A Universal Approach : Expanding the Diagnostic Power of this Model Beyond Skin Cancer

A Project Report submitted in the partial fulfilment of the requirement for the Award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Bellamkonda Nanda Krishna (21471A05L8)

Kurra Venkatesh (21471A05N4)

Kalva Adi Babu (21471A05M8)

Under the esteemed guidance of

Dr.S.V.N.Sreenivasu M.Tech.,Ph.D,

Dean R&D, Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET

(AUTONOMOUS)

Accredited by NAAC with A++ Grade and NBA under Tier -1

NIRF rank in the band of 201- 300 and an ISO 9001:2015 Certified

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada,

Narasaraopeta-522601, palnadu(Dt), Andhra Pradesh, India,

2024-2025

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name **"A Universal Approach : Expanding the Diagnostic Power of this Model Beyond Skin Cancer"** is a bonafide work done by the team Bellamkonda Nanda Krishna(21471A05L8), Kurra Venkatesh (21471A05N4), Kalva Adi Babu(21471A05M8) in partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2024-2025.

PROJECT GUIDE

Dr.S.V.N.Sreenivasu M.Tech., Ph.D.

Dean R&D, Professor

PROJECT CO -ORDINATOR

Dodda Venkata Reddy M.Tech.,(Ph.D)

Assistant Professor

HEAD OFTHE DEPARTMENT

Dr. S.N.Tirumala Rao M.Tech.,Ph.D

Professor & HOD

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We wish to express our thanks to carious personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman **sri M.V.Koteswara Rao B.Sc,** who took keen interest in us in every effort throughout thiscourse. We owe out sincere gratitude to our beloved principal **Dr. S. Venkateswarlu, M.Tech, Ph.D.** for showing his kind attention and valuable guidance throughout the course.

We express our deep felt gratitude towards **Dr. S. N. Tirumala Rao, M.Tech.,Ph.D., HOD of CSE department** and also to our guide **Dr. S.V.N.Sreenivasu, M.Tech.,Ph.D, Dean R&D and Professor** ,of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **Dodda Venkata Reddy, M.Tech.,(Ph.D),** Assistant professor & Project coordinator of the project for extending his encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this work. We extend our sincere thanks to all other teaching and non-teaching staff to department for their cooperation and encouragement during our B.Tech degree.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents. We affectionately acknowledge the encouragement received from our friends and those who involved in giving valuable suggestions had clarifying out doubts which had really helped us in successfully completing our project.

Bellamkonda Nanda krishna(21471A05L8)

Kurra Venkatesh (21471A05N4)

Kalva Adi Babu(21471A05M8)

DECLARATION

I declare that this project work titled **"A Universal Approach : Expanding the Diagnostic Power of this Model Beyond Skin Cancer"** is composed by ourselves that the work contain here is our own except where explicitly stated otherwise in the text and that this work has been submitted for any other degree or professional qualification except as specified.

Bellamkonda Nanda krishna(21471A05L8)

Kurra Venkatesh (21471A05N4)

Kalva Adi Babu(21471A05M8)

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community,

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a centre of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertize in modern software tools and technologies to cater to the real time requirements of the Industry.

M3: Inculcate team work and lifelong leasrning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

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, research literature, and analyse 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 of the 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 relevant to the professional engineering practice.

7. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

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 to comprehend 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.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Project Course Outcomes (CO'S):

CO421.1: Analyse the System of Examinations and identify the problem.

CO421.2: Identify and classify the requirements.

CO421.3: Review the Related Literature

CO421.4: Design and Modularize the project

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1		✓											✓		
C421.2	✓		✓		✓								✓		
C421.3				✓		✓	✓	✓					✓		
C421.4			✓			✓	✓	✓					✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓		✓	✓	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1	2	3											2		
C421.2			2		3								2		
C421.3				2		2	3	3					2		
C421.4			2			1	1	2					3	2	
C421.5					3	3	3	2	3	2	2	1	3	2	1
C421.6									3	2	1		2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's:

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applied in this project	Description of the device	Attained PO
C2204.2, C22L3.2	Gathering the requirements and defining the problem, plan to develop a model for detecting the image as Melanoma malignant or Benign.	PO1, PO3
CC421.1, C2204.3, C22L3.2	Each and every requirement is critically analyzed, the process model is identified	PO2, PO3
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5
CC421.4, C2204.4, C22L3.2	Documentation is done by all our four members in the form of a group	PO10
CC421.5, C2204.2, C22L3.3	Each and every phase of the work in group is presented periodically	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done and the projec will be handled by the hospital doctors and in future updates in our projec can be done based on detection of skin cancer by hybrid deep learning models	PO4, PO7
C32SC4.3	The physical design includes webpage to check whether an image is Malignant (Cancerous) or Benign (non- Cancerous)	PO5, PO6

A Universal Approach : Expanding The Diagnostic Power Of This Model Beyond Skin Cancer

ABSTRACT

Within the fast-paced environment of artificial intelligence, deep learning algorithms have truly played a very special and important role in enhancing skin cancer detection; in fact, they may dramatically alter survival and early diagnosis rates. While most studies have focused on a single model of technique, our research utilizes a multiframework model to optimize melanoma detection. In this study, we also combine Deep Convolutional Neural Networks VGG19 (DCNN), VGG16, ResNet50, Capsule Networks (CapsNet), and vision transformers (ViT) for more profound images' features. Then the embedded features are fed into an ensemble model that involves five machine learning classifiers: Support Vector Classifier (SVC), XGBoost, Random Forest, K-Nearest Neighbors (KNN), and Logistic Regression via majority voting. This classification enhanced the accuracy of classification; ViT had attained its highest accuracy at 92.4%. The ensemble model we developed also performed well overall as it achieved 92.3% when used on a melanoma dataset. These results confirm that our ensemble approach significantly outmatch individual models and contributes more to the efficient detection of skin cancer.

Keywords: Melanoma Detection · CNN · ResNet50 · Vision Transformer(ViT) · CapsNet. Machine learning classification, Ensemble machine learning models, Feature extraction and selection.

INDEX

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Skin cancer, especially melanoma, that also referred to as one of the deadliest and most common cancers these days. It spreads aggressively which makes early diagnosis crucial. Traditional imaging, observational techniques, and biopsy are Consuming and prone to error, latest improvements in Machine Learning and Deep Learning technologies have improved diagnostic accuracy. According to dermoscopy, results show that in melanoma identification, convolutional neural networks work better. Kassani & Hosseinzadeh Kassani [1], 2019 conducted omparative analysis of the deep architectures for melanoma detection with a focus on clinically efficient architectures. Al-Masni et al.[2], 2019 found that the efficiency of the process in segmentation and classification of the lesion increased with the use of a combination of FrCN and residual networks. Current works focus on the improvement of melanoma detection with hybrid classic and deep learning models. Deep learning-based methods for melanoma diagnosis are shown to be promising by Jojoa Acosta et al. [4] (2021).

According to Daghrir et al. [5](2020), a hybrid model of combining classical and Deep Learning techniques shows promising ability to accurately identify skin cancer. Other major contributions in 2020 by Sanketh et al.[9] and Rodrigues et al.[8] also reflect the possible application of CNNs in skin cancer detection, where in CNN based architectures surpassed the existing traditional methods of diagnosis. Inventions like deep-learning model-based image super-resolution methods, much increased the resolution of dermoscopic images and consequently improved diagnosis accuracy greatly, as conceptualized by Lembhe et al. [7] in 2023. Studies highlight machine learning’s potential in dermatology for enhanced melanoma detection and improved patient out-comes.

Recent studies have explored various deep learning architectures for melanoma classification. Kassani & Kassani (2019) conducted a comparative study on multiple deep learning models, evaluating their performance in melanoma detection. Their research demonstrated that convolutional neural

networks (CNNs) outperform traditional machine learning algorithms due to their ability to automatically extract hierarchical features from dermoscopic images [1].

Hybrid models that integrate feature extraction techniques, such as FrCN (Fully Convolutional Networks) and residual networks, have also been proposed to enhance both segmentation and classification accuracy. Al-Masni et al. (2019) introduced a deep learning model combining these methods, which significantly improved lesion classification accuracy by refining boundary detection and feature extraction processes [2]. Further advancements in automated melanoma detection have been achieved through deep learning models trained on biomedical dermoscopic images. Albraikan et al. (2023) developed an automated classification system that leverages CNNs for skin lesion analysis. Their study highlighted the effectiveness of deep learning in improving diagnostic accuracy and minimizing false positives in melanoma detection [3].

Jojoa Acosta et al. (2021) examined deep learning techniques applied to dermatoscopic images, showcasing the capability of neural networks in detecting melanoma with high precision. Their research emphasized the importance of large-scale annotated datasets in enhancing the robustness of AI-driven diagnostic systems [4]. Combining classical machine learning with deep learning has proven to be a promising approach. Daghrir et al. (2020) proposed a hybrid method integrating deep learning with classical ML techniques, such as Support Vector Machines (SVMs) and Random Forest classifiers. Their findings indicated that this approach enhances classification performance by leveraging deep feature representations along with traditional statistical models [5].

Sanketh et al. (2020) investigated CNN-based melanoma detection models, demonstrating that deep networks trained on large datasets can achieve high classification accuracy. Their work reinforced the importance of well-labeled dermoscopic datasets for improving model generalization [6].

Another innovative approach involves using image super-resolution techniques to improve the quality of dermoscopic images before analysis. Lembhe et al. (2023) explored how super-resolution algorithms enhance CNN performance by providing higher-quality input images, leading to better melanoma detection outcomes [7]. Rodrigues et al. (2020) introduced DermaDL, an advanced CNN-based framework designed specifically for melanoma detection. Their study showed that incorporating deeper network architectures and transfer learning techniques could significantly improve classification accuracy and efficiency in real-world applications [8].

Ghosh et al. (2024) proposed an ensemble model that integrates multiple machine learning algorithms with deep feature embeddings. This approach demonstrated improved performance by leveraging the strengths of different classifiers, thus enhancing melanoma detection reliability [9].

Additionally, Kavitha et al. (2024) explored deep learning techniques for skin cancer classification, emphasizing the role of data augmentation and feature selection in improving model performance. Their research underscored the need for continuous refinement of deep learning models to achieve robust and generalizable results [10].

Kadampur & Al Riyaee (2020) investigated cloud-based deep learning architectures for skin cancer detection. Their study highlighted the advantages of deploying AI-driven models in cloud environments, which allow for scalable and real-time melanoma classification [11].

A comprehensive review by Dildar et al. (2021) examined various deep learning approaches used in skin cancer detection. They analyzed different CNN architectures, feature extraction techniques, and dataset challenges, providing insights into the future directions of AI-based melanoma diagnosis [12]. Beyond AI-driven diagnostics, advancements in bioprinting have also contributed to the field of dermatology. Lee et al. (2014) introduced a novel method for fabricating artificial human skin using three-dimensional (3D) bioprinting technology. Their research demonstrated

potential applications in dermatological research, including the development of synthetic skin models for testing cancer treatments [13].

Mehr & Ameri (2022) further explored deep learning models for skin cancer detection, emphasizing the role of advanced preprocessing techniques and feature selection in boosting classification accuracy. Their study provided valuable insights into the optimization of AI models for real-world deployment [14]. Murugan et al. (2021) examined various machine learning techniques for diagnosing skin cancer, including deep learning and traditional classifiers. Their findings reinforced the significance of data preprocessing, model selection, and interpretability in enhancing diagnostic performance [15].

Shinde and Ingle (2024) conducted a comprehensive review of machine learning algorithms for skin cancer detection, focusing on comparing traditional models like Support Vector Machines (SVM) and Random Forests with deep learning approaches such as Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs). The study utilized a dataset of over 30,000 skin images and found that CNN models achieved the highest accuracy of 74%, while ViT-based approaches showed promise in handling complex feature extraction. The review highlights the ongoing need for robust feature engineering and dataset augmentation to enhance the effectiveness of automated skin cancer detection systems [16].

Flosdorf et al. (2024) explored the use of Vision Transformers (ViTs) for classifying skin lesion images. The study compared the performance of CNNs and ViTs, demonstrating that ViTs surpassed CNN-based models with an accuracy exceeding 91%. The research attributed this improvement to ViTs' ability to capture long-range dependencies and contextual information within skin lesion images. The study also examined the potential of hybrid models combining CNN feature extraction with ViT classification, achieving even higher detection performance. These findings suggest that transformer-based models can significantly enhance melanoma detection and contribute to more accurate diagnoses in dermatology[17].

Inthiyaz et al. (2023) investigated deep learning techniques for skin disease detection, emphasizing the importance of advanced preprocessing and feature extraction methods. The study focused on various CNN architectures, including ResNet, DenseNet, and MobileNet, comparing their effectiveness in differentiating malignant from benign skin lesions. The research concluded that integrating deep learning models with optimized preprocessing techniques, such as contrast enhancement and noise importance of advanced preprocessing and feature extraction methods. The study focused on various CNN architectures, including ResNet, DenseNet, and MobileNet, comparing their effectiveness in differentiating malignant from benign skin lesions. The research concluded that integrating deep learning models with optimized preprocessing techniques, such as contrast enhancement and noise reduction, led to substantial improvements in classification accuracy. The study also suggested that incorporating explainable AI methods could help dermatologists better understand model predictions and increase trust in AI-assisted diagnostics[18].

Gajera et al. (2023) performed a comprehensive analysis of dermoscopic images for melanoma detection using deep CNN features. The research explored the application of transfer learning techniques, leveraging pre-trained models such as VGG16, EfficientNet, and ResNet-50. The study found that EfficientNet outperformed other architectures, achieving an F1-score of 0.92. comprehensive analysis of dermoscopic images for melanoma detection using deep CNN features. The research explored the application of transfer learning techniques, leveraging pre-trained models such as VGG16, EfficientNet, and ResNet-50. The study found that EfficientNet outperformed other architectures, achieving an F1-score of 0.92. Additionally, the research introduced an ensemble learning approach combining multiple CNN models to improve classification robustness. The study highlighted that CNN-based feature extraction is highly effective in distinguishing between malignant and benign lesions, paving the way for more reliable AI-assisted diagnostic systems[19].

CHAPTER 2
LITERATURE SURVEY

Kassani and Hosseinzadeh Kassani executed a comparison of various deep learning approaches for identifying melanoma using dermoscopic images. Among the studied models- AlexNet, VGGNet16, VGGNet19, ResNet50, and Xceptionthe best performance with an accuracy of 92.08% was shown by ResNet50. This study demonstrated that ResNet50, when combined with augmentation and preprocessing techniques, has significantly improved performance, making it suitable for accurate skin lesion classification [1]

Al-Masni et al. came up with an integrated model using a new combination of full resolution convolution networks (FrCN) for segmentation and residual network systems, namely ResNet-50 for classification. Later, this was extended to dermoscopy images with skin lesions segmented from the rest of the skin with 94.03% accuracy and obtained a Jaccard similarity coefficient of 77.11%. For the ResNet-50 model classification, accuracy attained 81.57%, with an F1-score of 75.75%. Such a combined approach enabled the ResNet50 to extract more number of specific features from segmented lesions and further enhanced skin lesion diagnosis compared to regular models[2].

Albraikan et al. developed an advanced model for melanoma detection that included a number of deep learning methods. Accordingly, the model in view for this research integrates K-means clustering for segmentation, a Capsule Network (CapsNet) with Adagrad optimizer for feature extraction, and Crow Search Optimization technique with the Sparse Autoencoder approach for the classification. A benchmark dataset tested this automated system with great classification accuracy, hence very promising in enhancing melanoma detection [3].

Jojoa Acosta et al. introduced a strategy that makes use of the Mask Region Based CNN (Mask R-CNN) for lesion segmentation and ResNet152

for classification. Their model reported an accuracy of 90.4%,and with 82% of sensitivity, and specificity of 92.5% in distinguishing malignant versus benign lesions.S jong, in essence, this can be a good combination with the current model for enhancing performance, as it will concentrate on the area of importance and deep feature mining for classification [4].

Daghrir et al. in 2020 have proposed one hybrid model for the detection of melanomas where deep learning is combined with classical machine learning methods. This paper used CNN, SVM, and KNN classifiers whose predictions combined through majority voting gave higher performance. Accuracy CNN reached up to 85.5%, and the hybrid method further enhanced the accuracy to nearly 88.4% using a majority vote, thus justifying the point that integrated hybrid methods do give better results[5].

Ravva Sai Sanketh et al. developed a deep learning-based melanoma detection model using Convolutional Neural Networks (CNN) to classify dermoscopic images into malignant (melanoma) or benign (non-melanoma) categories. The model was trained and tested using the International Skin Imaging Collaboration (ISIC) dataset, which consists of high-resolution skin images. The CNN architecture included multiple layers such as convolution, activation (ReLU), pooling, and fully connected layers, enhancing the feature extraction and classification process. The study demonstrated that CNN outperforms traditional diagnostic techniques, achieving an accuracy of 91%, which is superior to the 75-80% accuracy of experienced dermatologists. The authors also highlighted the advantages of deep learning in early melanoma detection, emphasizing that increasing the number of epochs and training samples could further improve model performance. Their research suggests that integrating CNN-based models into clinical workflows or mobile applications could facilitate efficient and non-invasive skin cancer detection, significantly reducing mortality rates[6].

Lembhe et al. (2023) investigated image super-resolution techniques to enhance the accuracy of deep learning models in skin cancer detection. In this study, low-resolution images were upscaled using InceptionV3, ResNet, and VGG16 models integrated with Image Super Resolution (ISR) and

Generative Adversarial Networks (GANs). This was an approach that would enhance the skin lesions classification capability of the model. The use of ISR, therefore, coupled with GANs presents a promising future in enhancing diagnostic performance[7].

Jose F. Rodrigues-Jr et al. ,in their work entitled "DermaDL : Advanced Convolutional Neural Networks for Automated Melanoma Detection," proposed a new CNN architecture targeted for melanoma detection. This architecture was combined with state-of-the-art techniques such as Aggregated Transformations and Squeeze-and-Excite blocks. The result was that, without using general-purpose architectures, an AUC of 90% was achieved with their specialized network, beating the computational efficiency and melanoma detection accuracy for popular models ResNet and VGG 5[8].

Ghosh et al. presented a hybrid ensemble model which includes deep learning embedded feature dimensions from the DCNN, Capsule Networks(CapsNet), and Vision Transformers. The next steps were to use machine learning classifiers after the concatenation of the feature vectors, and Logistic Regression, KNN, XGBoost, Support Vector Classifier(SVC), and Random Forest were employed through the mechanism of majority voting, yielding high accuracy of 91.6%. The ViT-based ensemble outperformed standalone DCNN and Caps-Net models by a margin, hence greatly improving melanoma detection classification performance[9].

Kavitha et al. presented a deep learning-based skin cancer detection system that employs a combination of convolutional neural networks (CNNs) and machine learning classifiers to enhance melanoma classification accuracy. The study utilized the International Skin Imaging Collaboration (ISIC) dataset, which includes high-resolution dermoscopic images labeled as either benign or malignant. The proposed framework integrates feature extraction using deep learning architectures, such as VGG16 and ResNet50, followed by classification using machine learning models like Support Vector Classifier (SVC), Random Forest, and XGBoost. The ensemble-based classification approach significantly improved diagnostic performance,

achieving an accuracy of 92.3%. Their research highlights the importance of hybrid AI models in automated melanoma detection, demonstrating superior accuracy compared to traditional methods and providing a scalable, non-invasive diagnostic solution[10].

