatment of skin illnesses, including benign and malignant lesions, depends on early and accurate diagnosis. This research focusses on conventional machine learning models and assesses how well they perform in categorising various skin disorders, even though deep learning techniques have become more popular in medical picture analysis.

According to the results, SVM and Random Forest outperform other algorithms and achieve higher classification accuracy because of their proficiency with high- dimensional feature spaces. Although k-NN and Naïve Bayes are computationally efficient, the study shows that they often perform poorly when it comes to differentiating between specific types of skin diseases. The results highlight how crucial it is to choose the right feature extraction methods and classifiers when developing computer-aided diagnostic(CAD) systems for the identification of skin diseases.To sum up, the study offers insightful information on how well machine learning models perform when classifying skin diseases using manually created colour and texture parameters. According to the study, hybrid models that combine deep learning and machine learning techniques may help future studies further increase classification accuracy.

4. **Hameed N., Shabut A. M., & Hossain M. A. (2018, December). Multi-class skin disease classification using deep convolutional neural network and support vector machine. In 2018, the 12th International Conference on Software, Knowledge, Information Management & Applications (SKIMA) (pp. 1-7). IEEE.**

   By combining deep convolutional neural networks (CNNs) and support vector machines (SVMs), the paper "Multi-Class Skin Disease Classification Using Deep Convolutional Neural Network and Support Vector Machine" by Hameed, Shabut, and Hossain, which was presented at the 12th International Conference on Software, Knowledge, Information Management & Applications (SKIMA) 2018, suggests a hybrid approach for multi-class skin disease classification. The study tackles the increasing demand for automated and precise skin disease diagnosis because conventional techniques are frequently laborious, arbitrary, and necessitate skilled dermatologists. To improve categorisation accuracy and dependability, the researchers use machine learning and deep learning approaches.

   The study emphasises how these hybrid models can be used in computer-aided diagnosis (CAD) systems to provide automated, non-invasive, and affordable dermatological analysis options. The authors do, however, recognise certain difficulties, including the requirement for real-world clinical validation, computational complexity, and dataset imbalance. Subsequent research endeavours may concentrate on augmenting model resilience via expanded datasets, enhanced preprocessing methodologies, and instantaneous implementation in telemedicine applications.

   All things considered, this study advances the creation of AI-powered skin disease classification systems by proving how well CNN-based feature extraction and SVM classification work together to increase accuracy and generalisation. The work uses deep learning and hybrid AI models to pave the stage for future developments in dermatological diagnostics and medical image analysis.

5. **Liu X., Hu G., Ma X., & Kuang H. (2019, June). An enhanced neural network based on deep metric learning for skin lesion segmentation. In 2019 Chinese Control And Decision Conference (CCDC) (pp. 1633-1638). IEEE.**

   An advanced neural network model incorporating deep metric learning for skin lesion segmentation is proposed in the paper "An Enhanced Neural Network Based on Deep Metric Learning for Skin Lesion Segmentation" by Liu, Hu, Ma, and Kuang, which was presented at the 2019 Chinese Control and Decision Conference (CCDC). Because it directly affects the functionality of later classification and detection algorithms, accurate skin lesion segmentation is an essential stage in computer-aided diagnostic (CAD) systems. In

dermoscopic pictures, noise, low contrast, and uneven lesion margins are common problems for traditional segmentation techniques. The researchers use a deep metric learning-based neural network to improve segmentation accuracy.

The suggested approach enhances feature representation and border delineation by combining deep metric learning with convolutional neural networks (CNNs). To improve performance even further, future research might concentrate on combining real-time segmentation models, multi-scale feature extraction, and attention mechanisms.Overall, this study shows the promise of deep metric learning in medical image processing by offering a unique deep learning technique that enhances the precision and dependability of skin lesion segmentation. By using AI-powered methods to detect and diagnose skin conditions early, the suggested model lays the groundwork for more efficient and scalable CAD systems

6. **Mahbod, A., Schaefer, G., Wang, C., Ecker, R., & Ellinge, I. (2019, May). Skin lesion classification using hybrid deep neural networks. In ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP) (pp. 1229-1233).**

   A hybrid deep learning method for skin lesion classification is examined in the paper "Skin Lesion Classification Using Hybrid Deep Neural Networks" by Mahbod, Schaefer, Wang, Ecker, and Ellinge, which was presented at the ICASSP 2019 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). Deep learning-based automated classification of skin lesions has drawn a lot of attention due to the rising prevalence of skin cancer, especially melanoma. Despite their high performance, conventional deep learning models like CNNs sometimes suffer from overfitting, feature redundancy, and class imbalance. The researchers suggest a hybrid deep neural network (DNN) model that incorporates many architectures to improve classification performance in order to overcome these drawbacks.

   The study comes to the conclusion that hybrid architectures, which combine numerous deep neural networks, can greatly enhance the categorisation of skin cancer and result in computer-aided diagnostic (CAD) systems that are more accurate and dependable. Nonetheless, the researchers recognise difficulties including the requirement for larger annotated datasets and the high computing cost. To improve classification performance even more, future research should investigate lightweight hybrid models. Overall, this study shows that combining many CNN designs results in a more reliable and effective categorisation of skin lesions, highlighting the potential of hybrid deep neural networks in

automated dermatological diagnosis. Future developments in AI- powered medical imaging and early skin disease identification are made possible by this study.

7. **Kondaveeti H. K., & Edupuganti, P. (2020, December). Skin cancer classification using transfer learning. In 2020 IEEE International Conference on Advent Trends in Multidisciplinary Research and Innovation (ICATMRI) (pp. 1-4). IEEE.**

   Kondaveeti and Edupuganti's paper "Skin Cancer Classification Using Transfer Learning" examines the use of transfer learning for automated skin cancer classification. It was presented at the 2020 IEEE International Conference on Advent Trends in Multidisciplinary Research and Innovation (ICATMRI). Given the rising incidence of skin cancer, especially melanoma, early detection is essential to successful treatment. Large datasets and intensive feature engineering are frequently needed for traditional machine learning techniques, which makes them impractical for practical clinical applications. The study uses transfer learning, which makes use of pre-trained deep learning models to enhance classification performance with sparse labelled data, to overcome these constraints.

   The work concludes by demonstrating that pre-trained CNN models may attain high accuracy even with a small amount of labelled data, thereby validating the efficacy of transfer learning in medical picture classification. In order to further improve automated skin cancer diagnosis, the study proposes that future developments might use clinical metadata, ensemble learning, and real-time mobile applications. The development of AI-powered computer-aided diagnostic (CAD) devices is facilitated by this study, providing a non-invasive, affordable, and easily accessible method of early skin cancer detection.

8. **Ali Ahmed S. A., Yanıkoğlu B., Gabay Ö., &Aptoula E. (2020). Skin lesion classification with deep CNN ensembles.**

   The application of ensemble deep learning models for automated skin lesion classification is examined in the work "Skin Lesion Classification with Deep CNN Ensembles" by Ali Ahmed, Yanıkoğlu, Gabay, and Aptoula (2020). One of the most prevalent and lethal types of cancer is skin cancer, especially melanoma, and increasing survival rates requires early detection. Even though deep convolutional neural networks (CNNs) have demonstrated encouraging outcomes in the categorisation of medical images, individual CNN models frequently have trouble generalising, particularly when working with intricate and unbalanced datasets. The paper suggests an ensemble-based deep learning strategy to

   According to the findings, ensemble learning greatly enhances the classification of skin lesions, which makes it a viable strategy for dermatology computer-aided diagnostic (CAD)

systems. According to the study, real-time deployment for telemedicine applications, attention mechanisms, and the integration of clinical metadata could be the main areas of future research. The study demonstrates how deep learning ensembles may be used to analyse medical images using AI, providing a more accurate and dependable method for automated skin cancer identification.

9. **Raut, R., Borole, Y., Patil, S., Khan, V. N., &Takale, D. G. (2022).Skin disease classification using machine learning algorithms. NeuroQuantology, 20(10), 9624-9629**

The use of machine learning algorithms for automated skin illness categorisation is examined in the work "Skin Disease Classification Using Machine Learning Algorithms" by Raut, Borole, Patil, Khan, and Takale (2022), which was published in NeuroQuantology.For effective treatment, skin diseases—both benign and malignant— need to be diagnosed as soon as possible. Conventional diagnosis techniques rely on the subjective and time-consuming expertise of dermatologists. The study investigates different machine learning (ML) methods to effectively categorise skin disorders in order to overcome these issues.

Because of their capacity to manage intricate, high-dimensional feature spaces, the results show that SVM and Random Forest classifiers outperform other ML models and attain the maximum accuracy. The study shows that even while k-NN and Decision Trees are computationally efficient, they often poorly when skin lesions show notable intra-class variability. The results highlight how crucial feature selection and preprocessing methods like noise reduction and image enhancement are to raising classification accuracy.The study concludes by showing that machine learning algorithms are capable of accurately classifying skin conditions, laying the groundwork for computer-aided diagnostic (CAD) systems. This study advances the creation of automated, non-invasive, and easily accessible diagnostic tools for the early diagnosis of skin diseases by utilising AI-driven classification algorithms. lesions show notable intra-class variability. The results highlight how crucial feature selection and preprocessing methods like noise reduction and image enhancement are to raising classification accuracy.The study concludes by showing that machine learning algorithms are capable of accurately classifying skin conditions, laying the groundwork for computer-aided diagnostic (CAD) systems. This study advances the creation of automated, non-invasive, and easily accessible diagnostic tools for the early diagnosis of skin diseases by utilising AI-driven classification algorithms.

lesions show notable intra-class variability. The results highlight how crucial feature selection and preprocessing methods like noise reduction and image enhancement are to raising classification accuracy.The study concludes by showing that machine learning algorithms are capable of accurately classifying skin conditions, laying the groundwork for computer-aided diagnostic (CAD) systems. This study advances the creation of automated, non-invasive, and easily accessible diagnostic tools for the early diagnosis of skin diseases by utilising AI-driven classification algorithms.

10. **Debelee, T. G. (2023). Skin lesion classification and detection using machine learning techniques: a systematic review. Diagnostics, 13(19), 3147.**

A thorough analysis of machine learning (ML) methods used for skin lesion classification and detection may be found in Debelee's study "Skin Lesion Classification and Detection Using Machine Learning Techniques: A Systematic Review" (2023), which was published in Diagnostics. Automated diagnostic systems have drawn a lot of attention due to their potential to improve early detection and accuracy as the number of skin cancer cases worldwide continues to climb. The results of several investigations are combined in this study, which contrasts deep learning, hybrid, and conventional machine learning techniques for skin lesion analysis.

The review also emphasises the difficulties in classifying skin lesions, such as imbalanced datasets, variations in lesion appearance, and computational complexity. The paper examines methods including data augmentation, image preprocessing, attention processes, and multi-modal learning to overcome these problems. Additionally, as the majority of machine learning models are trained on publically accessible datasets that might not accurately reflect a range of skin types and disorders, it highlights the necessity of real-world clinical validation.To sum up, the study offers insightful information about the developments, difficulties, and potential paths of machine learning-based skin lesion classification. It implies that early diagnosis, treatment planning, and accessibility for patients globally could be greatly enhanced by incorporating AI-driven models into telemedicine and clinical decision support systems.