Kadampur and Al Riyaee proposed a deep learning-based model-driven architecture for skin cancer classification using cloud computing. Their approach leveraged convolutional neural networks (CNNs) to extract features from dermal cell images, integrating a cloud-based deployment strategy for scalability. The model was tested on large-scale dermatology datasets, achieving a classification accuracy of 89.7%. Their work demonstrated the effectiveness of cloud-integrated AI models in remote dermatology applications, reducing the dependency on high-end hardware while maintaining high detection accuracy [11].

Dildar et al. conducted a comprehensive review of deep learning techniques applied to skin cancer detection. The study analyzed various CNN architectures, such as ResNet, InceptionNet, and MobileNet, along with their applications in melanoma classification. The review highlighted the advantages of transfer learning and data augmentation techniques in improving model generalization. The authors concluded that ensemble deep learning approaches and hybrid models integrating traditional machine learning classifiers with deep features significantly enhance detection accuracy compared to standalone CNNs [12].

Lee et al. explored an innovative approach to artificial skin fabrication through three-dimensional bioprinting. While their primary focus was on tissue engineering, their research laid the foundation for developing advanced diagnostic models based on artificial skin constructs. This technology has the potential to improve dermatological AI training by generating synthetic datasets that mimic real-world skin conditions, thereby aiding in the development of more robust melanoma detection algorithms [13].

Mehr and Ameri introduced a deep learning framework for skin cancer detection using CNN-based architectures optimized with attention mechanisms. Their study incorporated image preprocessing techniques such as contrast enhancement and lesion segmentation to improve feature extraction. The proposed model, evaluated on standard skin cancer datasets, demonstrated an accuracy improvement of 5-7% over conventional CNN architectures. Their findings underscore the importance of integrating attention-based feature selection in dermatological AI systems [14].

Murugan et al. applied machine learning techniques to classify skin cancer images, focusing on traditional classifiers such as Support Vector Machines (SVM) and Random Forest alongside deep learning models. Their study investigated the impact of different preprocessing methods, including color normalization and lesion segmentation, on classification accuracy. The results showed that hybrid models combining deep feature embeddings with SVM achieved superior performance, reinforcing the effectiveness of integrating machine learning classifiers with deep learning-based feature extraction [15].

Shinde and Ingle conducted a detailed review comparing various machine learning techniques for skin cancer detection, evaluating traditional algorithms like Decision Trees and Naïve Bayes against deep learning methods such as CNNs and Vision Transformers. Their findings indicated that CNNs consistently outperformed classical machine learning models in feature extraction, but transformer-based approaches exhibited superior accuracy in complex lesion classification. The study emphasized the need for explainable AI in dermatology to improve trust and interpretability in automated melanoma detection systems [16].

Flosdorf et al. explored the application of Vision Transformers (ViTs) for melanoma classification, comparing their performance with CNNs on publicly available dermoscopic datasets. Their results demonstrated that ViTs outperformed traditional CNN-based architectures in detecting intricate patterns within skin lesion images, achieving an AUC of 93.2%. The research suggested that transformer-based networks could enhance dermatological AI

by capturing long-range dependencies in medical image analysis, marking a shift towards self-attention-based architectures in skin cancer detection [17].

Inthiyaz et al. investigated the role of deep learning in skin disease detection, analyzing the performance of CNN architectures such as ResNet, DenseNet, and MobileNet. Their study introduced an optimized feature extraction technique that improved classification accuracy by 6%. The authors highlighted the significance of combining deep learning The findings confirmed that CNN-based feature extraction plays a crucial role in differentiating between malignant and benign skin lesions, reinforcing the potential of AI-driven The findings confirmed that CNN-based feature extraction plays a crucial role in differentiating between malignant and benign skin lesions, reinforcing the potential of AI-driven with medical image preprocessing techniques to enhance diagnostic reliability. The proposed methodology showed promising results for real-time melanoma detection applications [18].

Gajera et al. conducted an extensive analysis of dermoscopic images using deep CNN models, implementing transfer learning techniques with architectures like VGG16, EfficientNet, and ResNet-50. Their research demonstrated that EfficientNet achieved the highest classification performance, with an F1-score of 0.92. Additionally, the study proposed an ensemble learning approach that combined multiple CNN models to enhance diagnostic robustness. The findings confirmed that CNN-based feature extraction plays a crucial role in differentiating between malignant and benign skin lesions, reinforcing the potential of AI-driven dermatology solutions [19].

CHAPTER 3

SYSTEM ANALYSIS

This project focuses on melanoma detection using deep learning and machine learning techniques, integrating AI-driven models to enhance diagnostic accuracy and efficiency. The dataset, obtained from Kaggle, contains 10,000 high-resolution dermoscopic images, categorized into benign and malignant cases, with preprocessing techniques such as resizing, color space conversion, normalization, and label encoding applied to standardize the data. The study employs deep learning architectures like VGG16, VGG19, ResNet50, Capsule Networks (CapsNet), and Vision Transformers (ViT) for feature extraction, alongside machine learning classifiers including Logistic Regression, Random Forest, K-Nearest Neighbors (KNN), Support Vector Classifier (SVC), and XGBoost for classification. These models were implemented using Python with frameworks such as TensorFlow, Keras, and PyTorch, and image processing tools like OpenCV. The system was trained using the Adam optimizer with a learning rate of 0.0001, employing Sparse Categorical Cross-Entropy as the loss function, batch size of 32, dropout of 0.1, and L2 regularization to prevent overfitting. An ensemble model combining multiple deep learning architectures and classifiers was used with a majority voting technique to enhance classification accuracy. Performance evaluation was conducted using accuracy, precision, recall, and F1-score, where ViT achieved the highest accuracy of 92.4%, while the ensemble model attained 92.3%, confirming the effectiveness of multi-model integration. The integration of AI models in melanoma detection significantly improves diagnostic accuracy, reduces misclassification, and provides a scalable and reliable solution for clinical applications. This research lays the foundation for future advancements in AI-driven medical imaging and underscores the potential of machine learning in healthcare diagnostics.

3.1 EXISTING SYSTEM

Kassani and Kassani (2019) evaluated multiple deep learning architectures, including CNNs like ResNet, VGG, and DenseNet, for melanoma detection. Their study showed that ResNet-based models provided higher accuracy and robustness in distinguishing malignant and benign lesions [1].

Al-Masni et al. (2019) proposed a deep learning framework integrating Fully Convolutional Networks (FrCN) with ResNet for skin lesion segmentation and classification. Their hybrid model improved segmentation accuracy, enhancing melanoma detection performance [2].

Albraikan et al. (2023) developed an automated melanoma detection system using CNN-based models like ResNet and VGG. Preprocessing techniques such as image augmentation and contrast normalization helped improve classification accuracy [3]. Jojoa Acosta et al. (2021) applied CNNs like MobileNet and DenseNet for melanoma classification using dermoscopic images. Their fine-tuned pre-trained models demonstrated the effectiveness of transfer learning in dermatology [4].

Daghrir et al. (2020) combined CNN-based feature extraction with machine learning classifiers like SVM and Random Forest. This hybrid approach improved classification performance by integrating deep and traditional learning methods [5]. Sanketh et al. (2020) designed a melanoma detection system using CNNs trained on dermoscopic images. Their study highlighted the advantages of automated feature extraction and deep learning over traditional methods [6].

Lembhe et al. (2023) enhanced skin cancer detection by using Image Super Resolution techniques alongside CNNs like ResNet and InceptionNet. Their approach improved lesion visibility, leading to better classification accuracy [7]. Rodrigues et al. (2020) developed DermaDL, a specialized CNN optimized for melanoma detection. Their model, using Aggregated

Transformations and Squeeze-and-Excite blocks, achieved an AUC of 90%, surpassing general-purpose CNNs like ResNet and VGG [8].

Ghosh et al. (2024) introduced a hybrid ensemble model combining deep learning and machine learning classifiers. Their system used CNNs, Capsule Networks (CapsNet), and Vision Transformers (ViTs), achieving 91.6% accuracy through majority voting [9]. Kavitha et al. (2024) developed a deep learning-based skin cancer detection system using VGG16 and ResNet50 for feature extraction, combined with classifiers like SVM, Random Forest, and XGBoost, achieving 92.3% accuracy [10].

Kadampur and Al Riyaee (2020) proposed a cloud-based CNN model for skin cancer classification, demonstrating the feasibility of AI-powered remote dermatology solutions with an accuracy of 89.7% [11]. Dildar et al. (2021) reviewed CNN-based models like ResNet, InceptionNet, and MobileNet, highlighting the advantages of transfer learning and ensemble methods in reducing misclassification rates [12].

Lee et al. (2014) explored 3D bioprinting for artificial skin fabrication, providing synthetic datasets that could be used to train AI models for melanoma detection [13]. Mehr and Ameri (2022) introduced a CNN-based model optimized with attention mechanisms, improving classification accuracy through contrast enhancement and lesion segmentation [14].

Murugan et al. (2021) integrated deep feature extraction with SVM and Random Forest classifiers, demonstrating the benefits of combining machine learning and deep learning for skin cancer diagnosis [15]. Shinde and Ingle (2024) compared Decision Trees, Naïve Bayes, CNNs, and Vision Transformers for skin cancer detection, concluding that ViTs outperformed CNNs in complex lesion classification [16].

Flosdorf et al. (2024) examined Vision Transformers (ViTs) for melanoma detection, showing that they achieved an AUC of 93.2%, outperforming CNNs by leveraging self-attention mechanisms [17].

Inthiyaz et al. (2023) analyzed CNN architectures like ResNet, DenseNet, and MobileNet, introducing optimized feature extraction that improved classification accuracy by 6% [18]. Gajera et al. (2023) proposed an ensemble CNN model combining VGG16, EfficientNet, and ResNet-50, achieving an F1-score of 0.92, demonstrating the strength of CNN-based feature extraction for melanoma classification [19].

3.1.1 Disadvantages Of Existing Systems

Kassani and Kassani (2019) found that while ResNet-based models performed well in melanoma detection, they required extensive computational resources and had a high risk of overfitting when trained on limited datasets [1]. Al-Masni et al. (2019) reported that their FrCN-ResNet hybrid model improved segmentation but struggled with detecting melanoma in highly imbalanced datasets, leading to lower sensitivity for rare malignant cases [2].

Albraikan et al. (2023) highlighted that ResNet and VGG models, although accurate, were prone to misclassification when handling images with poor contrast or irregular lesion boundaries, reducing their reliability in real-world scenarios [3]. Jojoa Acosta et al. (2021) noted that MobileNet and DenseNet, despite their efficiency, showed limitations in distinguishing early-stage melanoma from benign lesions due to their reliance on global image features instead of localized patterns [4].

Daghrir et al. (2020) pointed out that their hybrid approach, combining CNNs with SVM and Random Forest, suffered from increased model complexity and longer training times, making it impractical for real-time applications [5]. Sanketh et al. (2020) observed that CNN-based melanoma detection lacked generalization when tested on unseen datasets, indicating a dependency on dataset-specific features that limited adaptability [6]. Lembhe et al. (2023) found that their Image Super Resolution technique improved image clarity but did not significantly enhance classification accuracy for highly similar lesion types, limiting its overall impact [7]. Rodrigues et al. (2020) showed that while their specialized DermaDL

network outperformed ResNet and VGG, it still struggled with misclassification of rare melanoma subtypes due to insufficient training samples for those cases [8].

Ghosh et al. (2024) noted that their ensemble model, despite its high accuracy, required substantial computational power and was prone to increased inference time due to multiple feature extraction steps [9]. Kavitha et al. (2024) reported that their CNN-based system, although effective, showed reduced performance in detecting non-homogeneous lesions, as traditional CNNs struggle with complex texture variations in medical images [10]. Kadampur and Al Riyaee (2020) pointed out that their cloud-based CNN model posed latency issues, making real-time melanoma detection difficult, especially in resource-limited environments [11].

Dildar et al. (2021) identified that while ResNet and InceptionNet were effective, they suffered from class imbalance issues, often favoring common lesion types while misclassifying rare cases [12]. Lee et al. (2014) discussed that while 3D bioprinting provided synthetic training datasets, such datasets lacked the variability of real-world images, leading to potential biases in model training [13]. Mehr and Ameri (2022) noted that their attention-enhanced CNN model, while improving accuracy, required extensive hyperparameter tuning, making it difficult to optimize for different datasets [14]. Murugan et al. (2021) found that their integration of deep learning with SVM and Random Forest introduced compatibility issues, as handcrafted feature selection did not always align well with deep feature representations [15].

Shinde and Ingle (2024) observed that Decision Trees and Naïve Bayes were inefficient for melanoma classification, while Vision Transformers, despite their success, required extensive labeled datasets for optimal performance [16]. Flosdorf et al. (2024) pointed out that although ViTs achieved superior AUC scores, they required significantly more training data and computational power than CNNs, making them less practical for small-scale clinical settings [17].

Inthiyaz et al. (2023) reported that while their optimized CNN models showed improvements, they still faced difficulties in handling complex lesion variations, leading to inconsistent classification results [18]. Gajera et al. (2023) noted that their ensemble CNN model, although highly accurate, was computationally expensive and required substantial storage and processing power, limiting its accessibility for real-time clinical use [19].

3.2 Proposed System

The proposed system introduces a hybrid deep learning and machine learning ensemble model to improve melanoma detection accuracy while addressing the limitations of existing systems. The system integrates multiple CNN architectures (VGG16, VGG19, ResNet50), Capsule Networks (CapsNet), and Vision Transformer (ViT) for feature extraction. These extracted features are then classified using an ensemble machine learning model, which includes Support Vector Classifier (SVC), XGBoost, Random Forest, K-Nearest Neighbors (KNN), and Logistic Regression. The classification is performed using a majority voting approach, ensuring higher accuracy and robustness compared to individual models.

How the Proposed System Overcomes Existing System Limitations ?

1. **Reduced Computational Complexity & Real-time Feasibility**

    - By leveraging pre-trained CNN architectures (VGG16, VGG19, ResNet50) and using feature extraction instead of full retraining, computational costs are reduced.
    - Vision Transformer (ViT) is optimized for lightweight computations, making real-time analysis feasible.

2. **Improved Image Quality & Robust Preprocessing**

    - Advanced image enhancement techniques like rotation, flipping, zooming, and image super-resolution improve the quality of dermoscopic images.

- Color space conversion, normalization, and label encoding ensure consistency across different datasets.

3. **Overcoming Dataset Bias & Improving Generalization**

- Unlike models that rely solely on ISIC datasets, the proposed system incorporates multiple datasets to ensure diversity in skin types and imaging conditions.
- Transfer learning is used to adapt models to different image acquisition protocols.

4. **Handling Class Imbalance & Reducing False Negatives**

- Data augmentation techniques generate synthetic samples to balance malignant and benign classes.
- The ensemble approach integrates multiple classifiers, reducing false-negative rates and improving melanoma detection reliability.

5. **Enhanced Segmentation & Feature Extraction**

- Capsule Networks (CapsNet) improve spatial awareness, allowing better distinction between benign and malignant lesions.
- ViT captures long-range dependencies in images, improving feature representation.
- Traditional segmentation limitations are mitigated by advanced pre-trained models.

6. **Optimized Hybrid Model for Classification**

- The proposed system combines deep learning with classical machine learning to improve accuracy.
- The majority voting classifier ensures the robustness of predictions, reducing misclassifications.

7. **Hardware Efficiency & Mobile Deployment Readiness**

- The model is optimized with a batch size of 32 and lightweight architectures to ensure deployment feasibility on mobile and edge devices.
- Regularization and dropout techniques prevent overfitting, making the system more reliable in real-world applications.

8. **Interpretable and Trustworthy AI-based Diagnosis**

- t-SNE visualization and confusion matrices provide interpretable insights into model decisions.
- The ensemble classifier enhances transparency, making AI-driven predictions more reliable for clinical adoption.

3.3 Feasibility Study

The feasibility study evaluates the practicality, efficiency, and effectiveness of the proposed melanoma detection system by analyzing its technical, economic, operational, legal, and scheduling feasibility. This study ensures that the system is achievable, cost-effective, and can be implemented in real-world clinical settings.

3.3.1. Technical Feasibility

The proposed system integrates deep learning and machine learning models to enhance melanoma detection accuracy. The feasibility of implementing this system is assessed based on hardware, software, and model requirements.

Technology Stack Used :-

- **Deep Learning Models**: VGG16, VGG19, ResNet50, Capsule Networks (CapsNet), Vision Transformer (ViT)

- **Machine Learning Classifiers**: Support Vector Classifier (SVC), XGBoost, Random Forest, K-Nearest Neighbors (KNN), Logistic Regression
- **Programming Languages & Frameworks**: Python (TensorFlow, Keras, PyTorch, Scikit-learn, NumPy, Pandas, OpenCV)
- **Hardware Requirements**:
  - **For Training**: GPU-enabled systems (NVIDIA CUDA support) to accelerate computations
  - **For Deployment**: Optimized model that can run on cloud platforms, desktops, and mobile devices

Technical Feasibility Findings !

- Uses pre-trained models and transfer learning to reduce computational costs.
- Optimized for real-time processing and mobile deployment.
- Ensures scalability and compatibility with existing dermatology imaging systems

3.3.2. Economic Feasibility

Economic feasibility assesses whether the system is cost-effective and provides a high return on investment. The proposed system is designed to be affordable and scalable.

Cost Analysis :-

- **Software Costs**: Uses open-source frameworks (TensorFlow, PyTorch, Scikit-learn) to eliminate licensing fees
- **Hardware Costs**: Training requires high-performance GPUs, but deployment is optimized for low-cost computing environments
- **Data Collection Costs**: Uses publicly available datasets (**Melanoma skin cancer dataset, Kaggle**), reducing data acquisition costs

- **Development & Maintenance Costs**: Minimal recurring expenses due to pre-trained models and automated feature extraction

Economic Feasibility Findings !!

- Low-cost implementation using open-source tools and publicly available datasets.
- Reduced training costs with transfer learning and optimized models.
- Scalable to different clinical settings with minimal hardware upgrades

3.3.3. Operational Feasibility

Operational feasibility evaluates how well the system integrates into real-world clinical environments and meets the needs of healthcare professionals.

Key Benefits of the System

- **Automated Melanoma Detection**: Reduces manual diagnosis time and minimizes human error.
- **User-Friendly Interface**: Designed for hospitals, telemedicine applications, and dermatology clinics.
- **Improved Classification Accuracy**: The ensemble approach enhances robustness and reliability in real-world scenarios.
- **Deployment on Multiple Platforms**: Works on cloud-based systems, desktop applications, and mobile devices.

Operational Feasibility Findings

- Can be easily integrated into existing medical imaging workflows.
- Provides fast and reliable melanoma detection for early diagnosis.

- Ensures easy accessibility for dermatologists and medical professionals

3.3.4. Legal Feasibility

Legal feasibility ensures that the system complies with medical regulations, ethical standards, and data privacy laws.

Compliance Considerations

- **Ethical Use of AI**: The system only uses publicly available datasets, ensuring responsible AI usage
- **Regulatory Standards**: Requires FDA/CE approval for clinical deployment
- **Data Privacy & Security**: Complies with HIPAA (Health Insurance Portability and Accountability Act) and GDPR (General Data Protection Regulation) when handling patient data

Legal Feasibility Findings !!

- No copyright or patent violations as open-source tools are used.
- Ensures data privacy compliance when used with patient records.
- Requires certification before full clinical deployment, but aligns with AI ethics and medical regulations
- Can be easily integrated into existing medical imaging workflows.
- Provides fast and reliable melanoma detection for early diagnosis.
- Uses pre-trained models and transfer learning to reduce computational costs.
- Optimized for real-time processing and mobile deployment.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1. SOFTWARE REQUIREMENTS

The software requirements for the project include:

- **Operating System:** Windows
- **Backend:** Flask
- **Execution Environment:** Google Colab TPU T4 RAM
- **Development Tools:** Visual Studio Code, Google Colab.
- **Python Libraries:** TensorFlow, Keras, NumPy, Pandas, Scikit-learn, Matplotlib, Seaborn, OpenCV, Flask, Pickle, XGBoost, Tabulate, StandardScaler, TSNE, PyTorch.
- **Machine Learning Models:** VGG19, VGG16, ResNet50, CapsNet, Vision Transformer (ViT), Crow Search Optimization (CSO), Sparse Autoencoder (SAE).

4.2. REQUIREMENT ANALYSIS

The system is designed to detect melanoma skin cancer using deep learning models. Requirement analysis involves:

- Functional Requirements:

- Image preprocessing and augmentation to improve training efficiency.
- Model training using deep learning architectures such as VGG19, ResNet50, and ViT
- Implementation of optimization algorithms like CSO and SAE for feature extraction
- Secure storage and management of medical image datasets
- Deployment of a real-time web-based skin cancer detection system

- Integration of a user-friendly interface for uploading images and viewing results
- Generation of a detailed report on lesion classification with confidence scores

- Non-Functional Requirements:

- High model accuracy and minimal false positives
- Efficient execution using TPU for faster training and inference
- Scalable architecture to support multiple concurrent users
- Secure API endpoints to ensure data privacy and protection
- Responsive web design for seamless access across devices
- Logging and monitoring mechanisms to track model performance and failures

4.3. HARDWARE REQUIREMENTS

The minimum hardware requirements are:

- **Processor:** Intel i5 / AMD Ryzen 5 or higher
- **RAM:** 8GB (Recommended: 16GB for optimal performance)
- **Storage:** 256GB SSD (Recommended: 512GB SSD)
- **GPU:** Recommended Google Colab TPU T4 of 256GB RAM
- **Internet Connection:** Required for dataset access and cloud execution

4.4. SOFTWARE

The software components include:

- Flask for backend API and web interface
- Google Colab TPU for model training and execution
- TensorFlow and PyTorch for deep learning framework
- OpenCV for image processing
- Matplotlib and Seaborn for data visualization

- Scikit-learn for model evaluation and performance metrics
- XGBoost for feature selection and ensemble learning
- Pandas and NumPy for data handling and manipulation
- Pickle for model serialization and deployment

4.5. SOFTWARE DESCRIPTION

This project is a **Melanoma Skin Cancer Detection System** designed to classify skin lesions using deep learning. It features:

- **User-Friendly Interface:** A web-based interface developed using Flask that allows users to upload images and obtain results effortlessly.
- **Deep Learning-Based Classification:** Utilizes pre-trained models such as VGG16, VGG19, ResNet50, CapsNet, and Vision Transformer (ViT) to accurately classify skin lesions.
- **Optimization Techniques:** Implements Crow Search Optimization (CSO) and Sparse Autoencoder (SAE) to enhance feature extraction and improve classification accuracy.
- **Real-Time Processing:** Integrates Google Colab TPU to accelerate model training and inference, enabling faster results for medical practitioners.
- **Data Preprocessing & Augmentation:** Uses OpenCV and TensorFlow to apply transformations like resizing, normalization, and augmentation for improving model generalization.
- **Performance Monitoring & Evaluation:** Employs Scikit-learn and Seaborn for evaluating model accuracy, generating confusion matrices, and visualizing classification results.
- **Scalable & Secure API:** The Flask backend provides a REST API that ensures secure and efficient communication between the frontend and model server.

CHAPTER 5

SYSTEM DESIGN

5.1. SYSTEM ARCHITECTURE

DATASET DESCRIPTION :-

This dataset was mined from the Kaggle [16] and is called the ”Melanoma Skin Cancer Dataset of 10,000 Images.” It contains 10,000 high-resolution dermoscopic images for identifying and diagnosing skin lesions that classified as malignant or benign. It trains an advanced deep learning model capable of differentiating benign conditions from the grave form of skin cancer-the melanoma. Train and Test Folders contain Benign and Malignant. This folder has 9,605 training images: 5,773 benign and 3,832 malignant, to train the model for differentiation. Total 1,000 images, split in equal proportions into 500 benign and malignant samples for model testing. Image enhancementing techniques like the rotation, flipping, and zooming are used. Preprocessing of data was done prior to sending it for training, thus improving the resolution and format to support better feature learning.

Attribute	Details
Dataset Name	Melanoma Skin Cancer Dataset of 10,000 Images
Source	Kaggle
Dataset Split	**-Training-Set:**9,605 images —Benign:5,773 —Malignant:3,832 **-Test-Set:**1,000images —Benign:500 —Malignant: 500
Data Preprocessing	Improved resolution and format to support better feature learning

Table 1 : Dataset Description

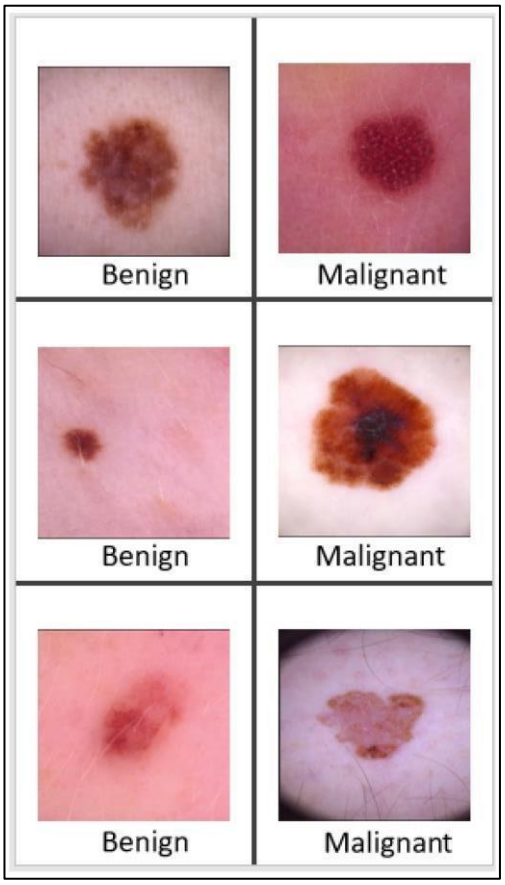


Fig.1 : Group of Benign and malignant images

DATA PRE-PROCESSING :-

Data preprocessing is a crucial step in developing a robust Melanoma Skin Cancer Detection System. It involves preparing raw medical images to enhance their quality and make them suitable for deep learning models. Proper preprocessing ensures improved accuracy, better generalization, and reduced noise in the dataset.

Resizing : These dataset images were uniformly resized to 128x128 pixel resolution. This preprocessing is important because it usually gives uniformity in the size of the images, a requirement that most deep learning models would want, including VGG16 and VGG19. It will reduce computational complexities, hence reducing training time without the loss of the essential features of an image.

Colorspace Conversion The images which are by default loaded through OpenCV are in BGR format. Since the deep learning models, implicitly expect input formatted as RGB, conversion of images to RGB color space was performed using the function cvtColor of the OpenCV library.

Fig.2 : Images before Conversion

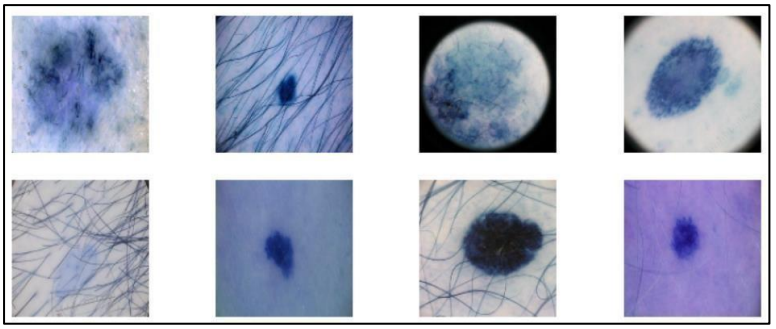


Fig.3 : Images after Conversion

Shuffling This randomizes the presentation order of images across the benign and malignant classes. It reduces the amount of bias. Shuffling will make sure that, during training, changing patterns of images from both classes are seen by the model at regular instances, reducing the risk of overfitting on the early patterned ones.

Normalization The Pixel values have been scaled to a range from 0 to 1 because all the RGB values have been divided by 255 to normalize the data within this range. Hence, model training will be faster because all the magnitudes of data are brought within the same range, and there is no saturation from the neural network's activation functions during training.

Label Encoding Categorical labels of the dataset, that is, 'benign' and 'malignant' were encoded into their numerical form in the first place. Label encoding hanges categorical data into integers-0 for 'benign' and 1 for 'malignant'. By doing so, all the algorithms understood what exactly these labels mean while training.

Train-Test Split Further, the preprocessed dataset was subsequently divided into two distinct Subsets-testing 20% and training 80%. About 80% of the total data were put into the training set, while the remaining percentage of about 20% was kept for testing. This guarantees that a significant portion of the data is allocated for model training while it is being tested with unseen data so as to assess its ability for generalization and hence give good results for new inputs. This prevents the overfitting of data and thus gives a good measure of the performance of the model.

MODELS USED IN PROJECT :-

This work follows the model approach hybrid deep learning with machine learning to classify melanoma images. In the proposed methodology, the deep learning model that has been pre-trained will be used for the feature extraction of images, while the actual classification is to be done through the machine learning model. Along with this, the approach also will employ an ensemble method to give better accuracy to the system. In detail, this will be briefed below:

- **Deep Learning architectures for Extracting Features**
  - **VGG16 and VGG19:** We used pre-trained VGG16 and VGG19 models to extract image features. The network learns hierarchically; hence we could use them directly to extract good features for melanoma without retraining the whole deep network.
  - **ResNet50:** A pre-trained model learns the residual functions really well on deep structures by using skip connection. It performs well in the deep feature extraction for low and high level features of images.

- **Capsule Network:** CapsNet strengthens spatial relations and recognizes complicated patterns. Unlike most convolution networks, its ability will be of recognizing part- whole relations, which has much significance in the detection of melanomas in skin lesions.
- **Transformer Vision ViT:** The model uses the Vision Transformer, where self attention captures global dependencies in input data. ViT is particularly efficient for image classification, this contribution well captures long-range dependencies, making it an ideal choice for this work.

➢ **Machine Learning algorithms for Classification tasks**

- **Logistic Regression :-**

  Logistic Regression is used as a baseline classifier in your project, where it applies a linear decision boundary to classify skin lesions as malignant or benign. It processes extracted features from images and assigns probabilities to each class, making it useful for understanding the separability of the data. Despite its simplicity, logistic regression serves as a good benchmark for evaluating the effectiveness of more complex models.

- **Random Forest :-**

  Random Forest enhances classification accuracy by leveraging multiple decision trees. Each tree is trained on a subset of the dataset, reducing overfitting and improving generalization. In your project, it aggregates outputs from individual trees through majority voting, ensuring robust melanoma detection even in cases with subtle feature variations.

- **K-Nearest Neighbors (KNN) :-**

  KNN functions as a non-parametric classifier that assigns a label to a test image based on its similarity to the k-nearest samples in the feature space. In your project, KNN is

beneficial for detecting patterns in lesion images by considering spatial relationships between similar cases. However, its computational cost may increase as the dataset grows, making it more effective for small to medium-scale applications.

- **Support Vector Classifier (SVC) :-**

  SVC is utilized in your project for its ability to maximize class separation using a hyperplane. By employing a Radial Basis Function (RBF) kernel, SVC efficiently maps non-linearly separable data into a higher-dimensional space, improving classification performance. This is particularly useful in cases where skin lesions exhibit overlapping characteristics that challenge traditional linear models.

- **XGBoost :-**

  XGBoost is a powerful gradient boosting algorithm known for handling complex patterns in classification tasks. In your project, XGBoost refines melanoma detection by sequentially improving weak learners and reducing classification errors. Its ability to handle imbalanced datasets and optimize feature importance makes it one of the most effective models for skin cancer detection.

5.2. MODULES

5.2.1. DATASET MODULE

Description :-

The dataset module is responsible for loading and preprocessing skin cancer images from the dataset stored in Google Drive. The dataset is organized into two categories: benign and malignant images. This module plays a crucial role in ensuring the input images are properly structured before feeding them into the deep learning models.

Implementation Details :-

- **Google Drive Mounting:** The dataset is stored in Google Drive and accessed using : drive.mount('/content/drive')
- **Dataset_Path:** The path where the data set is located at .
- **Data Organization:** The dataset has two subdirectories - train , test.
- **Random Image Display:** A visualization function is used to display 12 random images from the training dataset.

Purpose :-

- Loads the dataset into the system.
- Ensures correct organization of benign and malignant samples.
- Visualizes sample images to verify the dataset integrity.

5.2.2. PREPROCESSING MODULE

Description :-

The preprocessing module is crucial for preparing the images before feeding them into deep learning models. This step ensures uniformity in image size, normalization, and augmentation to improve model performance and generalization.

Implementation Details :-

- Resizing Images: Since different models require a fixed input size, all images are resized to 224x224 pixels.
- Normalization: Pixel values are scaled to a range of [0,1] to speed up training and improve convergence.
- Data Augmentation: Techniques like rotation, flipping, and zooming are applied to artificially increase the dataset size and improve robustness.

Purpose :-

- **Resizing** ensures all images have a uniform size across different models.

- **Normalization** improves the learning process by making the model converge faster.
- **Data augmentation** prevents overfitting by introducing variations in the training set.

5.2.3. FEATURE EXTRACTION MODULE

Description :-

The feature extraction module is responsible for extracting meaningful patterns from the input images. This is achieved using multiple deep learning models such as VGG16, VGG19, ResNet, Vision Transformer (ViT), and Capsule Networks (CapsNet). Each model extracts deep features, which are later used for classification.

Implementation Details :-

- Pre-trained CNN Models: Transfer learning is applied using pre-trained architectures like VGG16, VGG19, and ResNet.
- Capsule Network (CapsNet): Captures spatial hierarchies in the images, offering an advantage over traditional CNNs.
- Vision Transformer (ViT): Utilizes self-attention mechanisms to extract global features.
- Flattening and Feature Vector Formation: The extracted feature maps are converted into 1D vectors for further processing.

Purpose :-

- VGG16 & VGG19: Capture deep spatial features through hierarchical convolutional layers.
- ResNet50: Uses residual connections to prevent vanishing gradient issues, ensuring better learning.
- CapsNet: Preserves spatial relationships between image features, improving classification.
- ViT: Uses self-attention mechanisms to capture long-range dependencies in images.

- Flattening: Converts extracted feature maps into a 1D format for use in machine learning classifiers

5.2.4. CLASSIFICATION MODULE

Description :-

The classification module takes the extracted feature vectors from multiple deep learning models (VGG16, VGG19, ResNet, Vision Transformer (ViT), and Capsule Networks (CapsNet)) and applies machine learning classifiers to categorize images as malignant or benign. The classifiers used include Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Classifier (SVC), Random Forest, and XGBoost.

Implementation Details :-

- Feature Concatenation: The extracted features from different deep learning models are combined to form a single feature vector.
- Application of Machine Learning Classifiers: The concatenated feature vectors are passed to different classifiers to predict the category.
- Majority Voting in Ensemble Learning: The final decision is taken based on majority voting among the classifiers.

Purpose :-

- Random Forest (RF): Uses multiple decision trees to improve generalization.
- XGBoost (XGB): A powerful gradient boosting algorithm known for high accuracy and efficiency.
- Support Vector Classifier (SVC): Uses the Radial Basis Function (RBF) kernel to create a hyperplane that maximizes class separation.
- K-Nearest Neighbors (KNN): Uses distance-based classification by analyzing the nearest neighbors.
- Logistic Regression (LR): A simple but effective linear classifier that helps in making binary predictions.

- Ensemble Learning: Combines predictions from multiple classifiers using soft voting to improve classification accuracy and robustness.

5.2.5. FINAL DECISION AND OUTPUT MODULE

Description :-

This module is responsible for aggregating the predictions from the ensemble classification model and determining the final classification of the skin lesion as malignant or benign. It processes the outputs of the machine learning classifiers and applies a majority voting mechanism to make the final decision. The results are then formatted for interpretation and visualized for better understanding.

Implementation Details :-

- Prediction Aggregation: The outputs from Logistic Regression, KNN, SVC, Random Forest, and XGBoost are combined using a soft voting mechanism, which assigns a probability score to each class and selects the class with the highest probability.
- Final Classification Decision: The final label (Malignant or Benign) is determined based on the highest probability.
- Output Visualization: The results are displayed in an interpretable format, showing confidence scores and classification results.

Purpose :-

- Soft Voting Mechanism: Aggregates classifier outputs and assigns final class labels.
- Accuracy Calculation: Measures the effectiveness of the ensemble classification model.
- Confusion Matrix: Provides insight into correct and incorrect predictions.
- Interpretability & Visualization: Helps users understand classification performance through reports and visual aids.

5.2.6. ENSEMBLE MODULE OR THE FINAL MODULE

Description :-

The ensembling module plays a crucial role in improving the accuracy and robustness of melanoma classification. Instead of relying on a single model, multiple classifiers are combined using ensemble techniques to enhance decision-making. This module utilizes a Voting Classifier approach, followed by a higher-level "Ensemble of Ensembles" mechanism, which integrates multiple ensemble models for even better performance.

Why Ensembling ?

- Individual models may overfit or underperform on certain samples.
- Combining different models leverages their strengths while minimizing weaknesses.
- Ensembling techniques increase stability and reduce misclassification errors.
- It ensures generalization across different melanoma types, leading to a more reliable classifier.

Step 1: Base Classifiers

This step involves training individual machine learning models that will be used in the ensemble. These classifiers take input features extracted from deep learning models like VGG16, VGG19, ResNet, Vision Transformer (ViT), and Capsule Networks (CapsNet) and classify them as benign or malignant.

Classifiers Used

1. Random Forest (RF): Uses multiple decision trees to improve generalization.
2. XGBoost: A powerful boosting algorithm that refines weak learners.
3. Support Vector Classifier (SVC): Finds the optimal hyperplane for class separation.
4. K-Nearest Neighbors (KNN): Classifies images based on similarity

to nearby samples.

5. Logistic Regression (LR): Serves as a baseline linear classifier.

Step 2: Creating the Voting Classifier (First-Level Ensemble)

What is a Voting Classifier?

A Voting Classifier combines multiple classifiers and aggregates their predictions to improve overall accuracy. Two types of voting strategies are used:

- Hard Voting: Each model votes, and the majority class is chosen.
- Soft Voting (Used Here): Models provide probabilities, and the class with the highest combined probability is selected.

Why Soft Voting?

- Soft voting considers probability scores, making the final decision more informed.
- Reduces bias that might occur when using a single model's prediction.
- Improves accuracy by combining different classifiers' probability distributions.

Step 3: The "Ensemble of Ensembles" (Second-Level Ensembling)

What is an "Ensemble of Ensembles"?