Table-2.1 Literature survey

Year	Project Title	Author Name	Methodology	References
2023	"DeepSkin: a deep learning approach for skin cancer classification." *IEEE Access* 11 (2023):50205-50214.	Gururaj Harinahalli Lokesh, et al.	PreprocessedHAM10000 and ISIC data, applied K-means, and trained CNN, SVM, and transfer learning models.	[1]
2017	Skin lesion images classification using new color pigmented boundary descriptors.In *2017 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA)* (pp. 102-107). IEEE. (2017, April).	Mahdiraji S. A., Baleghi Y., &Sakhaei S. M. (2017, April).	The study used a hybrid method combining AlexNET for feature extraction with ECOC SVM for skin lesion classification, validated on 9,144 images using 10-fold cross-validation.	[2]
2018	Comparison of machine learning algorithms for skin disease classification using color and texture features. IEEE. (2018, September).	Hegde P. R., Shenoy M. M., & Shekar B. H.	Classified skin diseases using RGB and GLCM features with LDA, SVM,ANN, and Naive Bayes on 157 images, finding LDA and SVM to be the most effective.	[3]
2018	Multi-class skin diseases classification using deep convolutional neural network and support vector machine. IEEE. (2018, December).	Hameed N., Shabut A. M., & Hossain M. A.	This study used a hybrid method combining AlexNET for feature extraction with SVM for skin lesion classification, validated on 9,144 images using 10-fold cross-validation	[4]

2019	An enhanced neural network based on deep metric learning for skin lesion segmentation. In *2019 Chinese Control And Decision Conference (CCDC)* (pp. 1633-1638). IEEE. (2019, June)	Liu X., Hu G., Ma X., &Kuang H.	The study developed a DMLEN for skin lesion segmentation, achieving a 3% reduction in MAE with ISIC 2017 and PH2 datasets, using TensorFlow, Keras, and OpenCV.	[5]
2019	Skin lesion classification using hybrid deep neural networks.“ *IEEE. ICASSP-2019*	MahbodAmirreza, et al.	Fusing pre-trained model with Alexnet along with multi-classifier SVM for evaluation results	[6]
2020	Skin cancer classification using transfer learning. In *2020 IEEE International Conference on Advent Trends in Multidisciplinary Research and Innovation (ICATMRI)* (pp. 1-4). IEEE. (2020, December).	Kondaveeti H. K., & Edupuganti P.	Utilized HAM10000 data and transfer learning with CNNs (ResNet50, InceptionV3, MobileNet, Xception) on Google Colab.	[7]
2020	Skin lesion classification with deep CNN ensembles. (2020).	Ali Ahmed S. A., Yanıkoğlu B., Gabay Ö., &Aptoula E.	Used CNNs and Isolation Forest to achieve top results in the ISIC2019 challenge, utilizing TensorFlow/Keras and LightGBM.	[8]

2022	Skin disease classification using machine learning algorithms.*NeuroQuantology*, *20*(10), 9624- 9629. (2022).	Raut R., Borole Y., Patil S., Khan V., &Takale D. G.	The study classified skin diseases using ANN, SVM, KNN, and DT on the PH2 and additional datasets, implemented in MATLAB and Keras, achieving the highest accuracy with ANN	[9]
2023	Skin Lesion Classification and Detection Using Machine Learning Techniques: *Diagnostics*, *13*(19), 3147. (2023).	Debelee, T. G.	The research explored deep learning and machine learning algorithms for skin disease classification, emphasizing advancements, accuracy challenges, and the integration of AI into clinical practice.	[10]

Chapter-3

METHODOLOGY

3.1 DATASET

3.1.1 DATASET DESCRIPTION

The **HAM10000 (Human Against Machine with 10000 training images)** dataset is a large collection of dermatoscopic images used for training and testing machine learning models in skin lesion classification. It was created to support research in dermatology and artificial intelligence, specifically for the automatic detection of skin cancer and other dermatological conditions. The dataset consists of **10,015** high-resolution images, which include both common and rare skin diseases. These images were collected from different sources, ensuring diversity in patient demographics, imaging techniques, and lesion types.

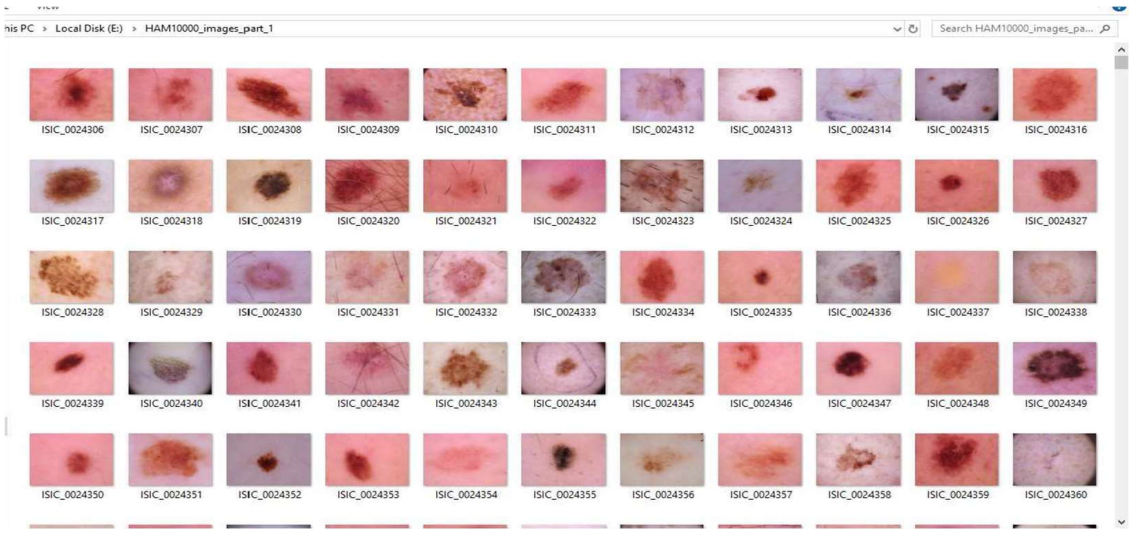


Fig 3.1– Image Dataset

HAM10000 includes seven different types of skin lesions:(NV), melanomas (MEL), benign keratosis-like lesions melanocytic nevi (BKL), basal cell carcinomas (BCC), actinic keratoses and intraepithelial carcinomas (AKIEC), vascular lesions (VASC), and dermatofibromas (DF). Each image in the dataset is labeled with one of these seven categories, allowing researchers to train supervised machine-learning models for accurate diagnosis. The dataset was carefully curated by dermatology experts to ensure high-quality annotations.

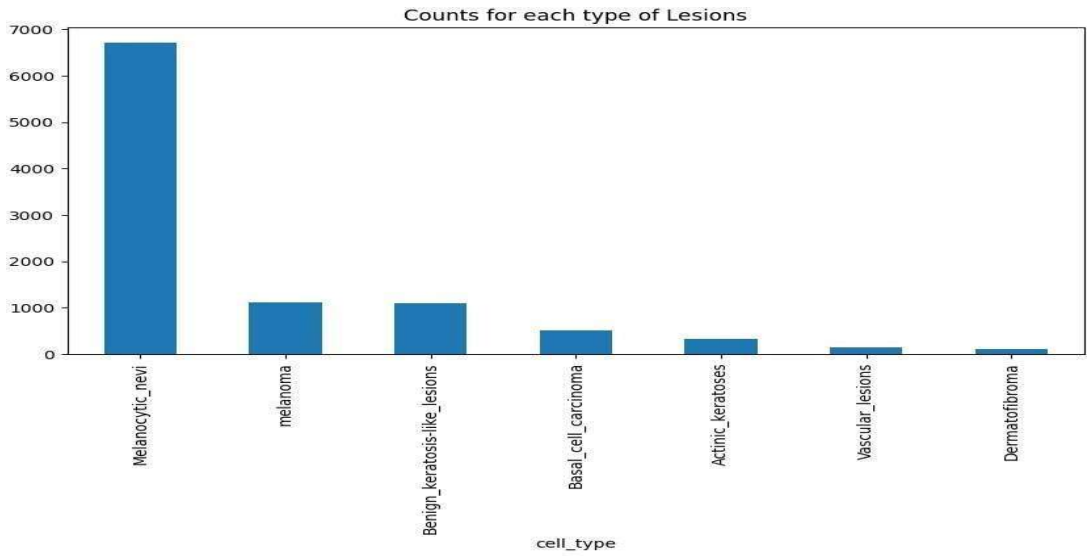


Fig 3.2 – Dataset Information

To create a diverse and balanced dataset, images were gathered from different sources, including the Department of Dermatology at the Medical University of Vienna and the Skin Cancer Practice in Queensland, Australia. Some images were collected retrospectively, while others were acquired through teledermatology services. The dataset was pre-processed to ensure consistency in image quality, and some images were augmented to increase their representation in underrepresented categories.

HAM10000 has been widely used in medical AI research, particularly for deep learning applications in dermatology. Researchers have used it to train convolutional neural networks (CNNs) and other machine learning models for automated skin lesion classification. The dataset is publicly available on platforms such as Kaggle and the International Skin Imaging Collaboration (ISIC) archive, making it an essential resource for advancements in medical imaging and artificial intelligence.

3.1.2 Data Preprocessing

This dataset class distribution is of 8061zeroes and 1954 ones. SMOTE enhances minority class representation, while random sampling maintains the original dataset size. Replacing zeros with columns ensures smoother data and reduces feature bias. The final balanced dataset with 5060 zeros and 4955 ones,is then saved as undersampled_balanced_ham10000.csv,is better prepared for training models, leading to improved classification performance and more reliable predictions.

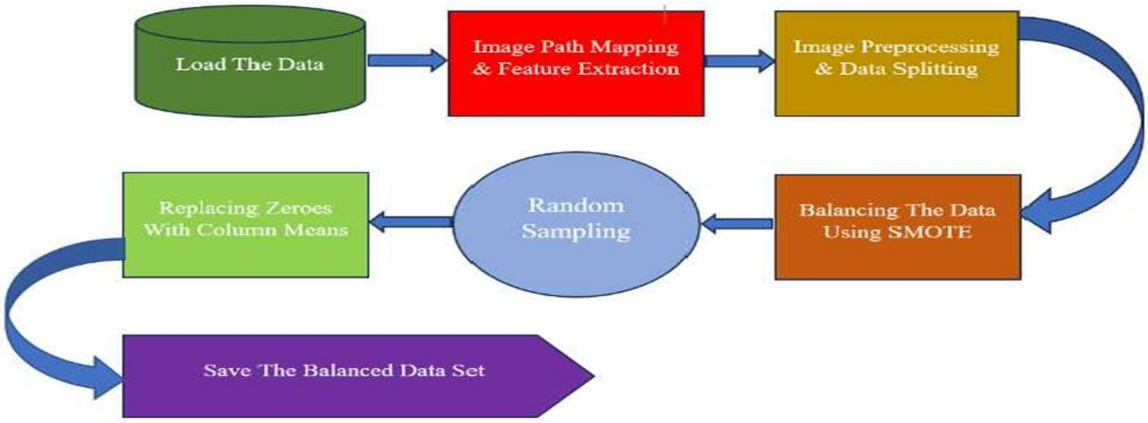


Fig.3.3 Feature preprocessing

3.2 MACHINE LEARNING

Machine learning is the study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

In machine learning, an unknown universal dataset is assumed to exist, which contains all the possible data pairs as well as their probability distribution of appearance in the real world. The acquired dataset is called the training set (training data) and used to learn the properties and knowledge of the universal dataset. In general, vectors in the training set are assumed independently and identically sampled from the universal dataset.

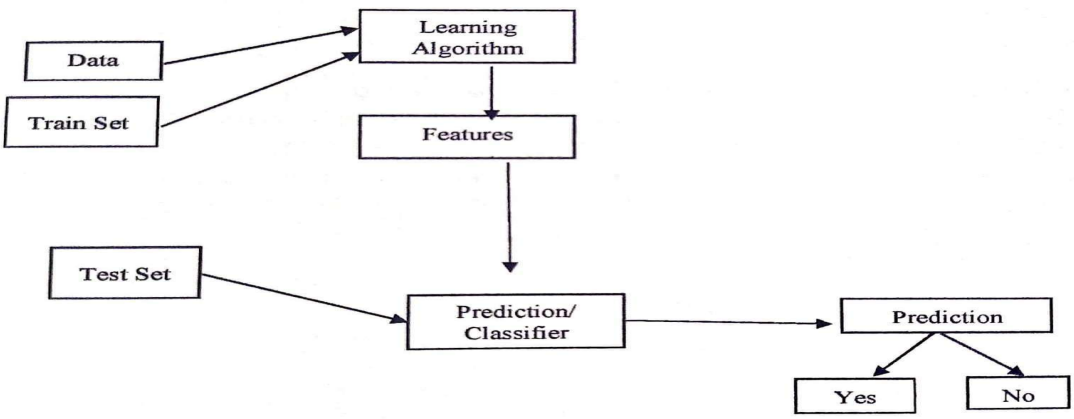


Fig 3.4 –Machine Learning

The figure 3.2 shows the process of machine learning in which the data sets are classified as train set and test set are forwarded to the learning algorithms then it predicts the presence or absence of certain features.

In machine learning, these learned properties cannot only explain the training set, but also be used to predict unseen samples or future events. In order to examine the performance of learning, another dataset may be reserved for testing, called the test set or test data.For example, before final exams, the teacher may give students several questions for practice (training set), and the way he judges the performances of students is to examine them with another problem set (test set)]. In order to distinguish the training set and the test set D and D when they appear together by using train and test to denote them, respectively.

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.