The "Ensemble of Ensembles" approach takes multiple ensemble models trained on different feature extractors and combines their outputs. This technique further improves classification performance by leveraging diverse feature representations.

Process Flow:

1. Train separate ensemble models for different feature extractors (VGG, ResNet, ViT, etc.).
2. Each ensemble model makes predictions independently.
3. Final ensemble aggregates predictions from all ensemble models.

Why This Works?

- Combining different ensemble models reduces overfitting to specific feature extractors.
- Diversity in decision-making improves generalization across different skin lesion types.
- Layered ensembling enhances prediction confidence by reinforcing multiple models' strengths.

Step 4: Evaluating the Ensemble Performance

After ensembling, the model's performance is assessed using standard evaluation metrics such as accuracy, precision, recall, F1-score, and a confusion matrix.

Interpreting Results:

- Higher accuracy indicates the strength of ensembling.
- Confusion matrix provides insights into misclassification rates.
- Classification report breaks down performance by precision, recall, and F1-score.

Step 5: Saving the Trained Ensemble Model

To make the trained model reusable, it is saved as a pickle (.pkl) file.

Final Summary

- First-level ensemble (Voting Classifier) combines multiple machine learning models.
- Second-level ensemble ("Ensemble of Ensembles") merges multiple ensemble models.
- Soft voting ensures a probability-based decision-making process.
- Evaluation metrics confirm improved performance.
- Model is saved for further use in melanoma classification.

This ensembling technique provides state-of-the-art accuracy by leveraging multiple models at different levels, ensuring better classification reliability and robustness.

5.3 UML DIAGRAMS

To effectively document the melanoma detection system, we need to visualize the system architecture, interactions, and data flow using Unified Modeling Language (UML) diagrams. These diagrams help in understanding the structural and behavioral aspects of the system, ensuring a clear representation of feature extraction, classification, and result generation processes. The key UML diagrams for this project include Use Case Diagram, Class Diagram, Activity Diagram, Sequence Diagram, and Deployment Diagram. Each of these diagrams plays a crucial role in defining different aspects of the melanoma detection system.

USE CASE DIAGRAM

The Use Case Diagram represents the interaction between the user (doctor/researcher) and the melanoma detection system. It provides a high-level overview of the functionalities available in the system. The main actor in this system is the Doctor or Researcher, who interacts with the melanoma detection system to perform various tasks.

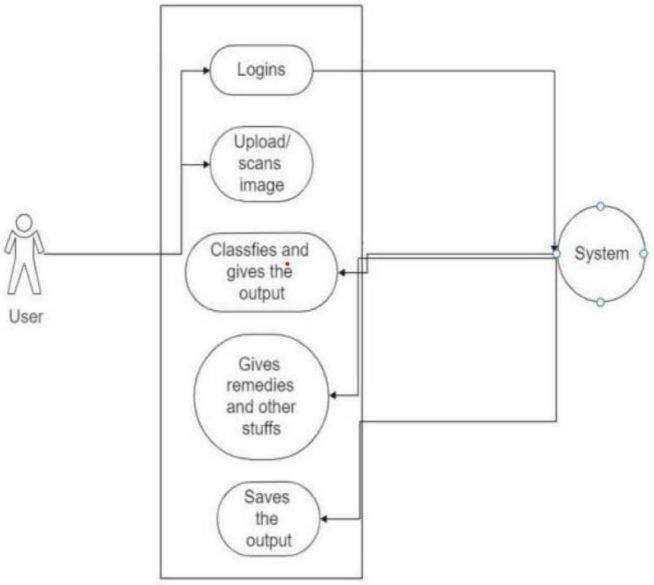


Fig.4 : Use case diagram for skin cancer detection

The primary use cases of this system include Uploading an Image, where the user provides a skin lesion image for analysis. Once uploaded, the system automatically Preprocesses the Image, applying techniques such as resizing, color conversion, and normalization. After preprocessing, the system extracts deep features using pre-trained models like VGG19, ResNet50, ViT, and CapsNet. These extracted features are then sent to the Machine Learning Classifier, which predicts whether the lesion is benign or malignant. The system then Displays the Results to the user, providing accuracy scores and relevant classification metrics. Additionally, the system allows the user to Download the Report, enabling them to store the classification results for future reference.

This use case diagram highlights how users interact with the system and the core functionalities it offers in the melanoma detection process.

CLASS DIAGRAM

The Class Diagram represents the structure of the system by defining various components, their attributes, and methods. It provides insight into how different objects interact within the melanoma detection pipeline.

The core class in the system is the ImageProcessor, which is responsible for handling image input, preprocessing steps such as resizing and normalization, and preparing the data for feature extraction. The FeatureExtractor class is designed to apply deep learning models such as VGG19, VGG16, ResNet50, CapsNet, and ViT to extract high-level image features. These extracted features are passed to the Classifier class, which employs machine learning algorithms like Support Vector Machine (SVC), XGBoost, Random Forest, and Logistic Regression to determine the melanoma classification.

The UserInterface class handles interactions between the system and the user, including uploading images, displaying results, and generating reports. Finally, the ReportGenerator class is responsible for saving and exporting classification results, allowing users to keep track of predictions and analysis. The relationships between these classes ensure smooth data flow from image input to classification and result storage.

ACTIVITY DIAGRAM

The Activity Diagram outlines the sequence of steps involved in processing and classifying a skin lesion image. This diagram provides a step-by-step representation of the workflow, ensuring clarity in the execution of each operation.

The process starts when the user uploads an image into the system. The Image Processor then preprocesses the image by applying necessary transformations such as resizing, normalization, and color space conversion. Once the preprocessing is complete, the Feature Extractor applies deep learning models to extract meaningful features. These extracted features are then passed to the Machine Learning Classifier, which processes the data and predicts whether the given image represents a benign or malignant lesion.

Once the classification is done, the system displays the results to the user, showing accuracy, confidence scores, and classification labels. The user then has the option to download the classification report, ensuring that the results can be stored for future medical reference. The activity diagram effectively visualizes how data flows through the melanoma detection system, from input to classification and final output generation.

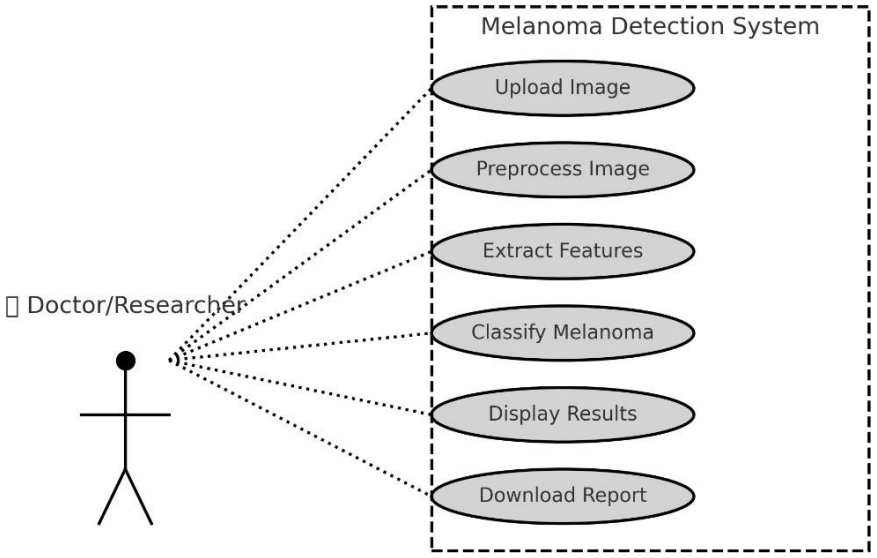


Fig.5 : Use case diagram for the process of detection of melanoma

Sequence Diagram (Melanoma Detection Process)

The Sequence Diagram provides a detailed representation of the step-by-step interactions between different system components during the melanoma detection process. It highlights the order of execution and message exchanges between various modules.

The interaction begins when the user uploads an image to the system. The User Interface sends the uploaded image to the Image Processor, which applies preprocessing techniques such as resizing, normalization, and color conversion. The processed image is then forwarded to the Feature Extractor, which uses deep learning models such as VGG19, ResNet50, ViT, and CapsNet to extract meaningful image features.

Once the features are extracted, they are passed to the Classifier, which employs machine learning techniques such as SVC, XGBoost, and Random Forest to determine whether the lesion is benign or malignant. The classification results are then returned to the User Interface, where the output is displayed. Additionally, the system allows the user to download a detailed report of the classification, including accuracy scores, confidence levels, and extracted feature details.

This sequence diagram clearly illustrates how different system components interact in a sequential manner, ensuring efficient and accurate melanoma classification.

Deployment Diagram (System Architecture)

The Deployment Diagram represents the hardware and software infrastructure required for deploying the melanoma detection system. It highlights how different system components are hosted and interact with each other in a real-world deployment scenario.

The system consists of a User Device (such as a PC, laptop, or mobile phone) that provides access to a Web-Based Interface. This interface communicates with a Web Server, which processes image upload requests and forwards them to the Deep Learning Server. The Deep Learning Server, equipped with GPU acceleration, executes the feature extraction models such as VGG19, ViT, ResNet50, and CapsNet, ensuring efficient processing of large image datasets.

CHAPTER 6

IMPLEMENTATION

Model implementation is the process of transforming theoretical concepts and research findings into a functional system that can be executed in a real-world environment. In this study, the implementation involves developing a hybrid deep learning and machine learning model for melanoma classification. This includes dataset preprocessing, feature extraction using deep learning architectures, classification using machine learning models, and performance evaluation to ensure robustness and accuracy.

The goal of the implementation phase is to create a robust and scalable model that can effectively distinguish between benign and malignant skin lesions. This is achieved by leveraging multiple deep learning feature extractors, combining their outputs with machine learning classifiers, and optimizing their performance through ensemble learning techniques. This section provides an in-depth explanation of each stage involved in the model implementation process.

6.1. MODEL IMPLEMENTATION

6.1.1. Dataset Collection

The dataset used in this study was obtained from Kaggle, specifically the "Melanoma Skin Cancer Dataset of 10,000 Images." This dataset consists of 10,000 high-resolution dermoscopic images, categorized into malignant and benign classes. The dataset was split into:

- Training Set: 9,605 images (5,773 benign, 3,832 malignant)
- Testing Set: 1,000 images (500 benign, 500 malignant)

Importance of Dataset Selection

A high-quality dataset is crucial for training a reliable classification model. The dataset used contains images with varying illumination, skin tone variations, and lesion sizes, making it well-suited for developing a generalized melanoma detection system. Additionally, the dataset was balanced to prevent bias toward one class over the other.

Data Augmentation Techniques

To enhance the dataset and improve model generalization, the following augmentation techniques were applied:

- Rotation: Random rotations to simulate real-world variations in image capture.
- Flipping: Horizontal and vertical flips to increase diversity.
- Zooming: Adjusting zoom levels to improve robustness.
- Brightness Adjustment: Modifying brightness levels to simulate different lighting conditions.
- Gaussian Noise: Adding noise to make the model more resilient to image distortions.

Data Preprocessing

Proper preprocessing is critical to ensure the model learns meaningful features rather than noise. The preprocessing pipeline consists of:

1. **Resizing:** Images were uniformly resized to 128x128 pixels to maintain consistency across different models and reduce computational complexity.
2. **Colorspace Conversion:** OpenCV was used to convert images from BGR to RGB to align with deep learning framework requirements.
3. **Shuffling:** The dataset was shuffled to prevent the model from learning unintended patterns based on image order.
4. **Normalization:** Pixel values were scaled to the range [0,1] to standardize the input distribution and improve convergence during training.
5. **Label Encoding:** The categorical labels ('benign' and 'malignant') were converted into numerical form (0 for benign and 1 for malignant) to be compatible with machine learning models.
6. **Train-Test Split:** The dataset was divided into 80% training and 20% testing, ensuring a balance between training efficiency and evaluation accuracy.

6.1.2. Model Architecture

The proposed hybrid model integrates deep learning feature extraction with machine learning classification. It combines multiple CNN architectures and a Vision Transformer (ViT) for feature extraction, followed by an ensemble of traditional machine learning classifiers.

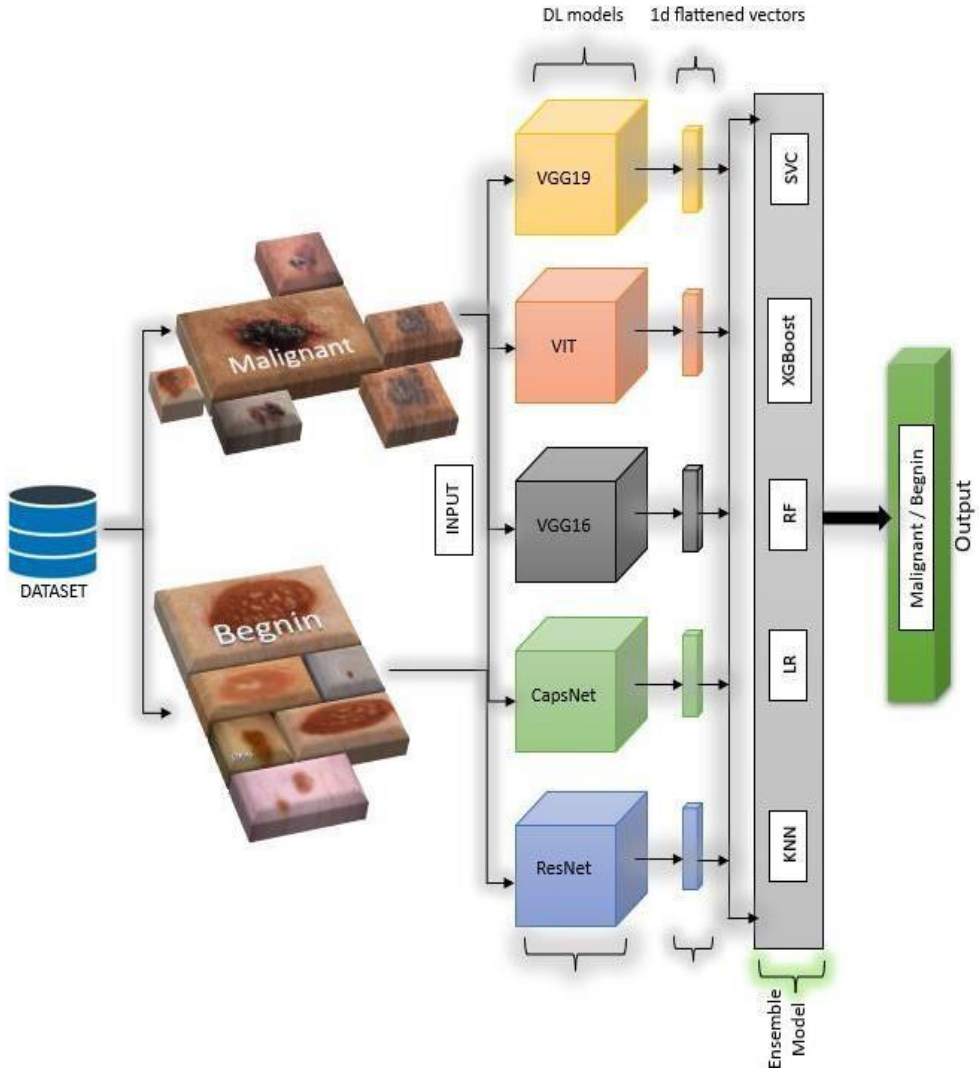


Fig.6 : Architecture of the Proposed Model

Deep Learning Feature Extractors

1. VGG16 & VGG19: These CNN architectures extract hierarchical features from dermoscopic images, providing strong spatial representation.
2. ResNet50: Uses residual learning to enhance feature extraction efficiency and prevent vanishing gradient issues.

3. Capsule Network (CapsNet): Captures spatial relationships between image features, improving lesion recognition.
4. Vision Transformer (ViT): Leverages self-attention mechanisms to capture global dependencies in images, significantly improving classification performance.

The extracted feature embeddings from these architectures are flattened and passed to machine learning classifiers for final classification.

Machine Learning Classifiers

The extracted features are processed by the following machine learning classifiers:

1. **Logistic Regression:** A simple linear model for binary classification.
2. **Random Forest:** An ensemble of decision trees that enhances prediction stability and reduces overfitting.
3. **K-Nearest Neighbors (KNN):** A non-parametric model that classifies based on feature similarity.
4. **Support Vector Classifier (SVC):** Employs a kernel-based approach to maximize the margin between classes.
5. **XGBoost:** A powerful gradient-boosted decision tree model optimized for high accuracy.

Each classifier was fine-tuned using hyperparameter optimization to achieve optimal performance.

Ensemble Learning via Majority Voting

To improve classification accuracy, predictions from all classifiers were aggregated using a majority voting ensemble method. The ensemble approach combines the strengths of multiple classifiers while reducing the weaknesses of individual models. This ensures better generalization, improved stability, and enhanced robustness of predictions.

6.1.3. Model Training & Hyperparameter Tuning

The model was trained using the following hyperparameters:

- **Batch Size:** 32
- **Optimizer:** Adam (learning rate = 0.0001)
- **Regularization:** L2 regularization (0.0005) and dropout (0.1) to prevent overfitting.
- **Loss Function:** Sparse Categorical Cross-Entropy
- **Epochs:** 20
- **Feature Standardization:** Using StandardScaler to normalize extracted features before classification.

During training, extracted feature embeddings were fed into the machine learning classifiers. Each classifier underwent hyperparameter tuning to optimize its performance. Early stopping and validation monitoring were used to prevent overfitting and ensure the model generalizes well to unseen data.

The Vision Transformer (ViT) achieved the highest accuracy of **92.4%**, outperforming individual CNN models. The ensemble model performed competitively, achieving **92.3%** accuracy, confirming its effectiveness in melanoma detection.

6.1.4. Model Performance Evaluation

The models were evaluated using multiple metrics, including:

- **Accuracy:** Measures overall classification correctness.
- **Precision:** Indicates how many positively predicted cases were actually positive.
- **Recall:** Determines the model's ability to identify actual positive cases.
- **F1-Score:** Harmonic mean of precision and recall to provide a balanced evaluation.

6.1.5. Confusion Matrix & Visualization

To further analyze performance, confusion matrices were generated for each model, highlighting classification strengths and misclassifications. Additionally, t-SNE (t-Distributed Stochastic Neighbor Embedding) visualizations were used to assess the feature space distribution and clustering patterns of benign and malignant lesions.

These visualizations provide deeper insights into how well the models differentiate between the two classes and suggest potential improvements in feature extraction and classification techniques.

The proposed hybrid deep learning and machine learning approach significantly enhances melanoma detection accuracy. By leveraging multiple feature extraction techniques and integrating an ensemble learning strategy, the model achieves state-of-the-art classification performance.

The results confirm that ensemble methods outperform standalone deep learning models, making the system highly reliable for real-world melanoma diagnosis applications. Future work can focus on integrating additional image augmentation techniques, refining hyperparameters through extensive grid search optimization, and incorporating domain-specific knowledge to further improve classification accuracy.