The discipline of machine learning employs various approaches to teach computers to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist, one approach is to label some of the correct answers as valid. This can then be used as training data for the computer to improve the algorithms it uses to determine correct answers.

Modern day machine learning has two objectives, one is to classify data based on models which have been developed, the other purpose is to make predictions for future outcomes based on these models. A hypothetical algorithm specific to classifying data may use computer vision of moles coupled with supervised learning in order to train it to classify the cancerous moles. Where as, a machine learning algorithm for stock trading may inform the trader of future potential predictions.

3.3 DEEP LEARNING

Deep Learning is a subset of machine learning that focuses on training artificial neural networks with multiple layers to learn from data. It is inspired by the structure and functioning of the human brain, where neurons process information through interconnected layers. Deep learning has gained immense popularity due to its ability to automatically extract features from raw data, reducing the need for manual feature engineering.

At the core of deep learning are artificial neural networks (ANNs), which consist of an input layer, multiple hidden layers, and an output layer. Each neuron in a layer is connected to neurons in the next layer through weighted connections. These weights are adjusted during training using optimization techniques like backpropagation and gradient descent, allowing the network to learn patterns and relationships in the data. The deeper the network (i.e., the more layers it has), the more complex representations it can learn.

Deep learning has been particularly successful in computer vision, natural language processing (NLP), speech recognition, and robotics. In computer vision, deep learning models like Convolutional Neural Networks (CNNs) are used for tasks such as image classification, object detection, and facial recognition. In NLP, architectures such as Recurrent Neural Networks (RNNs), Long Short- Term Memory (LSTMs), and Transformers are widely used for language translation, sentiment analysis, and text generation. Additionally, deep learning powers applications like voice assistants, self-driving cars, and medical diagnostics.One of the main advantages of deep learning is its ability to handle large and complex datasets. Traditional machine learning models often struggle with high- dimensional data, whereas deep learning models can automatically learn hierarchical features, making them highly effective for complex tasks. However, deep learning requires large amounts of data and significant computational resources, as training deep networks can be time-consuming and computationally expensive.

Despite these challenges, deep learning continues to evolve, with advancements in hardware (such as GPUs and TPUs) and techniques like transfer learning, self- supervised learning, and federated learning improving its efficiency. As research in deep learning progresses, its applications are expanding across various industries, making it one of the most transformative technologies in artificial intelligence today.

3.4 TRANSFER LEARNING

Transfer Learning is a machine learning technique where a model trained for one task is adapted to a different but related task. Instead of starting from scratch, transfer learning allows a model to leverage the knowledge it has already gained from solving a previous problem. This approach is particularly useful when dealing with limited labelled data and computational resources, making it a popular choice in deep learning applications.

In transfer learning, a model is first pre-trained on a large dataset, often for a general-purpose task such as image classification or language processing. The pre- trained model learns important features, such as edges, shapes, and patterns in images or word relationships in text. Once trained, the model can be fine-tuned on a smaller, task-specific dataset. This fine-tuning process involves modifying or retraining certain layers of the model while keeping others frozen to retain previously learned features.

This technique is widely used across different domains, including computer vision, natural language processing (NLP), and medical imaging. In computer vision, models like VGG16, ResNet, and EfficientNet are often pre-trained on large datasets like ImageNet and later fine-tuned for specific tasks such as object detection, facial recognition, or skin disease classification. In NLP, models such as BERT, GPT, and T5 are pre-trained on massive text corpora and then adapted to perform specific tasks like sentiment analysis, text summarization, or machine translation.

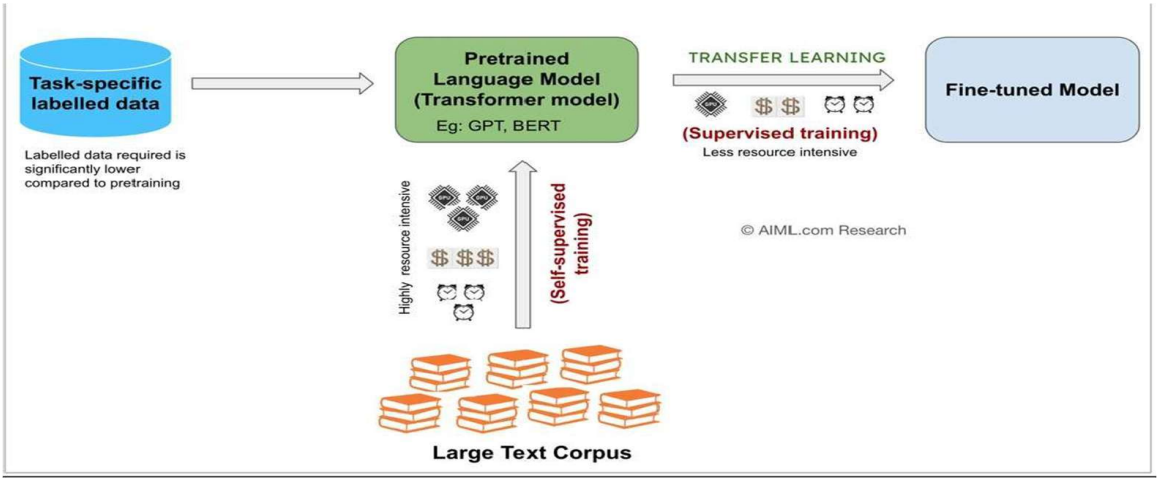


Fig 3.5- Transfer Learning

One of the key advantages of transfer learning is its ability to improve model performance with limited data**.** Training deep learning models from scratch requires extensive labeled datasets, which may not always be available, especially in specialized fields like healthcare and scientific research. Transfer learning reduces the need for large datasets by allowing models to generalize knowledge from one domain to another efficiently.

Additionally, transfer learning significantly reduces training time and computational costs. Since the model has already learned fundamental patterns from the pre-training phase, fine-tuning requires fewer training iterations, leading to faster convergence and lower hardware requirements. This makes transfer learning an attractive option for researchers and developers working with constrained resources. Overall, transfer learning has revolutionized the field of artificial intelligence by making deep learning more accessible, efficient, and adaptable. It enables breakthroughs in applications where data is scarce and helps improve model accuracy in real-world tasks. As AI continues to evolve, transfer learning will remain a powerful tool in developing smarter and more learning solutions. applications where data is scarce and helps improve model accuracy in real-world tasks.

3.5 TRANSFER LEARNING MODELS

3.5.1 VGG19

The VGG19 model is designed with a collection of convolutional layers that progressively incorporate more filters as additional layers are added, allowing for deeper feature extraction. It consists of sixteen convolutional layers, five pooling layers, and three fully connected layers, making it a deep and powerful architecture for image classification and other computer vision tasks. The pooling layers are implemented using 2 × 2 max pooling windows, which help reduce spatial dimensions while preserving important features.One of the key design principles of VGG19 is its use of small filters (3 × 3 kernel size), as research has shown that smaller receptive fields can be just as effective as larger ones while requiring fewer parameters and improving computational efficiency. By stacking multiple small filters, the model captures complex patterns and hierarchical features, leading to improved accuracy. This architectural choice makes VGG19 particularly effective for deep feature learning, making it a widely used model in applications such as image recognition, obj ect detection, and medical image analysis.

3.5.2 RESNET 50

ResNet50 (Residual Network 50) is a deep convolutional neural network architecture that addresses the challenge of training very deep networks by incorporating skip connections or residual connections. These connections allow gradients to flow directly through the network during backpropagation, effectively preventing the vanishing gradient problem, which often occurs in deep networks. By enabling identity mappings, ResNet allows deeper models to be trained without suffering from the degradation problem, where increasing depth does not necessarily lead to better performance. This architecture is based on residual learning, where the network learns residual functions instead of trying to learn the full transformation at once. This approach makes it easier to train deep networks while maintaining high accuracy and efficiency.

ResNet50 is a 50-layer deep version of ResNet, composed of convolutional layers, batch normalization, ReLU activation functions, and fully connected layers. It follows a pattern of stacked residual blocks, each containing convolutional layers and shortcut connections. The model has proven to be highly effective in image classification, object detection, and feature extraction tasks, particularly when pre- trained on large-scale datasets such as ImageNet. Due to its strong generalization ability and robustness, ResNet50 is widely used in real- world applications, including medical image analysis, autonomous driving, and face recognition. Its efficiency and accuracy make it one of the most popular architectures in deep learning.

3.5.3 DENSENET 201

DenseNet (Densely Connected Convolutional Networks) is a deep learning architecture that enhances information flow within the network by establishing dense connections between layers. Unlike traditional convolutional networks, where each layer only passes information to the next layer, DenseNet connects each layer to every other layer in a feed-forward fashion. This unique design ensures that feature maps learned by earlier layers are directly accessible to later layers, promoting feature reuse and reducing redundancy. One of the key advantages of this architecture is that it helps mitigate the vanishing gradient problem, which can occur in very deep networks by ensuring stronger gradient propagation during backpropagation. Additionally, DenseNet requires fewer parameters compared to other deep networks since it eliminates the need to learn redundant feature representations, making it computationally efficient and memory-friendly.

One of the most powerful variants of this architecture is DenseNet201, which consists of 201 layers and is highly effective in classification tasks. The model is structured using multiple dense blocks, where each layer receives input from all preceding layers and passes its own output to all subsequent layers. This approach improves training efficiency and leads to better generalization

on complex datasets. DenseNet201 has demonstrated outstanding performance in various applications, including image classification, medical image analysis, and object detection, particularly when pre-trained on large datasets such as ImageNet. Due to its ability to learn rich hierarchical features with fewer parameters, DenseNet201 has become a popular choice in deep learning research and real-world implementations.

3.5.4 INCEPTION V3

InceptionV3 is a deep convolutional neural network architecture known for its efficient computational design and ability to learn complex representations. Unlike traditional deep networks that stack layers sequentially, InceptionV3 incorporates a unique Inception module, which applies multiple convolutional filters of different sizes within the same layer. This allows the network to capture features at multiple scales simultaneously, improving its ability to recognize patterns across varying spatial dimensions. By using 1×1, 3×3, and 5×5 convolutions in parallel, the model can extract both fine and coarse details from input images, leading to better feature extraction and improved classification accuracy. Additionally, the model employs factorized convolutions and asymmetric convolutions to enhance computational efficiency, reducing the number of parameters while maintaining high performance.

A key advantage of InceptionV3 is its ability to achieve state-of-the-art performance while being computationally efficient, making it suitable for both high-end GPU processing and resource-constrained environments. The model incorporates techniques such as batch normalization, label smoothing, and auxiliary classifiers to stabilize training and improve generalization. InceptionV3 has been widely used in image classification, object detection, and transfer learning applications, particularly when pre-trained on large datasets like ImageNet. Its ability to learn multi-scale features effectively has made it a popular choice in various industries, including medical imaging, autonomous driving, and facial recognition, where high accuracy and efficiency are essential.

3.5.5 EFFICIENT NET B0

EfficientNet-B0 is a convolutional neural network (CNN) designed for high- performance image classification while maintaining computational efficiency. It was trained on over a million images from the ImageNet dataset, allowing it to learn complex patterns and features for various classification tasks. Unlike traditional deep networks that scale arbitrarily by increasing depth or width, EfficientNet-B0 utilizes a novel compound scaling method, which systematically balances depth (number of layers), width (number of channels), and resolution (input image size). This approach ensures that the network achieves optimal performance without unnecessary computational overhead, making it a highly efficient model for real-world applications.

One of the key advantages of EfficientNet-B0 is its compact architecture, which enables it to deliver high accuracy with fewer parameters and lower computational cost compared to larger CNNs. By optimizing the network's structure, EfficientNet-B0 maintains strong feature extraction capabilities while being suitable for deployment on mobile devices, embedded systems, and cloud platforms. Its ability to handle large-scale image classification tasks with minimal resource consumption has made it a popular choice in various domains, including medical imaging, object detection, and real-time AI applications. Due to its efficiency and scalability, EfficientNet has been extended into larger variants (EfficientNet-B1 to B7), offering improved accuracy while preserving its computational advantages.

3.5.6 MOBILENET V2

This lightweight convolutional neural network (CNN) architecture is designed for efficiency, making it ideal for devices with limited resources and mobility. A key feature of this architecture is the use of inverted residual blocks with linear bottlenecks, which optimize information flow while keeping computational costs low. Unlike traditional convolutional networks, it incorporates depthwise separable convolutions, which significantly reduce the number of parameters by performing spatial and depthwise operations separately. Additionally, the use of skip connections ensures that essential features are retained while minimizing redundancy, making the network both powerful and efficient. These design choices enable the model to perform complex tasks while requiring minimal computational power, making it highly suitable for real-time applications.