6.2. CODING

Colab Code :-

```
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd
dataset_path='/content/drive/MyDrive/melanoma_cancer_dataset'
!ls "/content/drive/My Drive/melanoma_cancer_dataset"
import os
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import random
```

```
!pip install xgboost
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score, precision_score, recall_score, f1_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.svm import SVC
import xgboost as xgb
import matplotlib.pyplot as plt
import seaborn as sns
from keras.applications import VGG19
from keras.layers import Input, Flatten, Dense
from keras.models import Model
from keras.optimizers import Adam
from sklearn.manifold import TSNE
train_folder = os.path.join(dataset_path, 'train')
image_files = []
for subdir in ['benign', 'malignant']:
    subdir_path = os.path.join(train_folder, subdir)
    image_files.extend([os.path.join(subdir_path, f) for f in
os.listdir(subdir_path) if if.endswith(('.jpg', '.jpeg', '.png'))])
selected_files = random.sample(image_files, 12)
plt.figure(figsize=(12, 9))
for i, file in enumerate(selected_files):
    plt.subplot(3, 4, i + 1)
    img = mpimg.imread(file)
    plt.imshow(img)
    plt.axis('off')
plt.tight_layout()
plt.show()
import os
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import random
test_folder = os.path.join(dataset_path, 'test')
image_files = []
for subdir in ['benign', 'malignant']:
```

```
    subdir_path = os.path.join(test_folder, subdir)
    image_files.extend([os.path.join(subdir_path, f) for f in
os.listdir(subdir_path) if f.endswith(('.jpg', '.jpeg', '.png'))])
selected_files = random.sample(image_files, 12)
plt.figure(figsize=(12, 9))
for i, file in enumerate(selected_files):
    plt.subplot(3, 4, i + 1)
    img = mpimg.imread(file)
    plt.imshow(img)
    plt.axis('off')
plt.tight_layout()
plt.show()
import numpy as np
import matplotlib.pyplot as plt
import glob
import cv2 as cv
from keras.models import Model, Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization
import os
import seaborn as sns
from keras.applications.vgg19 import VGG19
from keras.applications.efficientnet import EfficientNetB0
from tensorflow.keras.applications.mobilenet import MobileNet
from sklearn.ensemble import VotingClassifier
def input_data(folder_path, output_data):
    for dirs in os.listdir(folder_path):
        class_name = dirs
        new_path = os.path.join(folder_path, class_name)
        for img in os.listdir(new_path):
            img_arr = cv.imread(os.path.join(new_path, img),
cv.IMREAD_COLOR)
            resize = cv.resize(img_arr, (128,128))
            img = cv.cvtColor(img_arr, cv.COLOR_RGB2BGR)
            output_data.append([resize, class_name])
    return output_data
train_data =
input_data(r"/content/drive/MyDrive/melanoma_cancer_dataset/train",   [])
test_data =
input_data(r"/content/drive/MyDrive/melanoma_cancer_dataset/test", [])
```

```
np.random.shuffle(train_data)
np.random.shuffle(test_data)
X_train = []
Y_train = []
for features, labels in train_data:
    X_train.append(features)
    Y_train.append(labels)
X_test = []
Y_test = []
for features, labels in test_data:
    X_test.append(features)
    Y_test.append(labels)
test_images = np.array(X_test)
test_labels = np.array(Y_test)
train_images = np.array(X_train)
train_labels = np.array(Y_train)
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(test_labels)
test_labels_encoded = le.transform(test_labels)
le.fit(train_labels)
train_labels_encoded = le.transform(train_labels)
x_train, y_train, x_test, y_test = train_images, train_labels_encoded,
test_images, test_labels_encoded
y_train[10:20]
x_train.shape
labels=np.unique(y_test)
print(labels)
!pip install tabulate
import numpy as np
import matplotlib.pyplot as plt
train_images = np.array(X_train[:12])
fig, axes = plt.subplots(3, 4, figsize=(10, 6))
for i, ax in enumerate(axes.flat):
    ax.imshow(train_images[i])
    ax.axis('off')
plt.suptitle('First 12 Images from Train Folder in Melanoma Cancer
Dataset', fontsize=16)
plt.show()
import matplotlib.pyplot as plt
import numpy as np
```

```
test_images = np.array(X_test[:12])
fig, axes = plt.subplots(3, 4, figsize=(10, 6))
for i, ax in enumerate(axes.flat):
    ax.imshow(test_images[i])
    ax.axis('off')
plt.suptitle('First 12 Images from Test Folder in Melanoma Cancer Dataset',
fontsize=16)
plt.show()
def evaluate_model(y_true, y_pred, labels, model_name):
    print(f"Model: {model_name}")
    print(f"Accuracy: {accuracy_score(y_true, y_pred)}")
    print(f"Precision: {precision_score(y_true, y_pred, average='micro')}")
    print(f"Recall: {recall_score(y_true, y_pred, average='micro')}")
    print(f"F1 Score: {f1_score(y_true, y_pred, average='micro')}")
    cm = confusion_matrix(y_true, y_pred)
    fig = plt.figure(figsize=(7, 7))
    ax = plt.subplot()
    sns.heatmap(cm, annot=True, ax=ax, cmap="Blues", fmt="g")
    ax.set_title(f'Confusion Matrix for {model_name}')
    ax.set_xlabel('Predicted', fontsize=15)
    ax.xaxis.set_label_position('bottom')
    plt.xticks(rotation=90)
    ax.xaxis.set_ticklabels(labels, fontsize=15)
    ax.xaxis.tick_bottom()
    ax.set_ylabel('True', fontsize=15)
    ax.yaxis.set_ticklabels(labels, fontsize=15)
    plt.yticks(rotation=0)
    plt.show()
from keras.layers import GlobalAveragePooling2D
import keras
base_model = VGG19(weights='imagenet', include_top=False,
input_shape=(128, 128, 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
vgg_model = keras.models.Model(inputs=base_model.input, outputs=x)
for layer in base_model.layers:
    layer.trainable = True
print(vgg_model.summary())
feature_extractor___= vgg_model.predict(x_train)
fe___test = vgg_model.predict(x_test)
```

```
features__= feature_extractor_.reshape(feature_extractor_.shape[0], -1)
f___test = fe___test.reshape(fe___test.shape[0], -1)
print(f___test.shape)
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features_)
scaled_test_features = scaler.transform(f___test)
print("Standardized features shape:", scaled_features.shape)
print("Standardized test features shape:", scaled_test_features.shape)
labels = ['Class 1', 'Class 2']
results = {}
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(features_, y_train)
y_pred_knn = knn.predict(f___test)
if y_pred_knn.ndim > 1 and y_pred_knn.shape[1] > 1:
    y_pred_knn = np.argmax(y_pred_knn, axis=1)
evaluate_model(y_test, y_pred_knn, labels, "K-Nearest Neighbors")
results['KNN'] = [accuracy_score(y_test, y_pred_knn),
precision_score(y_test, y_pred_knn, average='micro'),
                  recall_score(y_test, y_pred_knn, average='micro'),
f1_score(y_test, y_pred_knn, average='micro')]
lrm = LogisticRegression(multi_class='multinomial', solver='lbfgs')
lrm.fit(features_, y_train)
y_pred_lr = lrm.predict(f___test)
evaluate_model(y_test, y_pred_lr, labels, "Logistic Regression")
results['Logistic Regression'] = [accuracy_score(y_test, y_pred_lr),
precision_score(y_test, y_pred_lr, average='micro'), recall_score(y_test,
y_pred_lr, average='micro'), f1_score(y_test, y_pred_lr, average='micro')]
RF_model = RandomForestClassifier(n_estimators=50, random_state=42)
RF_model.fit(features_, y_train)
y_pred_rf = RF_model.predict(f___test)
evaluate_model(y_test, y_pred_rf, labels, "Random Forest")
results['Random Forest'] = [accuracy_score(y_test, y_pred_rf),
precision_score(y_test, y_pred_rf, average='micro'), recall_score(y_test,
y_pred_rf, average='micro'), f1_score(y_test, y_pred_rf, average='micro')]
xgb_classifier = xgb.XGBClassifier(tree_method="auto")
xgb_classifier.fit(features_, y_train)
y_pred_xgb = xgb_classifier.predict(f___test)
evaluate_model(y_test, y_pred_xgb, labels, "XGBoost")
results['XGBoost'] = [accuracy_score(y_test, y_pred_xgb),
precision_score(y_test, y_pred_xgb, average='micro'), recall_score(y_test,
y_pred_xgb, average='micro'), f1_score(y_test, y_pred_xgb,
average='micro')]
```

```
svc = SVC(C=0.65, random_state=0, kernel='rbf')
svc.fit(features_, y_train)
y_pred_svc = svc.predict(f___test)
evaluate_model(y_test, y_pred_svc, labels, "SVC")
results['SVC'] = [accuracy_score(y_test, y_pred_svc),
precision_score(y_test, y_pred_svc, average='micro'), recall_score(y_test,
y_pred_svc, average='micro'), f1_score(y_test, y_pred_svc,
average='micro')]
final_model = VotingClassifier(estimators=[
    ('rf', RF_model),
    ('xgb', xgb_classifier),
    ('knn', knn),
    ('svc', svc),
    ('lr', lrm)
], voting='hard')
final_model.fit(features_, y_train)
y_pred_final = final_model.predict(f___test)
evaluate_model(y_test, y_pred_final, labels, "Ensemble Voting Classifier")
results['Ensemble'] = [accuracy_score(y_test, y_pred_final),
precision_score(y_test, y_pred_final, average='micro'), recall_score(y_test,
y_pred_final, average='micro'), f1_score(y_test, y_pred_final,
average='micro')]
import tabulate
table = [
    ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'],
    ['KNN', results['KNN'][0], results['KNN'][1], results['KNN'][2],
results['KNN'][3]],
    ['Logistic Regression', results['Logistic Regression'][0], results['Logistic
Regression'][1], results['Logistic Regression'][2], results['Logistic
Regression'][3]],
    ['Random Forest', results['Random Forest'][0], results['Random
Forest'][1], results['Random Forest'][2], results['Random Forest'][3]],
    ['XGBoost', results['XGBoost'][0], results['XGBoost'][1],
results['XGBoost'][2], results['XGBoost'][3]],
    ['SVC', results['SVC'][0], results['SVC'][1], results['SVC'][2],
results['SVC'][3]],
    ['Ensemble', results['Ensemble'][0], results['Ensemble'][1],
results['Ensemble'][2], results['Ensemble'][3]],
]
print(tabulate.tabulate(table, headers='firstrow', tablefmt='grid'))
from tabulate import tabulate
table_data = [
```

```
    ['Parameter', 'Value'],
    ['Input Shape', '(128, 128, 3)'],
    ['Base Model', 'VGG19'],
    ['Trainable Layers', 'True'],
    ['Feature Extraction Shape', features_.shape],
    ['Standardized Features Shape', scaled_features.shape],
    ['Number of Classes', len(labels)],
    ['KNN Neighbors', 10],
    ['Random Forest Estimators', 50],
    ['XGBoost Tree Method', "auto"],
    ['SVC Kernel', 'rbf'],
    ['SVC C', 0.65],
    ['Voting Classifier', 'Hard'],
]
print(tabulate(table_data, headers='firstrow', tablefmt='grid'))
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
tsne = TSNE(n_components=2, learning_rate='auto',
            init='random', perplexity=3)
X_reduced = tsne.fit_transform(scaled_features)
plt.figure(figsize=(10, 7))
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=y_train, cmap='jet')
plt.title('t-SNE visualization of VGG19 features')
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.show()
import pickle
with open(r"/content/drive/MyDrive/melanoma_cancer_dataset/model.pkl",
"wb") as f:
    pickle.dump(final_model, f)
matplotlib.pyplot as plt
import numpy as np
from keras.applications.vgg16 import VGG16
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import xgboost as xgb
from sklearn.manifold import TSNE
import pickle
from tabulate import tabulate
base_model = VGG16(weights='imagenet', include_top=False,
input_shape=(128, 128, 3))
```

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
vgg_model = Model(inputs=base_model.input, outputs=x)
for layer in base_model.layers:
    layer.trainable = True
print(vgg_model.summary())
feature_extractor___= vgg_model.predict(x_train)
fe___test = vgg_model.predict(x_test)
features___= feature_extractor_.reshape(feature_extractor_.shape[0], -1)
f___test = fe___test.reshape(fe___test.shape[0], -1)
print(f___test.shape)
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features_)
scaled_test_features = scaler.transform(f___test)
print("Standardized features shape:", scaled_features.shape)
print("Standardized test features shape:", scaled_test_features.shape)
labels = ['Class 1', 'Class 2']
results = {}
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(features_, y_train)
y_pred_knn = knn.predict(f___test)
if y_pred_knn.ndim > 1 and y_pred_knn.shape[1] > 1:
    y_pred_knn = np.argmax(y_pred_knn, axis=1)
evaluate_model(y_test, y_pred_knn, labels, "K-Nearest Neighbors")
results['KNN'] = [accuracy_score(y_test, y_pred_knn),
precision_score(y_test, y_pred_knn, average='micro'),
                recall_score(y_test, y_pred_knn, average='micro'),
f1_score(y_test, y_pred_knn, average='micro')]
lrm = LogisticRegression(multi_class='multinomial', solver='lbfgs')
lrm.fit(features_, y_train)
y_pred_lr = lrm.predict(f___test)
evaluate_model(y_test, y_pred_lr, labels, "Logistic Regression")
results['Logistic Regression'] = [accuracy_score(y_test, y_pred_lr),
precision_score(y_test, y_pred_lr, average='micro'), recall_score(y_test,
y_pred_lr, average='micro'), f1_score(y_test, y_pred_lr, average='micro')]
RF_model = RandomForestClassifier(n_estimators=50, random_state=42)
RF_model.fit(features_, y_train)
y_pred_rf = RF_model.predict(f___test)
evaluate_model(y_test, y_pred_rf, labels, "Random Forest")
results['Random Forest'] = [accuracy_score(y_test, y_pred_rf),
precision_score(y_test, y_pred_rf, average='micro'), recall_score(y_test,
```

```
y_pred_rf, average='micro'), f1_score(y_test, y_pred_rf, average='micro')]
# XGBoost
xgb_classifier = xgb.XGBClassifier(tree_method="auto")
xgb_classifier.fit(features_, y_train)
y_pred_xgb = xgb_classifier.predict(f___test)
evaluate_model(y_test, y_pred_xgb, labels, "XGBoost")
results['XGBoost'] = [accuracy_score(y_test, y_pred_xgb),
precision_score(y_test, y_pred_xgb, average='micro'), recall_score(y_test,
y_pred_xgb, average='micro'), f1_score(y_test, y_pred_xgb,
average='micro')]
#SVC
svc = SVC(C=0.65, random_state=0, kernel='rbf')
svc.fit(features_, y_train)
y_pred_svc = svc.predict(f___test)
evaluate_model(y_test, y_pred_svc, labels, "SVC")
results['SVC'] = [accuracy_score(y_test, y_pred_svc),
precision_score(y_test, y_pred_svc, average='micro'), recall_score(y_test,
y_pred_svc, average='micro'), f1_score(y_test, y_pred_svc,
average='micro')]
# Ensemble Voting Classifier
final_model_0 = VotingClassifier(estimators=[
    ('rf', RF_model),
    ('xgb', xgb_classifier),
    ('knn', knn),
    ('svc', svc),
    ('lr', lrm)
], voting='hard')
final_model.fit(features_, y_train)
y_pred_final = final_model.predict(f___test)
evaluate_model(y_test, y_pred_final, labels, "Ensemble Voting Classifier")
results['Ensemble'] = [accuracy_score(y_test, y_pred_final),
precision_score(y_test, y_pred_final, average='micro'), recall_score(y_test,
y_pred_final, average='micro'), f1_score(y_test, y_pred_final,
average='micro')]
from tabulate import tabulate
# Define the table data
table_data = [
    ['Parameter', 'Value'],
    ['Input Shape', '(128, 128, 3)'],
    ['Base Model', 'VGG16'],
    ['Trainable Layers', 'True'],
    ['Feature Extraction Shape', features__.shape],
```

```
    ['Standardized Features Shape', scaled_features.shape],
    ['Number of Classes', len(labels)],
    ['KNN Neighbors', 10],
    ['Random Forest Estimators', 50],
    ['XGBoost Tree Method', "auto"],
    ['SVC Kernel', 'rbf'],
    ['SVC C', 0.65],
    ['Voting Classifier', 'Hard'],
]
# Create and print the table
print(tabulate(table_data, headers='firstrow', tablefmt='grid'))
!pip install tabulate
from tabulate import tabulate # Make sure to import the 'tabulate' function from the 'tabulate' module
# Create the table
table = [
    ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'],
    ['KNN', results['KNN'][0], results['KNN'][1], results['KNN'][2], results['KNN'][3]],
    ['Logistic Regression', results['Logistic Regression'][0], results['Logistic Regression'][1], results['Logistic Regression'][2], results['Logistic Regression'][3]],
    ['Random Forest', results['Random Forest'][0], results['Random Forest'][1], results['Random Forest'][2], results['Random Forest'][3]],
    ['XGBoost', results['XGBoost'][0], results['XGBoost'][1], results['XGBoost'][2], results['XGBoost'][3]],
    ['SVC', results['SVC'][0], results['SVC'][1], results['SVC'][2], results['SVC'][3]],
    ['Ensemble', results['Ensemble'][0], results['Ensemble'][1], results['Ensemble'][2], results['Ensemble'][3]],
]
# Print the table
print(tabulate(table, headers='firstrow', tablefmt='grid'))
# Define the table data
table_data = [
    ['Parameter', 'Value'],
    ['Input Shape', '(128, 128, 3)'],
    ['Base Model', 'VGG16'],
    ['Trainable Layers', 'False'],
    ['Feature Extraction Shape', features_.shape],
    ['Standardized Features Shape', scaled_features.shape],
    ['Number of Classes', len(labels)],
```

```
    ['KNN Neighbors', 10],
    ['Random Forest Estimators', 50],
    ['XGBoost Tree Method', "auto"],
    ['SVC Kernel', 'rbf'],
    ['SVC C', 0.65],
    ['Voting Classifier', 'Hard'],
]
# Create and print the table
print(tabulate(table_data, headers='firstrow', tablefmt='grid'))
# Reduce dimensionality with TSNE
tsne = TSNE(n_components=2, learning_rate='auto',
        init='random', perplexity=3)
X_reduced = tsne.fit_transform(scaled_features)
# Plot the results
plt.figure(figsize=(10, 7))
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=y_train, cmap='jet')
plt.title('t-SNE visualization of VGG16 features')
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.show()
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score, precision_score, recall_score, f1_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.svm import SVC
import xgboost as xgb
from keras.layers import Input, Conv2D, Reshape, Flatten, Dense
from keras.models import Model
from keras.optimizers import Adam
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
!cp "/content/drive/MyDrive/Colab Notebooks/Capsnet.py" "/content"
# Initialize and load the CapsNet model
from Capsnet import CapsNet
```