By balancing high accuracy with low resource consumption, this architecture is well-suited for tasks such as object detection, image classification, and facial recognition, especially in mobile and embedded systems. Its ability to perform efficiently embedded systems. Its ability to perform efficiently on edge devices, such as smartphones and IoT hardware, has made it a preferred choice for AI applications that demand both speed and accuracy. Whether used for autonomous navigation, medical diagnostics, or security surveillance, this model ensures reliable performance without the need for high-end GPUs or extensive computational resources. Its compact yet powerful design has contributed to its widespread adoption in modern deep-learning applications.

3.6 CLASSIFIERS

Classifiers are machine learning models used to categorize data into predefined classes, making them essential for tasks like binary or multi-class classification. Each classifier type uses a unique algorithm to learn patterns in the data. Here's a brief overview

3.6.1 SUPPORT VECTOR MACHINE

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

Working of SVM

An SVM model is basically a representation of different classes in a hyper plane in multidimensional space. The hyper plane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyper plane.

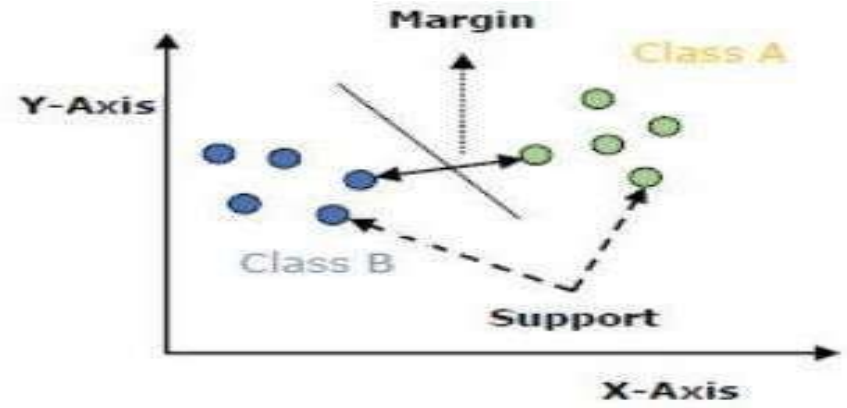


Fig 3.6 –SVM

Support Vectors - Data points that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.

Hyper Plane - As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.

Margin - It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margi

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane and it can be done in the following two steps –

1. SVM will generate hyper planes iteratively that segregates the classes in best way.
2. It will choose the hyper plane that separates the classes correctly.

3.6.2 RANDOM FOREST

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made un of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

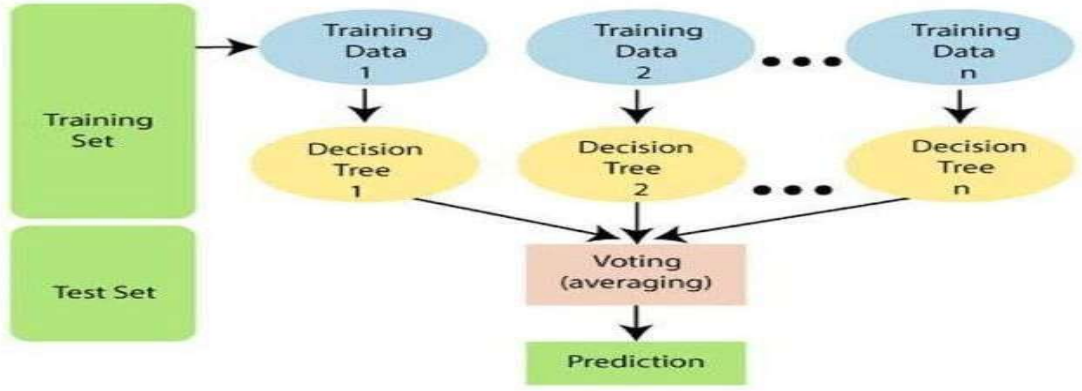


Fig 3.7 – Random Forest

3.6.3 XG- BOOST

XG-Boost is one of the most popular variants of gradient boosting. It is a decision- tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. XG- Boost is basically designed to enhance the performance and speed of a Machine Learning model. In prediction problems involving unstructured data (images, text, etc.), artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small to medium structured/tabular data, decision tree-based algorithms are considered best-in- class right now. XG- Boost uses pre-sorted algorithm & histogram-based algorithm for computing the best split. The histogram-based algorithm splits all the data points for a feature into discrete bins and uses these bins to find the split value of the histogram. Also, in XG-Boost, the trees can have a varying number of terminal nodes and left weights of the trees that are calculated with less evidence is shrunk more heavily.

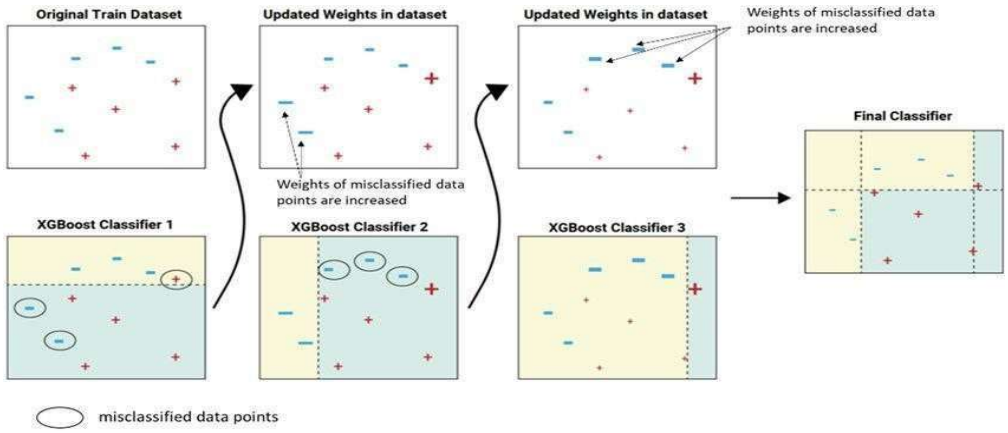


Fig 3.8 – XG Boost

3.6.4 MLP

A Multi-Layer Perceptron (MLP) is a type of artificial neural network that consists of multiple layers of neurons, making it a fundamental building block of deep learning. It is composed of an input layer, one or more hidden layers, and an output layer, where each neuron is fully connected to neurons in the subsequent layer. The MLP operates using feedforward propagation, where input data is passed through the network, and each neuron applies a weighted sum followed by a nonlinear activation function, such as ReLU (Rectified Linear Unit) or Sigmoid, to introduce non-linearity and improve the model's learning capability. During training, the network adjusts its weights using backpropagation and gradient descent, minimizing the error between predicted and actual outputs. This process enables MLPs to learn complex relationships within data, making them highly effective for classification, regression, and pattern recognition tasks.

One of the key advantages of MLPs is their ability to model complex, non-linear relationships, making them suitable for a wide range of applications, including image recognition, speech processing, financial forecasting, and medical diagnosis. However, traditional MLPs can become computationally expensive as the number of hidden layers and neurons increases, leading to challenges such as overfitting and slow convergence. To address these issues, techniques like dropout regularization, batch normalization, and adaptive learning rate optimization (e.g., Adam or RMSprop) are often used to improve stability and generalization. Although MLPs were among the earliest neural network architectures, they have evolved into deeper and more efficient models.

3.7 MODEL EVALUATION

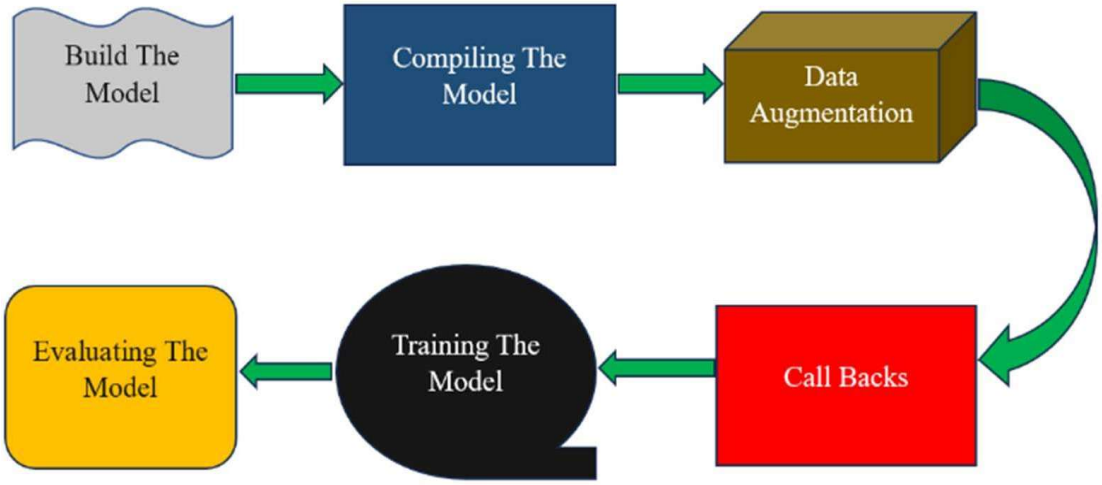


Fig.3.9 Model Development Pipeline

As depicted in Fig. 3.7 the process begins with building the model, followed by compiling it to configure the training settings. Data augmentation techniques are then applied to enhance model generalization and improve robustness against variations in input images. After augmentation, callbacks are implemented to monitor and optimize training, helping to prevent overfitting and improve performance. The model is then trained using the prepared dataset, leveraging augmented data and callbacks for better convergence. Finally, the trained model is evaluated to assess its accuracy and effectiveness in classifying skin lesions. This structured approach ensures the development of a well-optimized and reliable classification model.

Chapter-4

REQUIREMENT ANALYSIS AND TECHNICAL DESCRIPTION

4.1 SOFTWAREDESCRIPTION

4.1.1 Python

Python is a multi-paradigm programming language. Object-orientedprogramming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta programming and meta objects). Many other paradigms are supported via extensions, including design by contract and logic programming.Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds mean variable names during program execution.

Many other paradigms are supported via extensions, including design by contract and logic programming.

It is used for:

- Web development(server-side),
- Software development,
- System scripting.

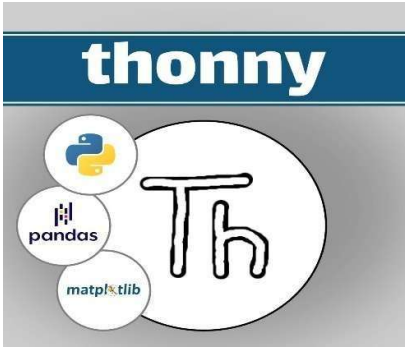


Fig 4.1 Thonny Python

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used along side software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Benefits of Python

- Presence of Third-Party Modules
- Extensive Support Libraries
- Open Source and Community Development
- Learning Ease and Support Available
- User-friendly Data Structures.

4.1.2 Windows

Windows is a desktop operating system developed by Microsoft. For the past three decades, windows have been the most popular operating system for personal computers. Each version of windows comes with a graphical user interface that includes a desktop with icons and a task bar that is displayed at the button of a screen by default. Most windows versions include a start menu, which provides quick access to files, settings and the windows search feature.

Windows runs standard x84 hardware, such as Intel and AMD processors. Software programs written for windows may be called apps, applications, executable files. Regardless of their label windows software programs have an .EXE file extension. 64-bit versions of windows run both 32 and 64-bits apps, while 32-bit versions only run 32-bit applications.

4.1.3 Google Colab

Google Colab is a cloud-based platform provided by Google that allows users to run Python code in a Jupyter notebook environment without the need for any local setup or installation. The platform provides free access to a virtual machine with GPU and TPU support, which can be used to train machine learning models, run deep learning experiments, and perform data analysis tasks.

Google Colab provides users with a number of useful features, such as:

- Free access to GPUs and TPUs for training machine learning models.
- Collaborative editing and sharing of notebooks with other users.
- Integration with popular computer vision libraries and frameworks, including cv2, yolo, Tensor Flow, PyTorch, pandas, e.t.c

One of the main advantages of Google Colab is that it allows users to run resource- intensive computations on powerful virtual machines without incurring the costs of running and maintaining their own hardware. Overall, Google Colab is a valuable tool for data scientists, machine learning engineers, and researchers who want to experiment with machine learning and deep learning techniques without investing in expensive hardware or infrastructure.