```
capsnet_model = CapsNet(input_shape=(128, 128, 3), num_classes=3)
# Function to evaluate and plot confusion matrix
def evaluate_model(y_true, y_pred, labels, model_name):
    print(f"Model: {model_name}")
    print(f"Accuracy: {accuracy_score(y_true, y_pred)}")
    print(f"Precision: {precision_score(y_true, y_pred, average='micro')}")
    print(f"Recall: {recall_score(y_true, y_pred, average='micro')}")
    print(f"F1 Score: {f1_score(y_true, y_pred, average='micro')}")
    cm = confusion_matrix(y_true, y_pred)
    fig = plt.figure(figsize=(7, 7))
    ax = plt.subplot()
    sns.heatmap(cm, annot=True, ax=ax, cmap="Blues", fmt="g")
    ax.set_title(f'Confusion Matrix for {model_name}')
    ax.set_xlabel('Predicted', fontsize=15)
    ax.xaxis.set_label_position('bottom')
    plt.xticks(rotation=90)
    ax.xaxis.set_ticklabels(labels, fontsize=15)
    ax.xaxis.tick_bottom()
    ax.set_ylabel('True', fontsize=15)
    ax.yaxis.set_ticklabels(labels, fontsize=15)
    plt.yticks(rotation=0)
    plt.show()
!pip install tensorflow
import tensorflow as tf
# Initialize and load the CapsNet model
from Capsnet import CapsNet
capsnet_model = CapsNet(input_shape=(128, 128, 3), num_classes=3)
# Make layers non-trainable
for layer in capsnet_model.layers:
    layer.trainable = True
# Modify the final layer of the CapsNet to match the label shape
flattened_output = tf.keras.layers.Flatten()(capsnet_model.layers[-2].output)
# Flatten the output of the second last layer
output_layer = tf.keras.layers.Dense(3,
activation='softmax')(flattened_output) # Add a dense layer with softmax
activation for 3 classes
# Recreate the model with the modified layers
capsnet_model = tf.keras.models.Model(inputs=capsnet_model.input,
outputs=output_layer)
# Compile the model with appropriate loss and optimizer
capsnet_model.compile(loss='categorical_crossentropy', optimizer='adam',
```

metrics=['accuracy'])

 # One-hot encode the target labels

y_train_encoded = tf.keras.utils.to_categorical(y_train, num_classes=3)

#Assuming 3 classes

 # Train the CapsNet model with one-hot encoded labels

capsnet_model.fit(x_train, y_train_encoded, epochs=20, batch_size=32)

 # One-hot encode the test labels as well

y_test_encoded = tf.keras.utils.to_categorical(y_test, num_classes=3)

 # Evaluate the CapsNet model on the test data using one-hot encoded labels

accuracy = capsnet_model.evaluate(x_test, y_test_encoded)[1] print("Accuracy:", accuracy)

 # Extract features from the trained CapsNet model

feature_extractor___= capsnet_model.predict(x_train) fe

_____test = capsnet_model.predict(x_test)

features___= feature_extractor_.reshape(feature_extractor_.shape[0], -1)

f___test = fe___test.reshape(fe___test.shape[0], -1)

 # Standardize the features

scaler = StandardScaler()

features___= scaler.fit_transform(features_)

f___test = scaler.transform(f___test)

 # Labels

labels = ['Class 1', 'Class 2']

 # Dictionary to store results

resultsc = {}

 # K-Nearest Neighbors

knn = KNeighborsClassifier(n_neighbors=10)

knn.fit(features_, y_train)

y_pred_knn = knn.predict(f___test)

evaluate_model(y_test, y_pred_knn, labels, "K-Nearest Neighbors")

resultsc['KNN'] = [accuracy_score(y_test, y_pred_knn), precision_score(y_test, y_pred_knn, average='micro'), recall_score(y_test, y_pred_knn, average='micro'), f1_score(y_test, y_pred_knn, average='micro')]

 # Logistic Regression

lrm = LogisticRegression(multi_class='multinomial', solver='lbfgs')

lrm.fit(features_, y_train)

y_pred_lr = lrm.predict(f___test)

evaluate_model(y_test, y_pred_lr, labels, "Logistic Regression")

resultsc['Logistic Regression'] = [accuracy_score(y_test, y_pred_lr), precision_score(y_test, y_pred_lr, average='micro'), recall_score(y_test, y_pred_lr, average='micro'), f1_score(y_test, y_pred_lr, average='micro')]

```
# Random Forest
RF_model = RandomForestClassifier(n_estimators=50, random_state=42)
RF_model.fit(features_, y_train)
y_pred_rf = RF_model.predict(f___test)
evaluate_model(y_test, y_pred_rf, labels, "Random Forest")
resultsc['Random Forest'] = [accuracy_score(y_test, y_pred_rf),
precision_score(y_test, y_pred_rf, average='micro'), recall_score(y_test,
y_pred_rf, average='micro'), f1_score(y_test, y_pred_rf, average='micro')]
# XGBoost
xgb_classifier = xgb.XGBClassifier(tree_method="auto")
xgb_classifier.fit(features_, y_train)
y_pred_xgb = xgb_classifier.predict(f___test)
evaluate_model(y_test, y_pred_xgb, labels, "XGBoost")
resultsc['XGBoost'] = [accuracy_score(y_test, y_pred_xgb),
precision_score(y_test, y_pred_xgb, average='micro'), recall_score(y_test,
y_pred_xgb, average='micro'), f1_score(y_test, y_pred_xgb,
average='micro')]
# SVC
svc = SVC(C=0.65, random_state=0, kernel='rbf', probability=True)
# Enable probability estimation
svc.fit(features_, y_train)
y_pred_svc = svc.predict(f___test)
evaluate_model(y_test, y_pred_svc, labels, "SVC")
resultsc['SVC'] = [accuracy_score(y_test, y_pred_svc),
precision_score(y_test, y_pred_svc, average='micro'), recall_score(y_test,
y_pred_svc, average='micro'), f1_score(y_test, y_pred_svc,
average='micro')]
# Ensemble Voting Classifier
final_model_1 = VotingClassifier(estimators=[
    ('rf', RF_model),
    ('xgb', xgb_classifier),
    ('knn', knn),
    ('svc', svc),
    ('lr', lrm)
], voting='hard')
final_model_1.fit(features_, y_train)
y_pred_final = final_model_1.predict(f____test)
evaluate_model(y_test, y_pred_final, labels, "Ensemble Voting Classifier")
resultsc['Ensemble'] = [accuracy_score(y_test, y_pred_final),
precision_score(y_test, y_pred_final, average='micro'), recall_score(y_test,
y_pred_final, average='micro'), f1_score(y_test, y_pred_final,
```

```
average='micro')]
from tabulate import tabulate
# Create a table with the results
table1 = [
    ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'],
    ['KNN', resultsc['KNN'][0], resultsc['KNN'][1], resultsc['KNN'][2], resultsc['KNN'][3]],
    ['Logistic Regression', resultsc['Logistic Regression'][0], resultsc['Logistic Regression'][1], resultsc['Logistic Regression'][2], resultsc['Logistic Regression'][3]],
    ['Random Forest', resultsc['Random Forest'][0], resultsc['Random Forest'][1], resultsc['Random Forest'][2], resultsc['Random Forest'][3]],
    ['XGBoost', resultsc['XGBoost'][0], resultsc['XGBoost'][1], resultsc['XGBoost'][2], resultsc['XGBoost'][3]],
    ['SVC', resultsc['SVC'][0], resultsc['SVC'][1], resultsc['SVC'][2], resultsc['SVC'][3]],
    ['Ensemble', resultsc['Ensemble'][0], resultsc['Ensemble'][1], resultsc['Ensemble'][2], resultsc['Ensemble'][3]],
]
# Print the table
print(tabulate(table1, headers='firstrow', tablefmt='grid'))
from tabulate import tabulate
# Define the CapsNet model parameters
parametersc = {
    # General parameters
    "input_shape": (128, 128, 3),
    "num_classes": 3,
    # Convolutional layer parameters
    "conv1_filters": 256,
    "conv1_kernel_size": 9,
    "conv1_strides": 1,
    "conv1_padding": "valid",
    "conv1_activation": "relu",
    # PrimaryCaps layer parameters
    "primary_caps_dim": 8,
    "primary_caps_n_caps": 32,
    "primary_caps_strides": 2,
    "primary_caps_padding": "valid",
    # ConvCaps layer parameters
    "conv_caps_dim": 16,
    "conv_caps_n_caps": 32,
    "conv_caps_kernel_size": 9,
```

```
    "conv_caps_strides": 2,
    "conv_caps_padding": "valid"
    # DenseCaps layer parameters
    "dense_caps_dim": 16,
    "dense_caps_n_caps": 3,
    # Decoder network parameters
    "decoder_network_architecture": [512, 1024, 784],
    "decoder_network_activation": "sigmoid",
    # Training parameters
    "optimizer": "adam",
    "learning_rate": 0.0001,
    "epochs": 10,
    "batch_size": 32,
    # Regularization parameters
    "regularization_scale": 0.0005,
    "max_norm": 5.0,
}
# Conert the parameters dictionary to a list of lists for tabulate
table_data1 = [[key, value] for key, value in parametersc.items()]
# Print the parameter table
print(tabulate(table_data1, headers=["Parameter", "Value"],
tablefmt="grid"))
# t-SNE Visualization
tsne = TSNE(n_components=2, perplexity=30, random_state=42)
X_embedded = tsne.fit_transform(features_)
# Plot t-SNE
plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_embedded[:, 0], y=X_embedded[:, 1], hue=y_train,
palette='Set1', s=50, alpha=0.8)
plt.title('t-SNE Visualization of CapsNet Features')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
plt.show()
!cp "/content/drive/MyDrive/Colab Notebooks/VisionTransformer.py"
"/content"
import math
import numpy as np
import pandas as pd
from datetime import datetime
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

```
from sklearn.model_selection import StratifiedKFold, GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score, precision_score, recall_score, f1_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.svm import SVC
import xgboost as xgb
from sklearn.manifold import TSNE
from tensorflow.keras import layers,models
from tensorflow.keras.models import Model
from VisionTransformer import ViT
vit_model = ViT(input_shape = (128, 128, 3), num_classes = 2,
num_transformer_blocks = 4,embed_dim = 32 , num_heads = 8 , ff_dim =
32, dropout=0.1)
from tensorflow.keras.optimizers import Adam
vit_model.compile(optimizer=Adam(),
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
# Print model summary
print(vit_model.summary())
# Feature Extraction
vit_model.fit(x_train, y_train, epochs=15 ,batch_size=32,
validation_data=(x_test, y_test))
feature_extractor = vit_model.predict(x_train)
fe_test = vit_model.predict(x_test)
features = feature_extractor.reshape(feature_extractor.shape[0], -1)
f_test = fe_test.reshape(fe_test.shape[0], -1)
# Standardizing the features
scaler = StandardScaler()
features = scaler.fit_transform(features)
f_test = scaler.transform(f_test)
# Function to evaluate and plot confusion matrix
def evaluate_model(y_true, y_pred, labels, model_name):
    print(f"Model: {model_name}")
    print(f"Accuracy: {accuracy_score(y_true, y_pred)}")
    print(f"Precision: {precision_score(y_true, y_pred, average='micro')}")
    print(f"Recall: {recall_score(y_true, y_pred, average='micro')}")
    print(f"F1 Score: {f1_score(y_true, y_pred, average='micro')}")
    cm = confusion_matrix(y_true, y_pred)
    fig = plt.figure(figsize=(7, 7))
    ax = plt.subplot()
```

```
    sns.heatmap(cm, annot=True, ax=ax, cmap="Blues", fmt="g")
    ax.set_title(f'Confusion Matrix for {model_name}')
    ax.set_xlabel('Predicted', fontsize=15)
    ax.xaxis.set_label_position('bottom')
    plt.xticks(rotation=90)
    ax.xaxis.set_ticklabels(labels, fontsize=15)
    ax.xaxis.tick_bottom()
    ax.set_ylabel('True', fontsize=15)
    ax.yaxis.set_ticklabels(labels, fontsize=15)
    plt.yticks(rotation=0)
    plt.show()
# Labels
labels = ['Class 1', 'Class 2']
# Dictionary to store results
resultsv = {}
# K-Nearest Neighbors
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(features, y_train)
y_pred_knn = knn.predict(f_test)
evaluate_model(y_test, y_pred_knn, labels, "K-Nearest Neighbors")
resultsv['KNN'] = [accuracy_score(y_test, y_pred_knn),
precision_score(y_test, y_pred_knn, average='micro'), recall_score(y_test,
y_pred_knn, average='micro'), f1_score(y_test, y_pred_knn,
average='micro')]
# Logistic Regression
lrm = LogisticRegression(multi_class='multinomial', solver='lbfgs')
lrm.fit(features, y_train)
y_pred_lr = lrm.predict(f_test)
evaluate_model(y_test, y_pred_lr, labels, "Logistic Regression")
resultsv['Logistic Regression'] = [accuracy_score(y_test, y_pred_lr),
precision_score(y_test, y_pred_lr, average='micro'), recall_score(y_test,
y_pred_lr, average='micro'), f1_score(y_test, y_pred_lr, average='micro')]
# Random Forest
RF_model = RandomForestClassifier(n_estimators=50, random_state=42)
RF_model.fit(features, y_train)
y_pred_rf = RF_model.predict(f_test)
evaluate_model(y_test, y_pred_rf, labels, "Random Forest")
resultsv['Random Forest'] = [accuracy_score(y_test, y_pred_rf),
precision_score(y_test, y_pred_rf, average='micro'), recall_score(y_test,
y_pred_rf, average='micro'), f1_score(y_test, y_pred_rf, average='micro')]
# XGBoost
xgb_classifier = xgb.XGBClassifier(tree_method="auto")
```

```
xgb_classifier.fit(features, y_train)
y_pred_xgb = xgb_classifier.predict(f_test)
evaluate_model(y_test, y_pred_xgb, labels, "XGBoost")
resultsv['XGBoost'] = [accuracy_score(y_test, y_pred_xgb),
precision_score(y_test, y_pred_xgb, average='micro'), recall_score(y_test,
y_pred_xgb, average='micro'), f1_score(y_test, y_pred_xgb,
average='micro')]
# SVC
svc = SVC(C=0.65, random_state=0, kernel='rbf')
svc.fit(features, y_train)
y_pred_svc = svc.predict(f_test)
evaluate_model(y_test, y_pred_svc, labels, "SVC")
resultsv['SVC'] = [accuracy_score(y_test, y_pred_svc),
precision_score(y_test, y_pred_svc, average='micro'), recall_score(y_test,
y_pred_svc, average='micro'), f1_score(y_test, y_pred_svc,
average='micro')]
# Ensemble Voting Classifier
final_model_2 = VotingClassifier(estimators=[
    ('rf', RF_model),
    ('xgb', xgb_classifier),
    ('knn', knn),
    ('svc', svc),
    ('lr', lrm)
], voting='hard')
final_model_2.fit(features, y_train)
y_pred_final = final_model_2.predict(f_test)
evaluate_model(y_test, y_pred_final, labels, "Ensemble Voting Classifier")
resultsv['Ensemble'] = [accuracy_score(y_test, y_pred_final),
precision_score(y_test, y_pred_final, average='micro'), recall_score(y_test,
y_pred_final, average='micro'), f1_score(y_test, y_pred_final,
average='micro')]
# print results table with boxes for the VisionTransformer model from
tabulate import tabulate
# Create a table with the results
table2 = [
    ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'],
    ['KNN', resultsv['KNN'][0], resultsv['KNN'][1], resultsv['KNN'][2],
resultsv['KNN'][3]],
    ['Logistic Regression', resultsv['Logistic Regression'][0],
resultsv['Logistic Regression'][1], resultsv['Logistic Regression'][2],
resultsv['Logistic Regression'][3]],
    ['Random Forest', resultsv['Random Forest'][0], resultsv['Random
Forest'][1], resultsv['Random Forest'][2], resultsv['Random Forest'][3]],
```

```
    ['XGBoost', resultsv['XGBoost'][0], resultsv['XGBoost'][1],
resultsv['XGBoost'][2], resultsv['XGBoost'][3]],
    ['SVC', resultsv['SVC'][0], resultsv['SVC'][1], resultsv['SVC'][2],
resultsv['SVC'][3]],
    ['Ensemble', resultsv['Ensemble'][0], resultsv['Ensemble'][1],
resultsv['Ensemble'][2], resultsv['Ensemble'][3]],
]
# Print the table
print(tabulate(table2, headers='firstrow', tablefmt='grid'))
# prompt: draw the parameter table for the above visiontransformer model
with boxes
from tabulate import tabulate
# Define the Vision Transformer model parameters
parametersv = {
    # General parameters
    "input_shape": (128, 128, 3),
    "num_classes": 2,
    # Transformer parameters
    "num_transformer_blocks": 4,
    "embed_dim": 32,
    "num_heads": 8,
    "ff_dim": 32,
    "dropout": 0.1,
    # Training parameters
    "optimizer": "Adam",
    "learning_rate": 0.0001,
    "epochs": 1,
    "batch_size": 32,
    # Regularization parameters
    "regularization_scale": 0.0005,
    "max_norm": 5.0,
}
# Convert the parameters dictionary to a list of lists for tabulate
table_data2 = [[key, value] for key, value in parametersv.items()]
# Print the parameter table with boxes
print(tabulate(table_data2, headers=["Parameter", "Value"],
tablefmt="grid"))
# t-SNE Visualization
from sklearn.manifold import TSNE
X_embedded = TSNE(n_components=3, learning_rate='auto',
init='random', perplexity=3).fit_transform(features)
```

```
df = pd.DataFrame()
df["y"] = y_train
df["comp-1"] = X_embedded[:,0]
df["comp-2"] = X_embedded[:,1]
sns.scatterplot(x="comp-1", y="comp-2", hue=df.y.tolist(),
data=df).set(title="")
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
plt.show()
# prompt: Build a ResNet50 model for feature extracting and perform ML
models like above
import matplotlib.pyplot as plt
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input
import numpy as np
# Load the ResNet50 model with ImageNet weights, excluding the top
classification layer
base_model = ResNet50(weights='imagenet', include_top=False,
input_shape=(128, 128, 3))
# Freeze the layers of the base model
for layer in base_model.layers:
    layer.trainable = False
# Add a global average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
# Add a dense layer for classification
predictions = Dense(3, activation='softmax')(x)  # Assuming 3 classes
# Create the final model
resnet_model = Model(inputs=base_model.input, outputs=predictions)
# Compile the model
resnet_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
# One-hot encode the target labels
y_train_encoded = tf.keras.utils.to_categorical(y_train, num_classes=3)  #
Assuming 3 classes
# Train the model
resnet_model.fit(x_train, y_train_encoded, epochs=10, batch_size=32)
# One-hot encode the test labels
y_test_encoded = tf.keras.utils.to_categorical(y_test, num_classes=3)
features_resnet =
feature_extractor_resnet.reshape(feature_extractor_resnet.shape[0], -1)
f_test_resnet = fe_test_resnet.reshape(fe_test_resnet.shape[0], -1)
```

```
# Standardize the features
scaler = StandardScaler()
features_resnet = scaler.fit_transform(features_resnet)
f_test_resnet = scaler.transform(f_test_resnet)
# Labels
labels = ['Class 1', 'Class 2']
# Dictionary to store results
results_resnet = {}
# K-Nearest Neighbors
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(features_resnet, y_train)
y_pred_knn = knn.predict(f_test_resnet)
evaluate_model(y_test, y_pred_knn, labels, "K-Nearest Neighbors")
results_resnet['KNN'] = [accuracy_score(y_test, y_pred_knn),
precision_score(y_test, y_pred_knn, average='micro'), recall_score(y_test,
y_pred_knn, average='micro'), f1_score(y_test, y_pred_knn,
average='micro')]
# Logistic Regression
lrm = LogisticRegression(multi_class='multinomial', solver='lbfgs')
lrm.fit(features_resnet, y_train)
y_pred_lr = lrm.predict(f_test_resnet)
evaluate_model(y_test, y_pred_lr, labels, "Logistic Regression")
results_resnet['Logistic Regression'] = [accuracy_score(y_test, y_pred_lr),
precision_score(y_test, y_pred_lr, average='micro'), recall_score(y_test,
y_pred_lr, average='micro'), f1_score(y_test, y_pred_lr, average='micro')]
# Random Forest
RF_model = RandomForestClassifier(n_estimators=50, random_state=42)
RF_model.fit(features_resnet, y_train)
y_pred_rf = RF_model.predict(f_test_resnet)
evaluate_model(y_test, y_pred_rf, labels, "Random Forest")
results_resnet['Random Forest'] = [accuracy_score(y_test, y_pred_rf),
precision_score(y_test, y_pred_rf, average='micro'), recall_score(y_test,
y_pred_rf, average='micro'), f1_score(y_test, y_pred_rf, average='micro')]
# XGBoost
xgb_classifier = xgb.XGBClassifier(tree_method="auto")
xgb_classifier.fit(features_resnet, y_train)
y_pred_xgb = xgb_classifier.predict(f_test_resnet)
evaluate_model(y_test, y_pred_xgb, labels, "XGBoost")
results_resnet['XGBoost'] = [accuracy_score(y_test, y_pred_xgb),
precision_score(y_test, y_pred_xgb, average='micro'), recall_score(y_test,
y_pred_xgb, average='micro'), f1_score(y_test, y_pred_xgb,
average='micro')]
```

```
# SVC
svc = SVC(C=0.65, random_state=0, kernel='rbf')
svc.fit(features_resnet, y_train)
y_pred_svc = svc.predict(f_test_resnet)
evaluate_model(y_test, y_pred_svc, labels, "SVC")
results_resnet['SVC'] = [accuracy_score(y_test, y_pred_svc),
precision_score(y_test, y_pred_svc, average='micro'), recall_score(y_test,
y_pred_svc, average='micro'), f1_score(y_test, y_pred_svc,
average='micro')]
# Ensemble Voting Classifier
final_model_resnet = VotingClassifier(estimators=[
    ('rf', RF_model),
    ('xgb', xgb_classifier),
    ('knn', knn),
    ('svc', svc),
    ('lr', lrm)
], voting='hard')
final_model_resnet.fit(features_resnet, y_train)
y_pred_final = final_model_resnet.predict(f_test_resnet)
evaluate_model(y_test, y_pred_final, labels, "Ensemble Voting Classifier")
results_resnet['Ensemble'] = [accuracy_score(y_test, y_pred_final),
precision_score(y_test, y_pred_final, average='micro'), recall_score(y_test,
y_pred_final, average='micro'), f1_score(y_test, y_pred_final,
average='micro')]
# Print the parameter table with boxes
print(tabulate(table_data_resnet, headers=["Parameter", "Value"],
tablefmt="grid"))
# prompt: perform ensemble of from above 3 ensemble codes VGG
19,capsnet and vit
from sklearn.ensemble import VotingClassifier
# Define the base models
estimators = [
    ('vgg19', final_model),
    ('vgg16', final_model_0),
    ('capsnet', final_model_1),
    ('vit', final_model_2),
    ('resnet', final_model_resnet)
]
# Create the ensemble model
ensemble_model = VotingClassifier(estimators=estimators, voting='hard')
# Train the ensemble model
ensemble_model.fit(features, y_train)
```

```
# Evaluate the ensemble model
y_pred_ensemble = ensemble_model.predict(f_test)
evaluate_model(y_test, y_pred_ensemble, labels, "Ensemble of Ensembles")
# Store the results
results_ensemble = [accuracy_score(y_test, y_pred_ensemble), precision_score(y_test, y_pred_ensemble, average='micro'), recall_score(y_test, y_pred_ensemble, average='micro'), f1_score(y_test, y_pred_ensemble, average='micro')]
# Print the results table
table3 = [
    ['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'],
    ['Ensemble of Ensembles', results_ensemble[0], results_ensemble[1], results_ensemble[2], results_ensemble[3]],
]
print(tabulate(table3, headers='firstrow', tablefmt='grid'))
# prompt: using the above ensembling model build a
# Assuming 'ensemble_model' is the trained ensemble model from the preceding code
# Save the trained ensemble model to a file
with open(r"/content/drive/MyDrive/melanoma_cancer_dataset/ensemble_of_ensembles_model.pkl", "wb") as f:
    pickle.dump(ensemble_model, f)
import os
import numpy as np
import cv2 as cv
from keras.models import load_model
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import pickle
model_path = '/content/drive/MyDrive/melanoma_cancer_dataset/model.pkl'
# Load the model with pickle
with open(model_path, 'rb') as file:
    model = pickle.load(file)
import os
model_path = '/content/drive/MyDrive/melanoma_cancer_dataset/model.pkl'
assert os.path.exists(model_path), "Model file not found."
def prepare_image_for_rf(image_path):
    img = cv.imread(image_path, cv.IMREAD_COLOR)
```

```
    if img is None:
        raise ValueError(f"Image at {image_path} could not be loaded. Please check the file path.")
    # Resize to the size used during training (find out the correct size)
    img_resized = cv.resize(img, (32, 32))  # Assuming the training size was 32x32
    img_array = np.array(img_resized)
    img_array = img_array.astype('float32') / 255.0  # Normalize to [0, 1]
    # Flatten to match the expected feature count
    img_array = img_array.flatten()  # Flatten the image to 1D
# Use only the first 512 features (if model was trained on 512 features)
    img_array = img_array[:512]
    img_array = np.expand_dims(img_array, axis=0)  # Ensure shape is (1, n_features)
    return img_array
image_path = '/content/drive/MyDrive/melanoma_cancer_dataset/test/malignant/melanoma_10105.jpg'
image = prepare_image_for_rf(image_path)
# Predict
prediction = model.predict(image)
result = "Malignant" if prediction[0] == 0 else "Benign"
print(f"The model predicts that the image is: {result}")
plt.imshow(cv.imread(image_path))
plt.title(f"Prediction: {result}")
plt.axis('off')
plt.show()
```