4.1.4 Applications of Python

- Web Development.
- Game Development.
- Machine Learning and Artificial Intelligence.
- Data Science and Data Visualization.
- Desktop GUI.
- Web Scraping Applications.
- Business Applications.
- Audio and Video Applications.

4.1.5 Dependent Python Packages

When managing Python environments, one of the key concerns is dependency management. Dependencies are all of the software components required by your project in order for it to work as intended and avoid runtime errors.You can count on PyPI (the Python Package Index) to provide packages that can help you get started on everything from data manipulation to machine learning to web development, and more. The Active State Platform is the only solution that currently supports both:

- Dependency Resolution - automatically ensures that all dependencies pulled in by a package are compatible with the rest of your Python environment.
- Dependency Conflict Resolution - when the Active State Platform cannot automatically resolve dependencies, it will suggest a manual solution to resolve the conflict.

4.1.6 Description of the Libraries

cv2 (OpenCV): OpenCV (Open Source Computer Vision Library) is an open- source computer vision and machine learning software library. It provides various functions for real-time computer vision tasks, such as image processing, object detection, video analysis, and more. OpenCV is written in C++ and has interfaces for Python, Java, and MATLAB. The Python interface is known as cv2. OpenCV is widely used in fields like robotics, augmented reality, facial recognition, and surveillance systems.

Pandas: Pandas is a powerful Python library used for data manipulation and analysis. It provides data structures like Series (1-dimensional labeled array) and Data Frame (2-dimensional labeled data structure) that are highly efficient and easy to use. Pandas allow for data cleaning, reshaping, merging, slicing, and visualization. It's widely used in data science, finance, economics, statistics, and other fields for handling structured data.

SSL (Secure Sockets Layer): SSL (Secure Sockets Layer) is a cryptographic protocol used to establish a secure connection between a client and a server over the internet. It ensures that the data transmitted between the client and the server is encrypted and remains private. SSL has been deprecated in favor of Transport Layer Security (TLS), but the term "SSL" is still commonly used to refer to the encryption protocols used to secure websites and web services.

Ultralytics YOLO: It an efficient tool for professionals working in computer vision and ML that can help create accurate object detection models. Ultralytics YOLO builds upon the original YOLO framework with improvements and optimizations, making it even more versatile and powerful. It provides pre-trained models as well as tools for training custom models on new datasets, allowing users to tailor object detection systems to their specific needs.

Playsound: Play sound is a Python library that allows you to play sound files directly from your Python script without the need for any external players. It provides a simple interface for playing sound files in various formats, such as WAV, MP3, and others.

4.2 REQUIREMENT ANALYSIS

4.2.1 Functional Requirements

A functional requirement defines a function of a system or its component. A function is describing a dataset of inputs, the behavior, and outputs.

Behavioral requirements describe all cases where the system uses the functional requirements and are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the designer's implementation (such as performance requirements, security, or reliability). Functional requirements concern the specific functions delivered by the system.

- The functional requirements of the system should be both complete and consistent.
- Completeness means that all the services required by the user should be defined.
- Consistency means that requirements should not have any contradictory definitions.
- Take user ID and password, match it with corresponding file entries. If a match is found, then continue; else, raise an error message.

4.2.2 Non-Functional Requirements

Non-functional requirements refer to the constraints or restrictions on the system. They may relate to emergent system properties such as reliability, response time, and store occupancy, or the selection of language, platform, implementation techniques, and tools.

Accuracy: The system should have high accuracy in detecting and tracking hurricanes from satellite images to ensure the safety of people in affected areas.

Speed and real-time performance: The system should be able to process satellite images in real-time, to provide timely warnings and responses in case of an impending hurricane. The speed of processing should also be optimized to reduce the processing time and deliver faster results.

Reliability: The system should be reliable and resilient to provide uninterrupted service even in challenging environmental conditions, such as adverse weather or satellite communication disruptions.

Scalability: The system should be able to scale up or down to handle large amounts of data during peak times or events.

Security and privacy: The system should maintain the confidentiality and integrity of the satellite image data, and adhere to privacy regulations when handling sensitive information.

Usability: The system should be user-friendly, with an easy-to-use interface that provides the necessary information to support decision-making during a hurricane event.

Accessibility: Accessibility is a general term used to describe the degree to which a product, device, service, or environment is accessible by as many people as

Maintainability: In software engineering, maintainability is the ease with which a software product can be modified in order to include new functionalities. It can be added to the project based on the user requirements just by adding the appropriate files to the existing project.

Scalability: The system is capable of handling increased total throughput under an increased load when resources (typically hardware) are added.

Portability: Portability is one of the key concepts of high-level programming. These prerequisites are known as (computer) system requirements and are often used as guidelines as opposed to an absolute rule.

4.2.3 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

- Computer/Server (A powerful multicore) CPU : Intel Core i7 (10th Gen) AMD Ryzen 7 (4000 series)
- GPU : NVIDIA GTX 1660 Ti (6GB VRAM) or AMD equivalent
- RAM : 16GB DDR4
- Storage : 512GB SSD + 1TB HDD (for dataset storage)
- OS : Windows 10/11

4.2.4 Software Requirements

Software requirements deal with defining software resource requirements and pre- requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed. The following are used in our project.

- Programming Language: Python(version-3.7.0and above)
- IDLE: GOOGLE COLAB
- Operating system: Any

Chapter – 5

IMPLEMENTATION

Feature_Extraction_Preprocessing.ipynb

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
from glob import glob
import seaborn as sns

file_path = '/content/drive/MyDrive/final_yr_project/HAM10000_metadata.csv'

skin_df = pd.read_csv(file_path)

base_skin_dir = "/content/drive/MyDrive/final_yr_project/img_dataset"
imageid_path_dict = {os.path.splitext(os.path.basename(x))[0]: x
                     for x in glob(os.path.join(base_skin_dir, '*', '*.jpg'))}

lesion_type_dict = {
    'nv': 'Melanocytic_nevi',
    'mel': 'melanoma',
    'bkl': 'Benign_keratosis-like_lesions',
    'bcc': 'Basal_cell_carcinoma',
    'akiec': 'Actinic_keratoses',
    'vasc': 'Vascular_lesions',
    'df': 'Dermatofibroma'
}

lesion_danger = {
    'nv': 0, # 0 for benign
    'mel': 1, # 1 for malignant
    'bkl': 0, # 0 for benign
    'bcc': 1, # 1 for malignant
    'akiec': 1, # 1 for malignant
    'vasc': 0,
    'df': 0
}

skin_df["path"] = skin_df["image_id"].apply(lambda x: os.path.join(base_skin_dir, x
+ ".jpg"))
skin_df["cell_type"] = skin_df["dx"].map(lesion_type_dict.get)
skin_df["Malignant"] = skin_df["dx"].map(lesion_danger.get)
skin_df["cell_type_idx"] = pd.Categorical(skin_df["cell_type"]).codes
from skimage.io import imread
```

```
skin_df["image"] = skin_df["path"].map(imread)
skin_df.iloc[0]["image"]
skin_df["image"].map(lambda x: x.shape).value_counts()

# let's have a look at the image data

n_samples = 5 # choose 5 samples for each cell type
fig, m_axs = plt.subplots(7, n_samples, figsize=(4*n_samples, 3 * 7))

for n_axs, (type_name, type_rows) in zip(m_axs,
skin_df.sort_values(["cell_type"]).groupby("cell_type")):
   n_axs[0].set_title(type_name)
   for c_ax, (_, c_row) in zip(n_axs, type_rows.sample(n_samples,
random_state=0).iterrows()):
      c_ax.imshow(c_row["image"])
      c_ax.axis("off")
# fig.savefig("category_samples.png", dpi=300)

rgb_info_df = skin_df.apply(lambda x: pd.Series({'
{}_mean'.format(k): v for k, v in zip(["Red", "Blue", "Green"],
         np.mean(x["image"], (0, 1)))}), 1)


gray_col_vec = rgb_info_df.apply(lambda x: np.mean(x), 1) # take the mean value
across columns of rgb_info_df
for c_col in rgb_info_df.columns:
   rgb_info_df[c_col] = rgb_info_df[c_col]/gray_col_vec
rgb_info_df["Gray_mean"] = gray_col_vec
rgb_info_df.sample(3)
for c_col in rgb_info_df.columns:
   skin_df[c_col] = rgb_info_df[c_col].values

sns.pairplot(skin_df[["Red_mean", "Green_mean", "Blue_mean", "Gray_mean",
"cell_type"]],
          hue="cell_type", plot_kws = {"alpha": 0.5})
n_samples = 5
for sample_col in ["Red_mean", "Green_mean", "Blue_mean", "Gray_mean"]:
   fig, m_axs = plt.subplots(7, n_samples, figsize=(4 * n_samples, 3 * 7))
   fig.suptitle(f"Change in cell type appearance as {sample_col} change")

   def take_n_space(in_rows, val_col, n):
      s_rows = in_rows.sort_values([val_col])
      s_idx = np.linspace(0, s_rows.shape[0] - 1, n, dtype=int)
      return s_rows.iloc[s_idx]

   for n_axs, (type_name, type_rows) in zip(m_axs,
skin_df.sort_values(["cell_type"]).groupby("cell_type")):
      for c_ax, (_, c_row) in zip(n_axs, take_n_space(type_rows, sample_col,
n_samples).iterrows()):
```

```
            c_ax.imshow(c_row["image"])
            c_ax.axis("off")
            c_ax.set_title('{:2.2f}'.format(c_row[sample_col]))
        n_axs[0].set_title(type_name)
    fig.savefig("{}_samples.png".format(sample_col), dpi=300)
from PIL import Image
reshaped_image = skin_df["path"].map(lambda x:
np.asarray(Image.open(x).resize((64,64), resample=Image.LANCZOS).\
                                            convert("RGB")).ravel())
out_vec = np.stack(reshaped_image, 0)
out_df = pd.DataFrame(out_vec)

out_df["label"] = skin_df["Malignant"].values
out_path = "hmnist_64_64_RBG.csv"
out_df.to_csv(out_path, index=False)
```

Final_Preprocessing.ipynb

```
data = pd.read_csv('/content/drive/MyDrive/Skin lesion detection and
classification/hmnist_64_64_RBG.csv')

label_column = 'label'
features = data.drop(columns=[label_column])
features = features.apply(lambda col: col.mask(col == 0, col.mean()), axis=0)
preprocessed_data = pd.concat([features, data[label_column]], axis=1)

label_column = 'label'  # Change this to match the actual name of your label column

# Check and print the count of zeros in each column before preprocessing
zero_counts = (data == 0).sum()
print("Count of zeros in each column before preprocessing:")
print(zero_counts)

# Separate features and label
features = data.drop(columns=[label_column])

# Replace zeros with column means in the feature columns
features = features.apply(lambda col: col.mask(col == 0, col.mean()), axis=0)

# Recombine the features and label
preprocessed_data = pd.concat([features, data[label_column]], axis=1)

zero_counts_after = (preprocessed_data == 0).sum()

from imblearn.under_sampling import RandomUnderSampler
undersampler = RandomUnderSampler(random_state=42)
features_resampled, labels_resampled = undersampler.fit_resample(features, labels)

# Apply SMOTE to balance the dataset without reducing the total number of records
```

```
smote = SMOTE(random_state=42, sampling_strategy='auto')
features_resampled, labels_resampled = smote.fit_resample(features, labels)

# Randomly sample the majority class to match the original dataset size
resampled_data = pd.concat([features_resampled, labels_resampled],
axis=1).sample(n=len(data), random_state=42)

# Check new label distribution
balanced_label_counts = resampled_data[label_column].value_counts()
print("Balanced label distribution after applying SMOTE and resampling to maintain
size:")
print(balanced_label_counts)

# Replace zeros with column means in the resampled feature columns
features_resampled = resampled_data.drop(columns=[label_column]).apply(lambda
col: col.mask(col == 0, col.mean()), axis=0)
labels_resampled = resampled_data[label_column]

# Recombine the resampled features and labels
balanced_data = pd.concat([features_resampled, labels_resampled], axis=1)

# # Save the balanced dataset to a new file
balanced_data.to_csv('undersampled_balanced_ham10000.csv', index=False)
```

MOBILNET_v2.ipynb

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping,
Callback
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
# Load your dataset (ensure this path is correct)
data =
pd.read_csv('/content/drive/MyDrive/skin_lesion/undersampled_balanced_ham10000.
csv')

# Assuming the last column is the label for binary classification
X = data.iloc[:, :-1].values  # Features
```

```
y = data.iloc[:, -1].values  # Labels
# Rescale and reshape the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Reshape the data for CNN input (Ensure correct shape for MobileNetV2)
X_reshaped = X_scaled.reshape(-1, 64, 64, 3)

# Resize images to at least 96x96 pixels for MobileNetV2
X_resized = tf.image.resize(X_reshaped, [96, 96]).numpy()

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_resized, y, test_size=0.2,
random_state=42)
# Load MobileNetV2 model pre-trained on ImageNet (exclude the top layers)
mobilenet_base = MobileNetV2(weights='imagenet', include_top=False,
input_shape=(96, 96, 3))

# Unfreeze some layers of the MobileNetV2 model for fine-tuning (consider
unfreezing more layers)
for layer in mobilenet_base.layers[-20:]:  # Unfreeze the last 20 layers for fine-tuning
    layer.trainable = True

from tensorflow.keras.regularizers import l2  # Import L2 regularization

# Build the model using Sequential with additional regularization
model = Sequential([
    mobilenet_base,
    GlobalAveragePooling2D(),  # Global pooling to reduce spatial dimensions
    Dense(128, activation='relu', kernel_regularizer=l2(0.001)),  # L2 regularization
added
    Dropout(0.5),  # Dropout to avoid overfitting
    Dense(1, activation='sigmoid', kernel_regularizer=l2(0.001))  # L2 regularization
on the output layer
])
# Compile the model with the same optimizer
model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy',
metrics=['accuracy'])
# Define Data Generator for Data Augmentation with enhanced transformations
datagen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
```

```
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Fit the data generator to the training data
datagen.fit(X_train)
#Define Learning Rate Reduction and Early Stopping for better convergence
lr_reduction = ReduceLROnPlateau(monitor='val_loss', patience=3, factor=0.5,
min_lr=0.001, verbose=1)
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

class MetricsCallback(Callback):
    def on_epoch_end(self, epoch, logs=None):
        # Get predictions on validation data
        y_pred = (self.model.predict(self.validation_data[0]) > 0.5).astype("int32")
        y_true = self.validation_data[1]

        # Print classification report metrics: precision, recall, f1-score
        report = classification_report(y_true, y_pred, output_dict=True,zero_division=0)
        print(f"Epoch {epoch + 1} Classification Report:")
        print(f"Precision: {report['1']['precision']:.4f}, Recall: {report['1']['recall']:.4f},
F1-Score: {report['1']['f1-score']:.4f}")

# Create an instance of the MetricsCallback and set validation data for it
metrics_callback = MetricsCallback()
metrics_callback.validation_data = (X_test, y_test)

# Train the model with augmented data and callbacks
batch_size = 32  # Experiment with this value if needed; smaller sizes can improve
generalization.
epochs = 30  # Increased epochs for better model training

history = model.fit(
    datagen.flow(X_train, y_train, batch_size=batch_size),
    validation_data=(X_test, y_test),
    epochs=epochs,
    callbacks=[lr_reduction, early_stopping, metrics_callback]
)
```

```
# Evaluate the model on the test set and print results
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
model.summary()
model.save("FINAL_MOBILE NET_model.h5")
model.save('FINAL_MOBILE NET_model.keras')
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import precision_score, recall_score, f1_score,
confusion_matrix, accuracy_score

y_train_pred = model.predict(X_train)
y_train_pred_classes = (y_train_pred > 0.5).astype(int) # Convert probabilities to
class labels (0 or 1)
#y_train_true_classes = np.argmax(y_train, axis=1) # This line is removed
y_train_true_classes = y_train # y_train already contains the true class labels

y_test_pred = model.predict(X_test)
y_test_pred_classes = (y_test_pred > 0.5).astype(int) # Convert probabilities to class
labels (0 or 1)
#y_test_true_classes = np.argmax(y_test, axis=1) # This line is removed
y_test_true_classes = y_test # y_test already contains the true class labels

# Metrics
train_precision = precision_score(y_train_true_classes, y_train_pred_classes,
average='weighted', zero_division=0)
train_recall = recall_score(y_train_true_classes, y_train_pred_classes,
average='weighted', zero_division=0)
train_f1 = f1_score(y_train_true_classes, y_train_pred_classes, average='weighted')
train_accuracy = accuracy_score(y_train_true_classes, y_train_pred_classes)

test_precision = precision_score(y_test_true_classes, y_test_pred_classes,
average='weighted', zero_division=0)
test_recall = recall_score(y_test_true_classes, y_test_pred_classes,
average='weighted', zero_division=0)
test_f1 = f1_score(y_test_true_classes, y_test_pred_classes, average='weighted')
test_accuracy = accuracy_score(y_test_true_classes, y_test_pred_classes)

# Print Metrics
print("Training Metrics:")
print(f" Accuracy: {train_accuracy:.4f},Precision: {train_precision:.4f}, Recall:
{train_recall:.4f}, F1-Score: {train_f1:.4f}")
```

```
print("\nTesting Metrics:")
print(f" Accuracy: {test_accuracy:.4f},Precision: {test_precision:.4f}, Recall:
{test_recall:.4f}, F1-Score: {test_f1:.4f}")
# Prepare data for plotting
metrics = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
train_values = [train_accuracy, train_precision, train_recall, train_f1]
test_values = [test_accuracy, test_precision, test_recall, test_f1]

# Set up the bar plot
width = 0.35  # Width of the bars
x = np.arange(len(metrics)) # The label locations

fig, ax = plt.subplots(figsize=(10, 6))

# Plot the bars for training and testing metrics
rects1 = ax.bar(x - width/2, train_values, width, label='Train', color='darkorange')
rects2 = ax.bar(x + width/2, test_values, width, label='Test', color='yellow')

# Add some text for labels, title and custom x-axis tick labels
ax.set_ylabel('Scores')
ax.set_title('Comparison of Metrics for Training and Testing')
ax.set_xticks(x)
ax.set_xticklabels(metrics)
ax.legend()

# Function to add the labels on top of the bars
def add_labels(rects):
    for rect in rects:
        height = rect.get_height()
        ax.annotate(f'{height:.2f}',
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')

# Add labels for each bar
add_labels(rects1)
add_labels(rects2)

# Show the plot
plt.tight_layout()
plt.show()
```

```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(X_test, y_test)

# Predict probabilities and binary classes
y_pred_probs = model.predict(X_test).flatten()
y_pred = (y_pred_probs > 0.5).astype(int)

# Classification Report
print(classification_report(y_test, y_pred))

from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Predict probabilities for the test set
y_pred_probs = model.predict(X_test).flatten()

# Compute false positive rate, true positive rate, and thresholds
fpr, tpr, thresholds = roc_curve(y_test, y_pred_probs)

# Calculate AUC
roc_auc = auc(fpr, tpr)

# Plot the AUC-ROC curve
plt.figure(figsize=(10, 6))
plt.plot(fpr, tpr, color='blue', label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label='Random Guess')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid()
plt.show()
```

EfficientNet_B0.ipynb

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping,
Callback
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf

# Load your dataset (ensure this path is correct)
data = pd.read_csv('/content/drive/MyDrive/hmnist_64_64_RBG.csv')

# Assuming the last column is the label for binary classification
X = data.iloc[:, :-1].values  # Features
y = data.iloc[:, -1].values   # Labels

# Rescale and reshape the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Reshape the data for CNN input (Ensure correct shape for EfficientNetB0)
X_reshaped = X_scaled.reshape(-1, 64, 64, 3)

# Resize images to 75x75 pixels for EfficientNetB0
X_resized = tf.image.resize(X_reshaped, [75, 75]).numpy()

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_resized, y, test_size=0.2,
random_state=42)

# Load EfficientNetB0 model pre-trained on ImageNet (exclude the top layers)
efficientnet_base = EfficientNetB0(weights='imagenet', include_top=False,
input_shape=(75, 75, 3))

# Unfreeze some layers of the EfficientNetB0 model for fine-tuning
for layer in efficientnet_base.layers[-20:]:  # Unfreeze the last 20 layers for fine-
tuning
    layer.trainable = True

# Build the model
```

```
model = Sequential([
    efficientnet_base,
    GlobalAveragePooling2D(),  # Global pooling to reduce spatial dimensions
    Dense(128, activation='relu'),  # Fully connected layer with ReLU activation
    Dropout(0.5),  # Dropout to avoid overfitting
    Dense(1, activation='sigmoid')  # Binary classification output layer
])

# Compile the model with a lower initial learning rate for fine-tuning
model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy',
metrics=['accuracy'])

# Define Data Generator for Data Augmentation with enhanced transformations
datagen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Fit the data generator to the training data
datagen.fit(X_train)

# Define Learning Rate Reduction and Early Stopping
lr_reduction = ReduceLROnPlateau(monitor='val_loss', patience=3, factor=0.5,
min_lr=0.00001, verbose=1)
early_stopping = EarlyStopping(monitor='val_loss', patience=3,
restore_best_weights=True)

# Custom Callback for Metrics
class MetricsCallback(Callback):
    def on_epoch_end(self, epoch, logs=None):
        y_pred = (self.model.predict(self.validation_data[0]) > 0.5).astype("int32")
        y_true = self.validation_data[1]
        report = classification_report(y_true, y_pred, output_dict=True)
        print(f"Epoch {epoch + 1} Classification Report:")
        print(f"Precision: {report['1']['precision']:.4f}, Recall: {report['1']['recall']:.4f},
F1-Score: {report['1']['f1-score']:.4f}")

# Create an instance of the MetricsCallback
metrics_callback = MetricsCallback()
metrics_callback.validation_data = (X_test, y_test)

# Train the model with augmented data and callbacks
batch_size = 32
epochs = 30
```

```
history = model.fit(
    datagen.flow(X_train, y_train, batch_size=batch_size),
    validation_data=(X_test, y_test),
    epochs=epochs,
    callbacks=[lr_reduction, early_stopping, metrics_callback]
)

# Print training accuracy from history object after training is complete.
train_accuracy = history.history['accuracy'][-1]
print(f"Training Accuracy: {train_accuracy * 100:.2f}%")

# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")

model.summary()
model.save("Efficient Net_model.h5")
model.save('Efficient Net_model.keras')
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import precision_score, recall_score, f1_score,
confusion_matrix, accuracy_score

y_train_pred = model.predict(X_train)
y_train_pred_classes = (y_train_pred > 0.5).astype(int) # Convert probabilities to
class labels (0 or 1)
#y_train_true_classes = np.argmax(y_train, axis=1) # This line is removed
y_train_true_classes = y_train # y_train already contains the true class labels

y_test_pred = model.predict(X_test)
y_test_pred_classes = (y_test_pred > 0.5).astype(int) # Convert probabilities to class
labels (0 or 1)
#y_test_true_classes = np.argmax(y_test, axis=1) # This line is removed
y_test_true_classes = y_test # y_test already contains the true class labels

# Metrics
train_precision = precision_score(y_train_true_classes, y_train_pred_classes,
average='weighted', zero_division=0)
train_recall = recall_score(y_train_true_classes, y_train_pred_classes,
average='weighted', zero_division=0)
train_f1 = f1_score(y_train_true_classes, y_train_pred_classes, average='weighted')
# train_accuracy = accuracy_score(y_train_true_classes, y_train_pred_classes)

test_precision = precision_score(y_test_true_classes, y_test_pred_classes,
average='weighted', zero_division=0)
test_recall = recall_score(y_test_true_classes, y_test_pred_classes,
average='weighted', zero_division=0)
```

```
test_f1 = f1_score(y_test_true_classes, y_test_pred_classes, average='weighted')
# test_accuracy = accuracy_score(y_test_true_classes, y_test_pred_classes)

# Print Metrics
print("Training Metrics:")
print(f"Precision: {train_precision:.4f}, Recall: {train_recall:.4f}, F1-Score:
{train_f1:.4f}")

print("\nTesting Metrics:")
print(f"Precision: {test_precision:.4f}, Recall: {test_recall:.4f}, F1-Score:
{test_f1:.4f}")
# Prepare data for plotting
metrics = ['Precision', 'Recall', 'F1 Score']
train_values = [train_precision, train_recall, train_f1]
test_values = [test_precision, test_recall, test_f1]

# Set up the bar plot
width = 0.35  # Width of the bars
x = np.arange(len(metrics))  # The label locations

fig, ax = plt.subplots(figsize=(10, 6))

# Plot the bars for training and testing metrics
rects1 = ax.bar(x - width/2, train_values, width, label='Train', color='red')
rects2 = ax.bar(x + width/2, test_values, width, label='Test', color='yellow')