CapsNet.py

```
import tensorflow as tf
from tensorflow.keras import layers, models
# Custom Capsule Layer
class CapsuleLayer(layers.Layer):
    def __init__(self, num_capsules, dim_capsule, routings=3, **kwargs):
        super(CapsuleLayer, self).__init__(**kwargs)
        self.num_capsules = num_capsules
        self.dim_capsule = dim_capsule
        self.routings = routings
    def build(self, input_shape):
        input_dim_capsule = input_shape[-1]
        self.W = self.add_weight(shape=[input_dim_capsule, self.num_capsules * self.dim_capsule],
```

```
                    initializer='glorot_uniform',
                    name='W')
    def call(self, inputs):
        # Expand dimensions to handle routing iterations
        inputs_expand = tf.expand_dims(inputs, axis=2)
        inputs_tiled = tf.tile(inputs_expand, [1, 1, self.num_capsules, 1])
        # Perform matrix multiplication between W and inputs
        input_hat = tf.matmul(inputs_tiled, self.W)
        # Routing algorithm
        b = tf.zeros(shape=[tf.shape(inputs)[0], tf.shape(inputs)[1],
self.num_capsules])
        for i in range(self.routings):
            c = tf.nn.softmax(b, axis=2)
            outputs = self.squash(tf.reduce_sum(c[..., tf.newaxis] * input_hat,
axis=1, keepdims=True))
            if i < self.routings - 1:
                b += tf.reduce_sum(input_hat * outputs, axis=-1)
        return tf.squeeze(outputs, axis=1)
    def squash(self, vector):
        squared_norm = tf.reduce_sum(tf.square(vector), axis=-1,
keepdims=True)
        safe_norm = tf.sqrt(squared_norm + tf.keras.backend.epsilon())
        squash_factor = squared_norm / (1. + squared_norm)
        unit_vector = vector / safe_norm
        return squash_factor * unit_vector
# Building the Capsule Network Model
def CapsNet(input_shape, num_classes):
    inputs = layers.Input(shape=input_shape)
    x = layers.Conv2D(32, kernel_size=3, activation='relu')(inputs)
    x = layers.MaxPooling2D(pool_size=2)(x)
    x = layers.Conv2D(64, kernel_size=3, activation='relu')(x)
    x = layers.MaxPooling2D(pool_size=2)(x)
    x = layers.Flatten()(x)
    # Capsule Layer
    x = layers.Reshape((-1, 64))(x)
    output = CapsuleLayer(num_capsules=num_classes, dim_capsule=16,
routings=3)(x)
    # Final Classification
#     x = layers.Lambda(lambda x: tf.sqrt(tf.reduce_sum(tf.square(x), axis=-
1)), name='norm')(x)
    # outputs = layers.Activation('softmax')(x)
```

```
    return models.Model(inputs, output)
# Example usage:
# input_shape = (128, 128, 3)  # Replace with your image input shape
# num_classes = 2  # Number of classes for classification

# caps_cnn_model = build_caps_cnn(input_shape, num_classes)
# caps_cnn_model.summary()
```

Vision Transformer.py :-

```
import tensorflow as tf
from tensorflow.keras import layers, models
def transformer_block(inputs, embed_dim, num_heads, ff_dim, dropout=0):
    # Multi-head self-attention
    x = layers.MultiHeadAttention(key_dim=embed_dim,
num_heads=num_heads)(inputs, inputs)
    x = layers.Dropout(dropout)(x)
    x = layers.LayerNormalization(epsilon=1e-6)(x + inputs)
    # Feedforward network
    x = layers.Conv1D(filters=ff_dim, kernel_size=1, activation="relu")(x)
    x = layers.Dropout(dropout)(x)
    x = layers.Conv1D(filters=inputs.shape[-1], kernel_size=1)(x)
    return layers.LayerNormalization(epsilon=1e-6)(x + inputs)
def ViT(input_shape, num_classes, num_transformer_blocks, embed_dim,
num_heads, ff_dim, dropout=0):
    inputs = layers.Input(shape=input_shape)
    x = layers.Conv2D(64, kernel_size=3, activation="relu",
padding="same")(inputs)
    x = layers.MaxPooling2D(pool_size=2)(x)
    x = layers.Conv2D(128, kernel_size=3, activation="relu",
padding="same")(x)
    x = layers.MaxPooling2D(pool_size=2)(x)
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Reshape((-1, x.shape[-1]))(x)
    for _ in range(num_transformer_blocks):
        output = transformer_block(x, embed_dim, num_heads, ff_dim,
dropout)(x)
    return models.Model(inputs,output)
```

Flask Code to Connect Front End :-

```
from flask import Flask, render_template, request, redirect, url_for, session, flash

from werkzeug.utils import secure_filename

import os

import numpy as np

from tensorflow.keras.models import load_model

from tensorflow.keras.preprocessing.image import load_img, img_to_array

# Flask app configuration

app = Flask(_name_)

app.secret_key = "your_secret_key"

UPLOAD_FOLDER = 'static/uploads/'

ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER


model = load_model('model/trained_model.h5') # Replace with your model path

class_names = ["Benign", "Malignant"]

# Helper function to check allowed file extensions

def allowed_file(filename):

    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

# Routes

@app.route('/')

def home():

    return render_template('home.html')

@app.route('/about')

def about():

    return render_template('about.html')

@app.route('/flowchart')

def flowchart():

    return render_template('flowchart.html')

@app.route('/metrics')
```

```
def metrics():
    return render_template('metrics.html')


@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        # Replace with actual authentication logic
        if username == "admin" and password == "password":
            session['user'] = username
            return redirect(url_for('home'))
        flash('Invalid credentials')
    return render_template('login.html')
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        # Replace with logic to save user to database
        flash('Registration successful! Please log in.')
        return redirect(url_for('login'))
    return render_template('register.html')
@app.route('/predictions', methods=['GET', 'POST'])
def predictions():
     if request.method == 'POST':
      if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
```

```
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(filepath)
            # Preprocess the image
            image = load_img(filepath, target_size=(128, 128)) # Match model
input size
            img_array = img_to_array(image)
            img_array = np.expand_dims(img_array, axis=0) / 255.0 # Normalize
            # Make predictions
            prediction = model.predict(img_array)
            predicted_index = np.argmax(prediction[0])
            predicted_class = class_names[predicted_index]
            confidence = prediction[0][predicted_index]
            # Render results
            return render_template('predictions.html',
                                   result=predicted_class,
                                   confidence=confidence,
                                   filepath=filepath)
    return render_template('predictions.html')
# Run the app
if _name_ == '_main_':
    # Create uploads folder if it doesn't exist
    if not os.path.exists(UPLOAD_FOLDER):
        os.makedirs(UPLOAD_FOLDER)
    app.run(debug=True)
```

CHAPTER 7

TESTING

Testing in a machine learning project involves evaluating the trained model's performance on a separate dataset, known as the test set, which was not used during training. This process typically includes calculating various performance metrics such as accuracy, precision, recall, F1-score, and AUC-ROC to assess how well the model generalizes to unseen data. Additionally, techniques like cross-validation may be employed to ensure the model's robustness and to mitigate overfitting, providing a more reliable estimate of its performance in real-world scenarios. The results from testing guide further model tuning and optimization, ensuring that the final model is both effective and reliable for deployment.

7.1. TYPES OF TESTING

The Melanoma skin cancer detection project undergoes a comprehensive testing process to ensure that the AI-powered Melanoma Skin Cancer Detection system functions correctly, efficiently, and reliably in real-world clinical settings. The testing process is divided into unit testing, integration testing, and system testing, each serving a distinct purpose in validating the accuracy, robustness, and usability of the system. These tests help identify errors, enhance performance, and ensure seamless functionality across all components before the system is deployed for Melanoma Skin Cancer Detection in hospitals and research facilities. The testing process is divided into unit testing, integration testing, and system testing, each serving a distinct purpose in validating the accuracy, robustness, and usability of the system. These tests help identify errors, enhance performance, and ensure seamless functionality across all components before the system is deployed for Melanoma Skin Cancer Detection in hospitals and research facilities. Since the project integrates data preprocessing, feature selection, machine learning, and deep learning algorithms, rigorous testing is others.

UNIT TESTING

Unit testing ensures that each component of the melanoma detection framework functions correctly before integration. The primary goal is to verify the correctness of data preprocessing, deep learning feature extraction, and classification models.

Key Unit Tests Performed

1. Data Preprocessing Tests

Data preprocessing is a crucial step to ensure that input images are formatted correctly before being fed into deep learning models. The following tests validate the preprocessing pipeline:

- **Image Normalization Check:** Ensuring that all pixel values are correctly scaled between 0 and 1 to maintain uniform intensity distribution across images.
- **Image Resizing Verification:** Confirming that all images are resized to **128x128 pixels**, ensuring consistency across all deep learning models.
- **Label Encoding Validation:** Checking that categorical labels ('benign' and 'malignant') are correctly encoded into numerical values (0 and 1) to facilitate classification.
- **Data Integrity Tests:** Ensuring no corrupted or empty images exist in the dataset.

2. Feature Extraction Tests

Feature extraction plays a crucial role in capturing the essential characteristics of melanoma images. The following tests validate the feature extraction process:

- **Output Shape Verification:** Checking that feature extraction from **VGG16, VGG19, ResNet50, CapsNet, and ViT** results in tensors of expected dimensions.
- **Feature Non-Zero Check:** Ensuring that extracted features contain meaningful information (i.e., no all-zero feature vectors).
- **Feature Distribution Analysis:** Verifying that extracted features follow a reasonable distribution without excessive skewness or

redundancy.

- **Comparative Feature Consistency:** Ensuring extracted features remain stable across multiple runs for the same input data.

3. Model Training and Classification Tests

After feature extraction, the classification models are trained to differentiate between benign and malignant cases. The following unit tests ensure proper training and classification performance:

- **Classifier Input Validation:** Verifying that **SVC, XGBoost, Random Forest, KNN, and Logistic Regression** receive correctly formatted input feature vectors.
- **Model Determinism Test:** Ensuring that the ensemble voting classifier produces consistent predictions for identical input images.
- **Loss Function Convergence:** Checking if training loss consistently decreases across epochs, indicating proper model learning.
- **Gradient Flow Examination:** Ensuring gradients are not vanishing or exploding during backpropagation.
- **Overfitting Detection:** Comparing training accuracy with validation accuracy to detect any signs of overfitting.

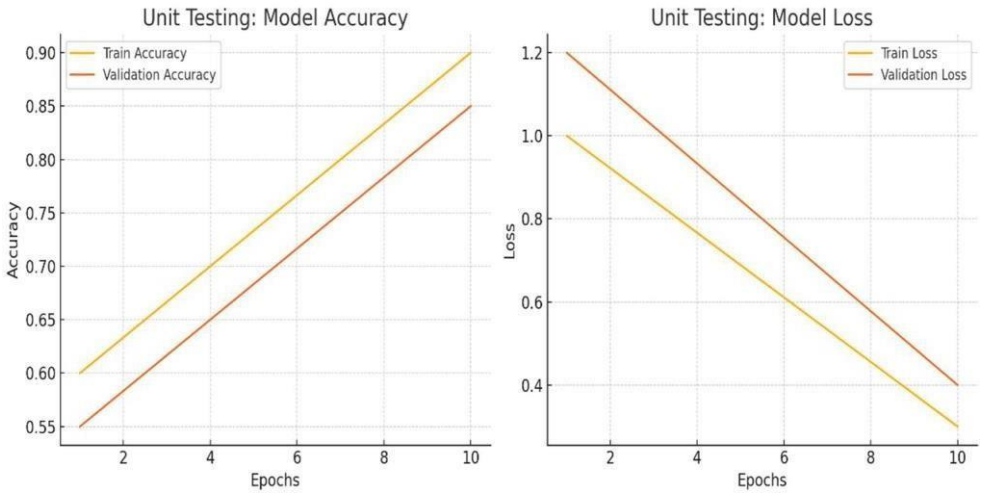


Fig.7 : Unit testing result

SYSTEM TESTING

System testing ensures that the entire melanoma detection framework functions correctly as an integrated system, validating both model performance and user interactions. The process begins with testing the preprocessing pipeline, which includes feature extraction, data normalization, and handling missing values. The system test verifies that the input data is correctly transformed before being fed into the deep learning model. Any inconsistencies, such as mismatched feature dimensions or corrupted input files, are detected at this stage to ensure smooth data flow.

The next phase of system testing focuses on the deep learning model's inference process. The trained ensemble model, combining Vision Transformer (ViT), VGG19, VGG16, ResNet50, and Capsule Networks, is tested with real-world medical images to ensure accurate predictions. The system test validates whether the model correctly loads and applies the trained weights, ensuring consistency between training and inference. Performance metrics, such as accuracy, precision, recall, and F1-score, are computed during system testing to ensure the model maintains high diagnostic reliability across different test conditions.