# Add some text for labels, title and custom x-axis tick labels
ax.set_ylabel('Scores')
ax.set_title('Comparison of Metrics for Training and Testing')
ax.set_xticks(x)
ax.set_xticklabels(metrics)
ax.legend()

# Function to add the labels on top of the bars
def add_labels(rects):
    for rect in rects:
        height = rect.get_height()
        ax.annotate(f'{height:.2f}',
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')

# Add labels for each bar
add_labels(rects1)
add_labels(rects2)

# Show the plot
plt.tight_layout()
```

```
plt.show()
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(X_test, y_test)

# Predict probabilities and binary classes
y_pred_probs = model.predict(X_test).flatten()
y_pred = (y_pred_probs > 0.5).astype(int)

# Classification Report
print(classification_report(y_test, y_pred))

from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Predict probabilities for the test set
y_pred_probs = model.predict(X_test).flatten()

# Compute false positive rate, true positive rate, and thresholds
fpr, tpr, thresholds = roc_curve(y_test, y_pred_probs)

# Calculate AUC
roc_auc = auc(fpr, tpr)

# Plot the AUC-ROC curve
plt.figure(figsize=(10, 6))
plt.plot(fpr, tpr, color='blue', label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label='Random Guess')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid()
plt.show()

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Predict the test set
y_pred = (model.predict(X_test) > 0.5).astype(int)

# Compute the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Display the confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Class 0',
'Class 1'])
disp.plot(cmap='Blues', values_format='d')
plt.title("Confusion Matrix")
plt.show()
```

Chapter – 6

RESULTS ANALYSIS

This project tracks various performance metrics related to the classification task, including training accuracy, test accuracy, precision, recall, F1-score, and area under the receiver operat ing characteristic (ROC) curve, in order to compare six distinct Deep Learning Models: Incep tionv3, ResNet50, Dense Net 201, VGG19, Mobile NetV2, and Efficient NetB0. While test ac curacy refers to the model's performance on unseen data, training accuracy relates to the model's performance on training data. The model's precision, which measures how well it predicts posi tive classes, is defined as the ratio of real positive predictions to all cases projected as positive. In contrast to accuracy, recall indicates the model's capacity to identify positives by calculating the ratio of genuine positive predictions to all actual positive occurrences. In order to balance precision and recall, particularly when the class distribution is unbalanced, the F1-score inte grates the two metrics into a single statistic. AUC-ROC is a crucial metric for assessing the model's capacity to differentiate between positive and negative classes for various thresholds

6.1 TRAINING AND MODEL PERFORMANCE:

TABLE 1: CNN MODELS PERFORMANCE METRICS

Model	Training Accuracy(%)	Test Accuracy(%)	Precision (%)	Recall (%)	F1Score (%)	AUC-ROC (%)
InceptionV3	96.73	88.87	96.75	96.73	96.73	0.96
ResNet50	73.29	71.89	74.59	73.29	72.98	0.72
DenseNet201	88.09	85.93	82.00	81.23	81.14	0.90
MobileNetV2	98.22	91.11	98.22	98.22	98.21	0.97
EfficientNetB0	97.79	90.21	97.81	97.79	97.79	0.97
VGG19	79.59	78.63	81.41	79.59	79.32	0.92

Table 1 presents the performance metrics of six different Convolutional Neural Network (CNN) architectures—InceptionV3, ResNet50, DenseNet201, MobileNetV2, EfficientNetB0, and VGG19 evaluated on training and test datasets. Among the models, EfficientNetB0 achieves the highest test accuracy (91.81%) and F1-score (0.97), indicating strong generalization and balanced performance across all metrics. MobileNetV2 also performs competitively, with a test accuracy of 91.11% and an F1-score of 0.91. In contrast, ResNet50 and VGG19 demonstrate comparatively lower performance, particularly in test accuracy and AUC-ROC, suggesting potential overfitting or less effective feature extraction. Overall, the results highlight EfficientNetB0 as the most robust CNN architecture for the given task, followed closely by MobileNetV2.

TABLE 2: CNN MODELS + CLASSIFIERS PERFORMANCE METRICS

Model	Accuracy(%)	AUC-ROC (%)
InceptionV3+SVC	76.88	0.84
InceptionV3+ RF	74.19	0.83
InceptionV3+XGBoost	75.14	0.84
InceptionV3+ MLP	77	0.85
ResNet50+SVM	69	0.74
ResNet50+RF	76	0.85
ResNet50+ XGBoost	76	0.84
ResNet50+MLP	72	0.83
DenseNet201+SVC	82.63	0.90
DenseNet201+RF	83.52	0.91
DenseNet201+XGBoost	83.62	0.91
DenseNet201+MLP	85.42	0.92
MobileNetV2+SVC	80.38	0.87
MobileNetV2+RF	78.08	0.87
MobileNetV2+XGBoost	78.53	0.88
MobileNetV2+MLP	81	0.88
EfficientNetB0+SVC	69.06	0.71
EfficientNetB0+RF	79.38	0.88
EfficientNetB0+XGBoost	80.43	0.89
EfficientNetB0+MLP	66.25	0.69

Table 2 evaluates the classification performance of various CNN architectures when combined with traditional machine learning classifiers such as SVC, Random Forest (RF), XGBoost, Multi-Layer Perceptron (MLP), and Support Vector Machines (SVM). The models are assessed using Accuracy and AUC-ROC metrics. Among the combinations, DenseNet201 with MLP and DenseNet201 with XGBoost achieved the highest performance, with accuracies of 85.42% and 83.62%, and AUC-ROC scores of 0.92 and 0.91 respectively. EfficientNetB0 with RF also performed well (Accuracy: 80.56%, AUC-ROC: 0.90). On the other hand, EfficientNetB0 with MLP yielded the lowest performance (Accuracy: 66.25%, AUC-ROC: 0.69), suggesting that certain classifier combinations may not generalize effectively. Overall, DenseNet201 emerges as the most effective feature extractor when paired with advanced classifiers like MLP and XGBoost.

TABLE 3: STACKING MODELS PERFORMANCE METRICS

Model	Testing Accuracy(%)	AUC-ROC (%)
InceptionV3+ MobileNetV2+ EfficientNetBo+SVM	84.02	0.87
InceptionV3+MobileNetV2+ EfficientNet Bo+ RF	83.57	0.91
InceptionV3+MobileNetV2+ EfficientNet Bo+ XG Boost	83.33	0.92
InceptionV3+MobileNetV2 +EfficientNetBo+ LogisticRegression	83.67	0.92
InceptionV3+MobileNetV2+ EfficientNet Bo+ MLP	84.02	0.92

Table 3 presents the evaluation of various stacking models created by combining predictions from multiple CNN architectures and classifiers. The performance is measured using Testing Accuracy and AUC-ROC. The combination of InceptionV3 + MobileNetV2+ + EfficientNetB0 yielded the highest AUC-ROC score of 0.92, with a strong testing accuracy of 84.02%, highlighting the effectiveness of ensemble learning. Similarly, InceptionV3 + MobileNetV2+ and EfficientNetB0 + XGBoost also achieved an AUC-ROC of 0.92, indicating that integrating diverse models can enhance predictive power. On the other hand, although EfficientNetB0 + MLP attained the lowest accuracy (83.02%), it still maintained a decent AUC-ROC (0.90). Overall, stacking models demonstrate improved generalization and discrimination capability over individual models, with combinations involving EfficientNetB0 consistently delivering high performance.

Theoretically, these results support the concepts of transfer learning, showing that models that have already been trained on extensive datasets (ImageNet) can be effectively adjusted for use in medical imaging applications. The results further highlight the importance of selecting the appropriate CNN architecture, as deeper models.

Key findings:
In comparison to hybrid models, MobileNetV2 demonstrated superior performance as a stand-alone resilient model, achieving an impressive 91.11% test accuracy in the HAM10000 dataset.

1. **Addressing Class Imbalance:** The work successfully reduced class imbalance problems by implementing SMOTE and undersampling, enhancing the model's resilience in identifying uncommon lesion types.

2. **Hybrid Model Perspectives:** The comparison with hybrid models demonstrates that, although integrating CNNs with machine learning classifiers can improve performance, when correctly tailored, well-trained solo models such as MobileNetV2 can outperform complicated hybrids if they are properly tuned.

6.2 AUC – ROC Curves

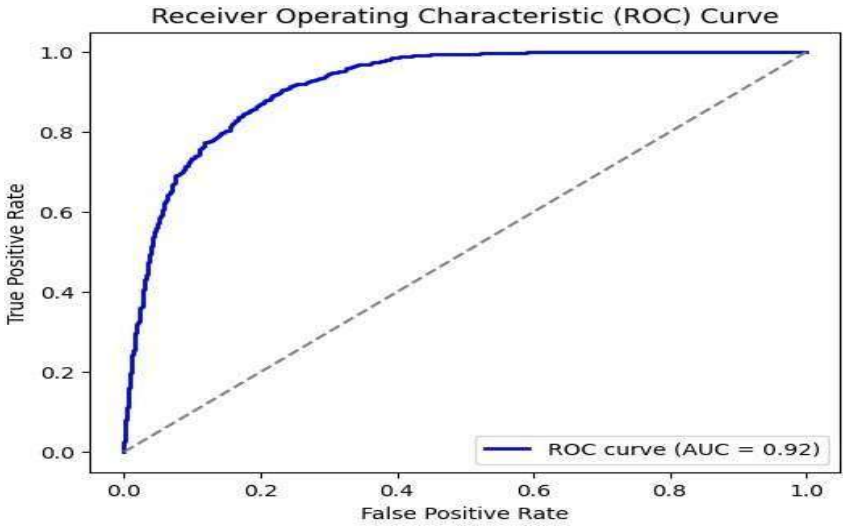


Fig 6.2 - VGG19+XG Boost Classifier

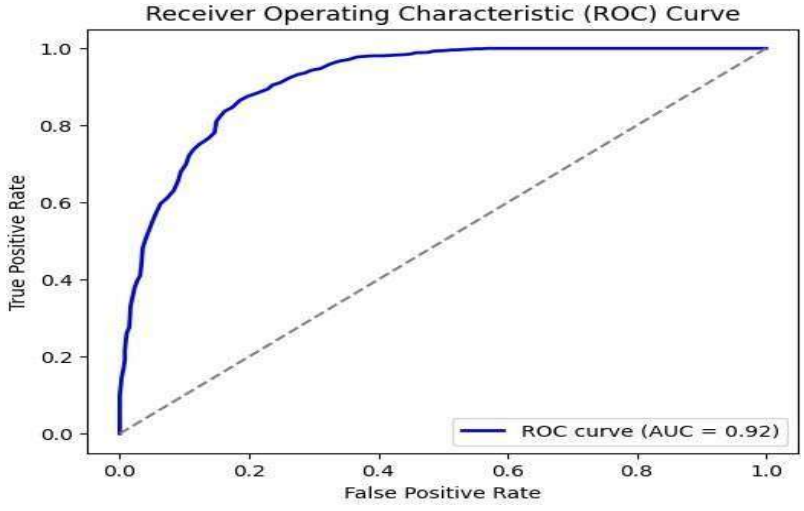


Fig 6.3- VGG19+Random Forest Classifier

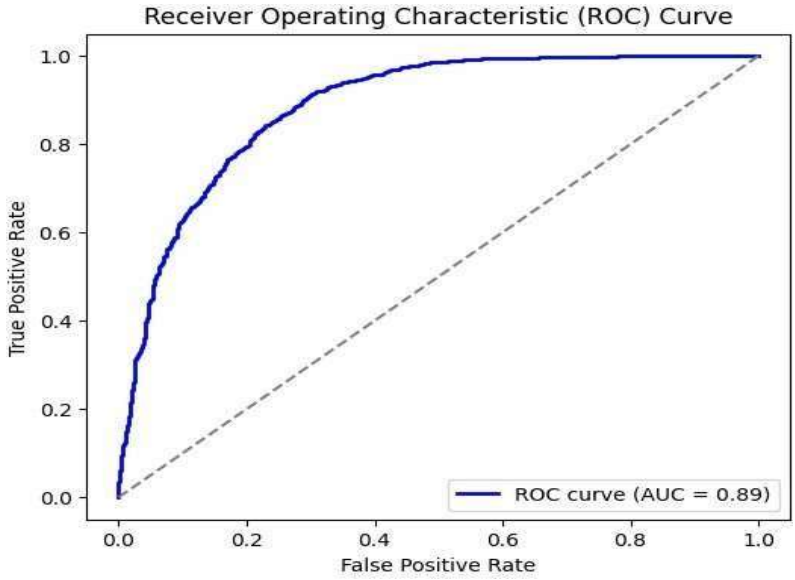


Fig 6.4 - EfficientNet-Bo +XG Boost Classifier

Comparision with previous Research with our findings:

- While DenseNet169 and ResNet50 were proven to be effective in studies like Gururaj et al. (2023) and Kondaveeti et al. (2020), MobileNetV2 and EfficientNetB0 outperformed them in our investigation. This emphasises the necessity of experimenting with various architectures because preprocessing methods, fine-tuning tactics, and dataset size can all affect optimal performance.

- In contrast to Mahbod et al. (2019), who discovered that a hybrid CNN- SVM strategy obtained an AUC of 90.69%. This study's MobileNetV2 model (AUC= 0.97) shows that a standalone CNN can achieve equivalent or superior performance without additional classifiers.

- As with previous research, this study also demonstrates that CNN-based transfer learning models perform better for skin lesion categorisation than conventional machine learning techniques.

Chapter-7

CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION:

This study investigated the efficacy of transfer learning models for skin lesion classification Using architectures like VGG19, InceptionV3, Efficient-NetB0, DenseNet201, MobileNetV2, and ResNet50, as well as hybrid approaches combining classifiers like XGBoost, Random Forest, and SVM.The results showed significant insights into model performance on the HAM10000 dataset. MobileNetV2, a strong standalone model, outperformed all other models, including VGG 19, DensNet 201, and Resnet 50,Inception V3,Efficient Net B0 and both hybrid models with classifiers, with an exceptional test accuracy of 91.11%. The performance of MobileNetV2 and EfficientNetB0 are closely identical, demonstrating the potential of how well they classify medical images. To improve clinical applicability, future research should concentrate on improving hybrid models, incorporating and tackling issues like dataset generalization and ethical considerations when implementing AI for medical diagnosis. Additional hyperparameter adjustment and explainable AI methods like GRAD-CAM may improve the interpretability of the model to highlight influential image regions in deriving decisions. Before classifying, lesion localization could be improved by using segmentation models such as Deep Metric Learning Enhanced Neural Networks (DMLEN).

FUTURE ENHANCEMENT:

The future enhancement will focus on Additional hyperparameter adjustment and explainable AI methods like GRAD-CAM may improve the interpretability of the model to highlight influential image regions in deriving decisions.Before classifying, lesion localization could be improved by using segmentation models such as Deep Metric Learning Enhanced Neural Networks (DMLEN).

REFERENCES

1. Gururaj Harinahalli Lokesh, et al. "DeepSkin: a deep learning approach for skin cancer classification." IEEE Access 11 (2023): 50205-50214.

2. Mahdiraji, S. A., Baleghi, Y., & Sakhaei, S. M. (2017, April). Skin lesion images classification using new color pigmented boundary descriptors. In 2017 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA) (pp. 102-107). IEEE.

3. Hegde P. R., Shenoy M. M., & Shekar, B. H. (2018, September). Comparison of machine learning algorithms for skin disease classification using color and texture features. In 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 1825-1828). IEEE.

4. Hameed N., Shabut A. M., & Hossain M. A. (2018, December). Multi-class skin disease classification using deep convolutional neural network and support vector machine. In 2018, the 12th International Conference on Software, Knowledge, Information Management & Applications (SKIMA) (pp. 1-7). IEEE.

5. Liu X., Hu G., Ma X., & Kuang H. (2019, June). An enhanced neural network based on deep metric learning for skin lesion segmentation. In 2019 Chinese Control And Decision Conference (CCDC) (pp. 1633-1638). IEEE.

6. Mahbod, A., Schaefer, G., Wang, C., Ecker, R., & Ellinge, I. (2019, May). Skin lesion classification using hybrid deep neural networks. In ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP) (pp. 1229-1233).

7. Kondaveeti H. K., & Edupuganti, P. (2020, December). Skin cancer classification using transfer learning. In 2020 IEEE International Conference on Advent Trends in Multidisciplinary Research and Innovation (ICATMRI) (pp. 1-4). IEEE.

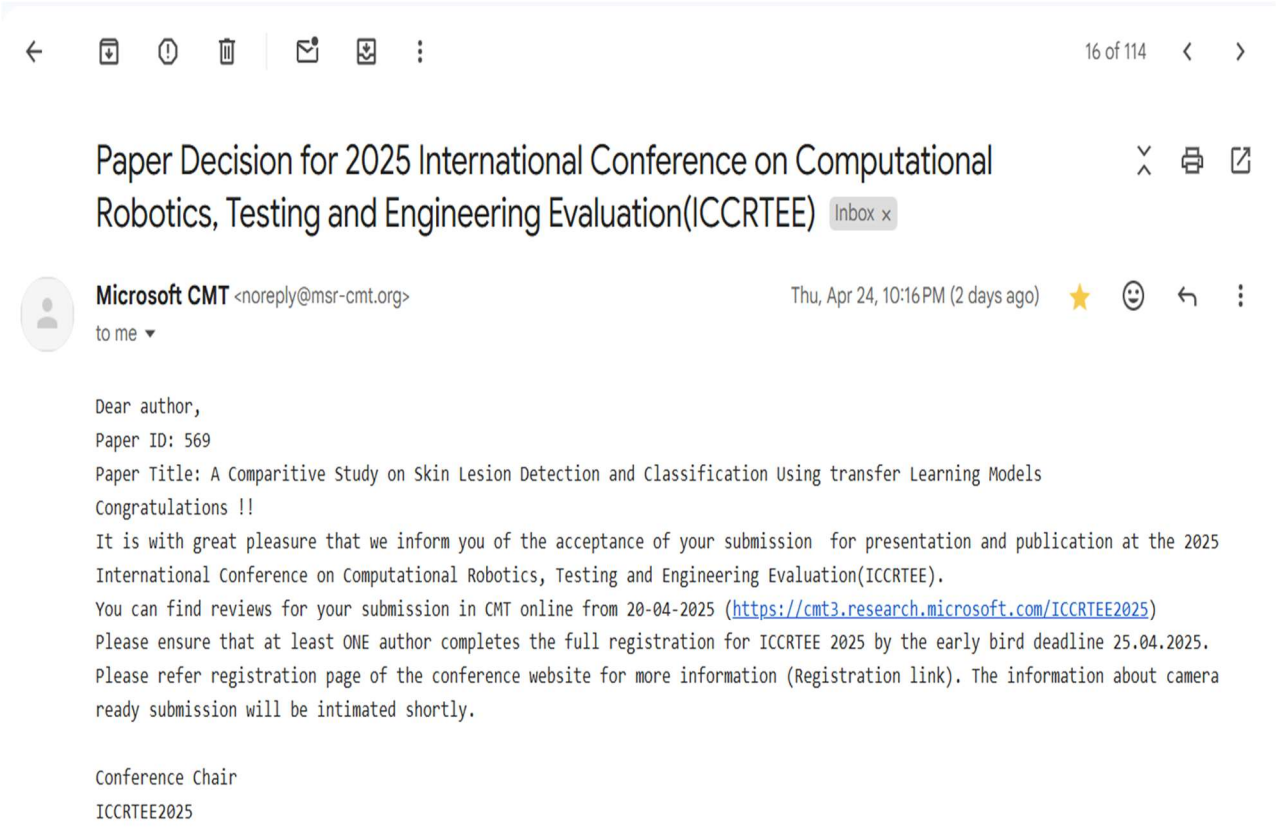
8. Ali Ahmed S. A., Yanıkoğlu B., Gabay Ö., & Aptoula E. (2020). Skin lesion classification with deep CNN ensembles.

9. Raut, R., Borole, Y., Patil, S., Khan, V. N., & Takale, D. G. (2022). Skin disease classification using machine learning algorithms. NeuroQuantology, 20(10), 9624-9629.

10. Debelee, T. G. (2023). Skin lesion classification and detection using machine learning techniques: a systematic review. Diagnostics, 13(19), 3147.

DISSEMINATION OF WORK

1. **Jada viswa Teja, Murakala Prasanthi, Banapuram Hemasundara rao,Arangi vikas,Chelluru pavan kumar (2025) INTERNATIONAL CONFERENCE ON COMPUTATIONAL ROBOTICS, TESTING AND ENGINEERING EVALUATION (ICCRTEE)** (Communicated).



16 of 114

Paper Decision for 2025 International Conference on Computational Robotics, Testing and Engineering Evaluation(ICCRTEE) Inbox ×

Microsoft CMT <noreply@msr-cmt.org> Thu, Apr 24, 10:16 PM (2 days ago)

to me

Dear author,

Paper ID: 569

Paper Title: A Comparitive Study on Skin Lesion Detection and Classification Using transfer Learning Models

Congratulations !!

It is with great pleasure that we inform you of the acceptance of your submission for presentation and publication at the 2025 International Conference on Computational Robotics, Testing and Engineering Evaluation(ICCRTEE).

You can find reviews for your submission in CMT online from 20-04-2025 (https://cmt3.research.microsoft.com/ICCRTEE2025)

Please ensure that at least ONE author completes the full registration for ICCRTEE 2025 by the early bird deadline 25.04.2025.

Please refer registration page of the conference website for more information (Registration link). The information about camera ready submission will be intimated shortly.

Conference Chair

ICCRTEE2025

II NPTEL Certificates

Elite

NPTEL ONLINE CERTIFICATION

(Funded by the MoE, Govt. of India)

Skill India
कौशल भारत-कुशल भारत

This certificate is awarded to

JADA VISWA TEJA

for successfully completing the course

Cloud Computing and Distributed Systems

with a consolidated score of 78 %

Online Assignments	22.08/25	Proctored Exam	55.5/75

Total number of candidates certified in this course: 4408

Prof. B. V. Ratish Kumar
Chairman, Centre for Continuing Education
IIT Kanpur

Jan-Mar 2025
(8 week course)

Prof. Satyaki Roy
NPTEL Coordinator
IIT Kanpur

Indian Institute of Technology Kanpur

swayam

Roll No: NPTEL25CS12S332400252 To verify the certificate No. of credits recommended: 2 or 3

Elite

NPTEL ONLINE CERTIFICATION

(Funded by the MoE, Govt. of India)

Skill India
कौशल भारत-कुशल भारत

This certificate is awarded to

MURAKALA PRASANTHI

for successfully completing the course

Cloud Computing and Distributed Systems

with a consolidated score of 65 %

Online Assignments	19.58/25	Proctored Exam	45/75

Total number of candidates certified in this course: 4408

Prof. B. V. Ratish Kumar
Chairman, Centre for Continuing Education
IIT Kanpur

Jan-Mar 2025
(8 week course)

Prof. Satyaki Roy
NPTEL Coordinator
IIT Kanpur

Indian Institute of Technology Kanpur

swayam

Roll No: NPTEL25CS12S432400449 To verify the certificate No. of credits recommended: 2 or 3

Elite

NPTEL ONLINE CERTIFICATION

(Funded by the MoE, Govt. of India)

Skill India
कौशल भारत-कुशल भारत

This certificate is awarded to

BANAPURAM HEMASUNDARA RAO

for successfully completing the course

Python for Data Science

with a consolidated score of **67** %

Online Assignments	24.83/25	Proctored Exam	42.61/75

Total number of candidates certified in this course: **15251**

Prof. Andrew Thangaraj
Chair
Centre for Outreach and Digital Education, IITM

Jan-Feb 2025
(4 week course)

Prof. Vignesh Muthuvijayan
NPTEL Coordinator
IIT Madras

Indian Institute of Technology Madras

swayam

Roll No: NPTEL25CS60S332400035 To verify the certificate No. of credits recommended: 1 or 2

NPTEL ONLINE CERTIFICATION

(Funded by the MoE, Govt. of India)

Skill India
कौशल भारत-कुशल भारत

This certificate is awarded to

CHELLURU PAVAN KUMAR

for successfully completing the course

Python for Data Science

with a consolidated score of **58** %

Online Assignments	23.75/25	Proctored Exam	34.7/75

Total number of candidates certified in this course: **15251**

Prof. Andrew Thangaraj
Chair
Centre for Outreach and Digital Education, IITM

Jan-Feb 2025
(4 week course)

Prof. Vignesh Muthuvijayan
NPTEL Coordinator
IIT Madras

Indian Institute of Technology Madras

swayam

Roll No: NPTEL25CS60S332400089 To verify the certificate No. of credits recommended: 1 or 2