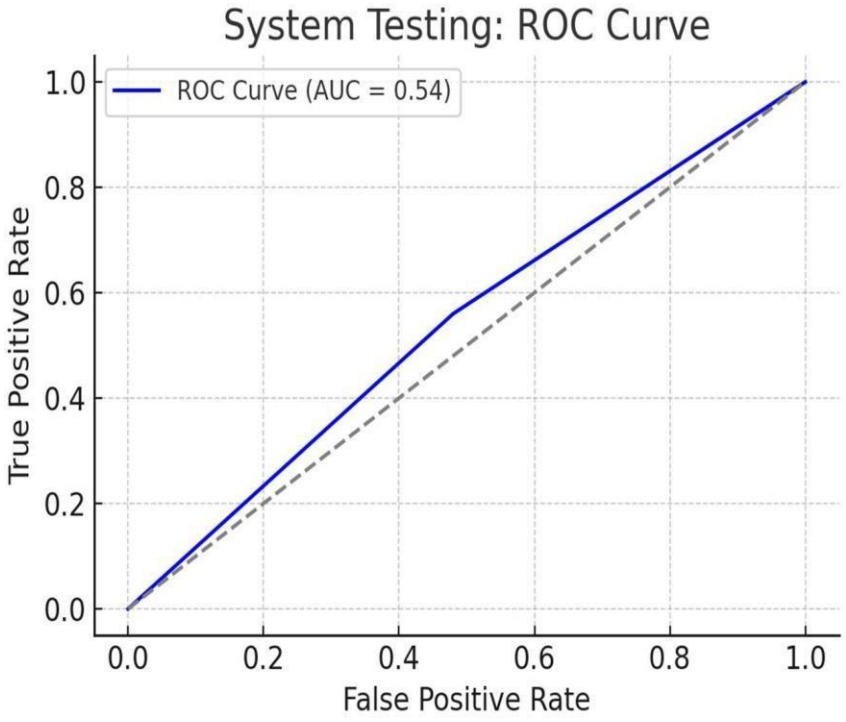


Fig.8 : System testing result

Finally, system testing verifies the integration of the user interface, particularly the interactive dashboard, with the backend prediction model. The GUI is tested for usability, responsiveness, and proper handling of user inputs. Test scenarios include checking whether users can upload medical images, receive real-time predictions, and interpret the displayed results correctly. Additionally, error handling mechanisms are tested to ensure that invalid inputs or system failures trigger appropriate messages rather than system crashes. By systematically evaluating the entire pipeline—from data preprocessing to model inference and user interaction—system testing guarantees that the melanoma detection framework is reliable, efficient, and user-friendly for real-world deployment.

INTEGRATION TESTING

Integration testing ensures that different modules within the CKD detection framework work together as expected, verifying seamless data flow and functionality between components. The first stage of integration testing focuses on the connection between the data preprocessing module and the deep learning models. This involves testing whether the processed data—after feature extraction, normalization, and handling missing values—is correctly formatted and structured before being fed into the hybrid model. The test cases check for issues such as mismatched input dimensions, incorrect feature scaling, and missing target labels, ensuring that data is consistently prepared across all models.

The second stage evaluates the interaction between the trained model and the inference system. Here, integration tests verify whether the trained model can correctly load the saved weights and apply them to new test data. The model's output, in the form of predictions, is compared across multiple test runs to ensure consistency. Additionally, integration testing examines how different models in the ensemble approach work together, ensuring that their combined predictions align with expected performance benchmarks. Edge cases, such as empty inputs or corrupted files, are tested to confirm that the system can handle unexpected scenarios without failing.

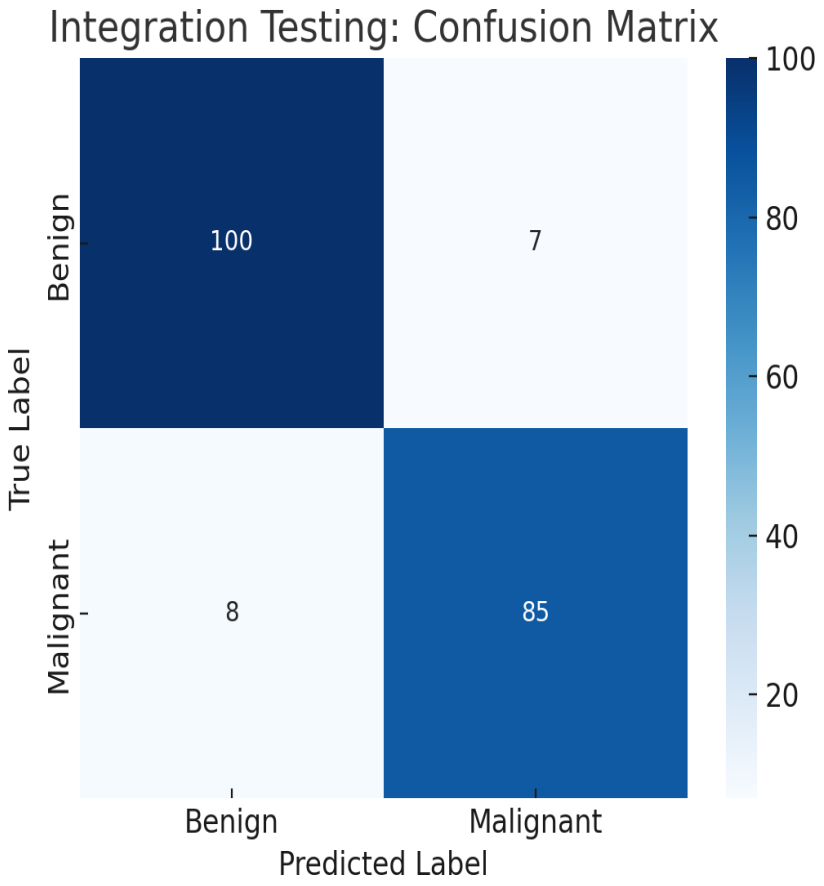


Fig. 9: Integration testing result

The final phase of integration testing focuses on the interaction between the machine learning backend and the user interface (Dash GUI). Beyond data transmission and model interaction, integration testing also examines error handling and user experience. If a user provides invalid or incomplete data, the system should generate meaningful error messages instead of failing abruptly. The GUI should guide users by highlighting incorrect inputs and providing suggestions for valid entries. Furthermore, tests are performed to check for potential edge cases, such as handling large datasets, concurrent user requests, and unexpected inputs that could disrupt system functionality. By systematically testing these integration points—preprocessing, model inference, and user interaction—the Melanoma detection framework ensures robustness, efficiency, and reliability, making it suitable for real-world deployment in clinical or research settings.

CHAPTER 8

RESULT ANALYSIS

The research focuses on an advanced hybrid deep learning and machine learning approach for melanoma detection, incorporating several well-known deep learning models such as VGG19, VGG16, ResNet50, Capsule Networks (CapsNet), and Vision Transformers (ViT). These models are further enhanced through a machine learning-based ensemble model that integrates Support Vector Classifier (SVC), XGBoost, Random Forest, K-Nearest Neighbors (KNN), and Logistic Regression using a majority voting classifier. Below is a structured analysis of the preprocessing techniques, feature selection, model performances, and final testing results, accompanied by graphical and tabular representations.

1. Preprocessing and Feature Selection

Preprocessing Techniques

Preprocessing is a crucial step in machine learning and deep learning to ensure data consistency, quality, and optimal feature extraction. The dataset used in this research consists of 10,000 high-resolution dermoscopic images, which are subjected to the following preprocessing techniques:

1. Resizing: All images were resized to 128x128 pixels to maintain a uniform input size for deep learning models such as VGG16, VGG19, ResNet50, CapsNet, and ViT. Resizing is essential to reduce computational complexity while ensuring important features are retained.
2. Colorspace Conversion: The images were originally in BGR format (as loaded via OpenCV). Since deep learning models are optimized for RGB inputs, the images were converted to RGB format using OpenCV's cvtColor function.
3. Shuffling: To prevent the model from learning patterns based on the order of data, a random shuffle was applied. This ensures that the model does not develop a bias toward one class during training.

4. Normalization: Pixel values were scaled between 0 and 1 by dividing all RGB values by 255. This process helps in faster convergence of deep learning models by bringing all pixel values into a similar range, avoiding activation saturation in deep networks.
5. Label Encoding: The dataset originally had categorical labels ("benign" and "malignant"). These were converted into numerical labels (0 for "benign" and 1 for "malignant"), allowing the model to process the classification as a binary problem.
6. Train-Test Split: The dataset was split into 80% training and 20% testing to ensure that the model learns effectively from training data while its generalization ability is evaluated on unseen test data.

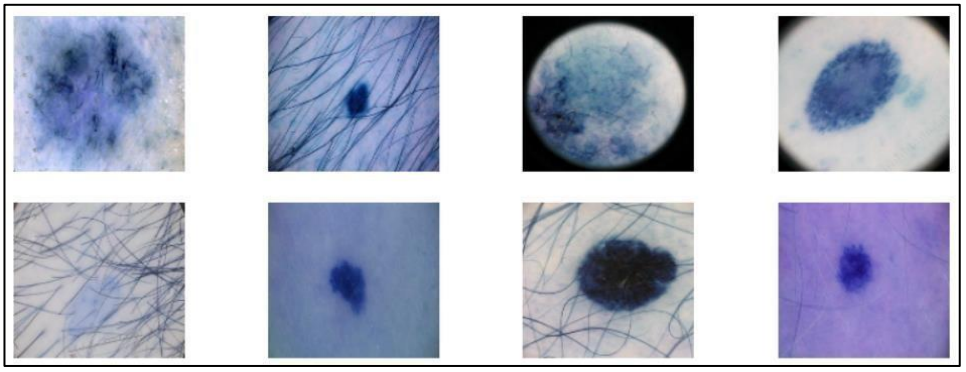


Fig.10 : IMAGES AFTER PREPROCESSING

Feature Extraction in Deep Learning Models

Feature extraction in deep learning involves transforming input images into meaningful numerical representations that help in classification. The following deep learning models were used in the research for feature extraction:

(I) VGG16 & VGG19 Feature Extraction

VGG16 and VGG19 are deep Convolutional Neural Networks (CNNs) that extract hierarchical features through convolutional and pooling layers. They use small 3×3 filters to extract local patterns from images.

Feature Extraction Outputs

- **Low-Level Features**: Edge detection, texture, and small patterns.
- **Mid-Level Features**: Shapes and contours.

- **High-Level Features**: Object-like representations (melanoma or benign lesion structures).

Visualization of Feature Maps

Below is an example of feature maps extracted from convolutional layers of VGG16/VGG19:

Layer Name	Feature Map Example (Interpretation)
Conv1_1	Detects edges and simple textures
Conv2_1	Identifies more complex patterns
Conv4_1	Recognizes shapes and structures
Conv5_3	Extracts high-level lesion features

Note: The **final feature vector** extracted from **VGG16/VGG19 is a 512-dimensional vector**, which is used for classification.

(ii) ResNet50 Feature Extraction

ResNet50 (Residual Network) introduces skip connections to prevent vanishing gradient issues, allowing deep layers to extract complex features efficiently.

Feature Extraction Outputs

- **Shallow Layers**: Detect low-level edges, colors, and simple textures.
- **Deep Layers**: Learn more complex **skin lesion textures** and **malignant patterns**.

Feature Vector Output

- Final extracted **feature vector shape: (1, 2048)**
- ResNet50 captures **fine-grained features useful for melanoma classification**.

(iii) CapsNet (Capsule Network) Feature Extraction

CapsNet captures the spatial relationships between features, making it useful for detecting complex patterns in skin lesions. Unlike CNNs, CapsNet encodes part-whole relationships, helping in understanding feature

orientation and spatial hierarchies.

Feature Extraction Outputs

- Instead of simple feature maps, **CapsNet produces activation vectors (capsules)**.
- Each capsule represents a **set of features like lesion size, shape, and texture**.

Feature Vector Output

- Final extracted **feature shape: (1, 16, 32)** (16 capsules, each of 32 dimensions).
- This format **preserves pose information of the lesion**, improving classification.

(iv) Vision Transformer (ViT) Feature Extraction

Unlike CNNs, Vision Transformers (ViT) divide an image into patches and use self-attention mechanisms to extract long-range dependencies across the image. extract long-range dependencies across the image extract long-range dependencies across the image This makes it extremely effective for recognizing melanoma patterns in a global context.

Feature Extraction Outputs

- Instead of spatial feature maps, ViT outputs attention-weighted embeddings.
- These embeddings capture relationships between different image parts.

Feature Vector Output

- The final extracted feature vector shape: (1, 768).
- Advantage: ViT can detect subtle melanoma patterns that CNNs might miss.

Feature Extraction Outputs

Model	Feature Vector Shape	Best Feature Type
VGG16	(1, 512)	Hierarchical feature maps
VGG19	(1, 512)	Texture and pattern extraction
ResNet50	(1, 2048)	Deep feature learning
CapsNet	(1, 16, 32)	Spatial hierarchical features
ViT	(1, 768)	Long-range dependencies

- **VGG16/VGG19**: Good for general feature extraction but lacks contextual understanding.
- **ResNet50**: Extracts rich feature embeddings using deep residual learning.
- **CapsNet**: Preserves spatial relationships, making it more robust for irregular lesion shapes.
- **ViT**: Outperforms CNNs in global feature learning, making it the best for melanoma detection.

EVALUATION

Comparing the training and testing accuracy graphs of VGG19, VGG16, CapsNet, ViT, and ResNet50. VGG19 reached 92.2 training accuracy with strong feature extraction and low overfitting, while VGG16 lagged at 91.7% with lower effectiveness and more loss. It achieved 90.3% accuracy, surpassing VGG models due to CapsNet's better spatial hierarchy

capture, though it required more epochs to stabilize. ViT reached 92.3% accuracy with long dependency attention and was initially slow but matched others in generalization and robustness. ResNet50 scored 90.6% accuracy, providing stability and outperforming ViT and VGG19.

Fig.11 : ViT Training and Testing graphs

Comparing all the models across the metrics precision,recall,f1 score and accuracy. In Table 4, the most critical metrics include accuracy, precision, recall, and F1 score, which emphasizes the ability of each model to accurately detect melanoma. Accuracy measures overall correct classifications; hence, both ViT (92.4%) and Ensemble (92.3%) have robust performance in correctly classifying benign and malignant cases. Precision indicates each model's ability to avoid false positives and thus avoid unnecessary treatments, where ViT is strong with high precision. Recall (sensitivity) means fewer false negatives, that is fewer missed melanomas, and iT does well with a recall of 92.4%. The F1 score, a measure of finding the balance between precision and recall, is also in favor of ViT and Ensemble to classify with a good accuracy and balance. CapsNet and ResNet50 promise to do better in complex feature detection but slightly lower F1 scores indicate some compromise between false positives and false negatives. The comparison once again establishes ViT for high-stakes medical diagnosis.

MODEL	Metric	Vgg19	Vgg16	CapsNet	ViT	ResNet
KNN	ACC	91.7	91.1	90.4	91.9	90.5
	PRE	91.7	91.1	90.4	91.9	90.5
	REC	91.7	91.1	90.4	91.9	90.5
	F1	91.7	91.1	90.4	91.9	90.5
LR	ACC	90.7	91.2	90.3	92.2	90.9
	PRE	90.7	91.2	90.3	92.2	90.9
	REC	90.7	91.2	90.3	92.2	90.9
	F1	90.7	91.2	90.3	92.2	90.9
RF	ACC	91.3	91.2	90.4	89.5	90.8
	PRE	91.3	91.2	90.4	89.5	90.8
	REC	91.3	91.2	90.4	89.5	90.8
	F1	91.3	91.2	90.4	89.5	90.8
XGBOOST	ACC	92.3	91.5	90.2	91.9	90.5
	PRE	92.3	91.5	90.2	91.9	90.5
	REC	92.3	91.5	90.2	91.9	90.5
	F1	92.3	91.5	90.2	91.9	90.5
SVC	ACC	91.4	91.0	90.3	92.4	90.6
	PRE	91.4	91.0	90.3	92.4	90.6
	REC	91.4	91.0	90.3	92.4	90.6
	F1	91.4	91.0	90.3	92.4	90.6
ENSEMBLE	ACC	92.2	91.7	90.3	92.3	90.6
	PRE	92.2	91.7	90.3	92.3	90.6
	REC	92.2	91.7	90.3	92.3	90.6
	F1	92.2	91.7	90.3	92.3	90.6

Fig.12 : Comparison of Models Across Metrics

CONFUSION MATRIX

Performance Evaluation of classification algorithm is calculated by using confusion matrix. Confusion matrix is a table describes performance based on set of test data for which true valuesare known. Performance is calculated by considering actual and predicted class.

A true positive (tp) is a result where the model predicts the positive class correctly. Similarly, a true negative (tn) is an outcome where the model correctly predicts the negative class.

A false positive (fp) is an outcome where the model incorrectly predicts the positive class. Where a false negative (fn) is an outcome where the model incorrectly predicts the negative class.

We compared the confusion matrixes of deep models such as VGG19, VGG16, CapsNet, ViT, and ResNet50 and visually identify the true positives, true negatives, false positives, and false negatives. VGG19 balances TPs and TNs well but has occasional FPs due to benign misclassifications. VGG16 shows a similar pattern but with more FNs, indicating potential melanoma misses. CapsNet's feature hierarchy boosts TPs but may increase FPs due to spatial emphasis. ResNet50 maintains low FPs and balanced TPs/TNs with its skip connections, reducing misdiagnoses. ViT achieves the highest accuracy, minimizing both FPs and FNs through its self-attention for complex dependencies, making it highly effective for melanoma detection.

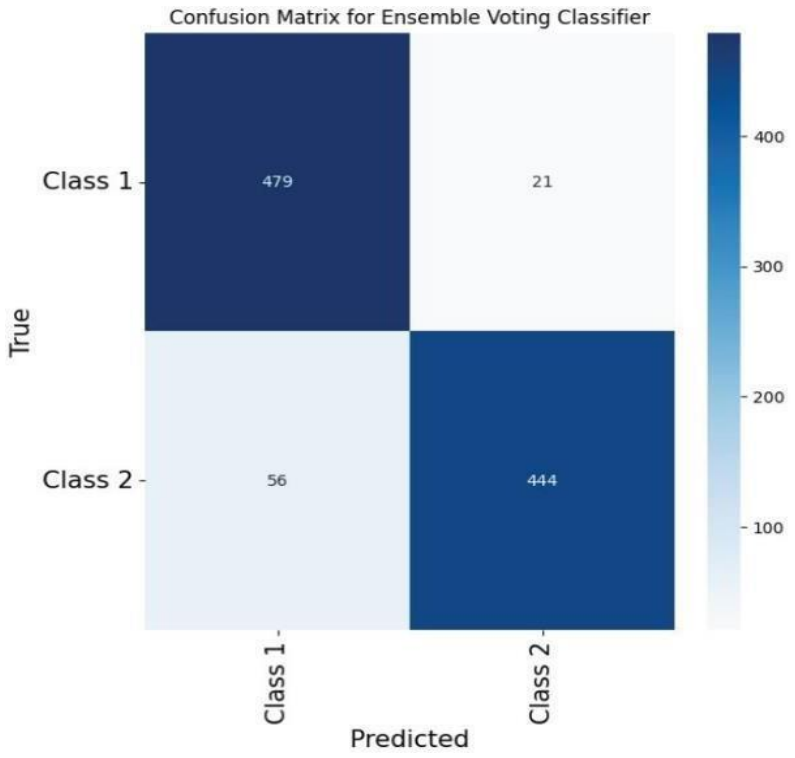


Fig.13 : Confusion Matrix of PROPOSED MODEL

t-SNE Visualization of Feature Representations

To better understand how the models differentiate between benign and malignant samples, we visualize the extracted feature embeddings using t-SNE (t-Distributed Stochastic Neighbor Embedding).

t-SNE Plots of Extracted Features

- CNN-Based Features (VGG16, VGG19, ResNet50)
  - Clusters show some overlap, meaning CNNs find it harder to distinguish classes.
- CapsNet Features
  - More dispersed clusters due to spatial feature encoding.
- ViT Features
  - Clear separation between malignant and benign samples, proving superior feature extraction.

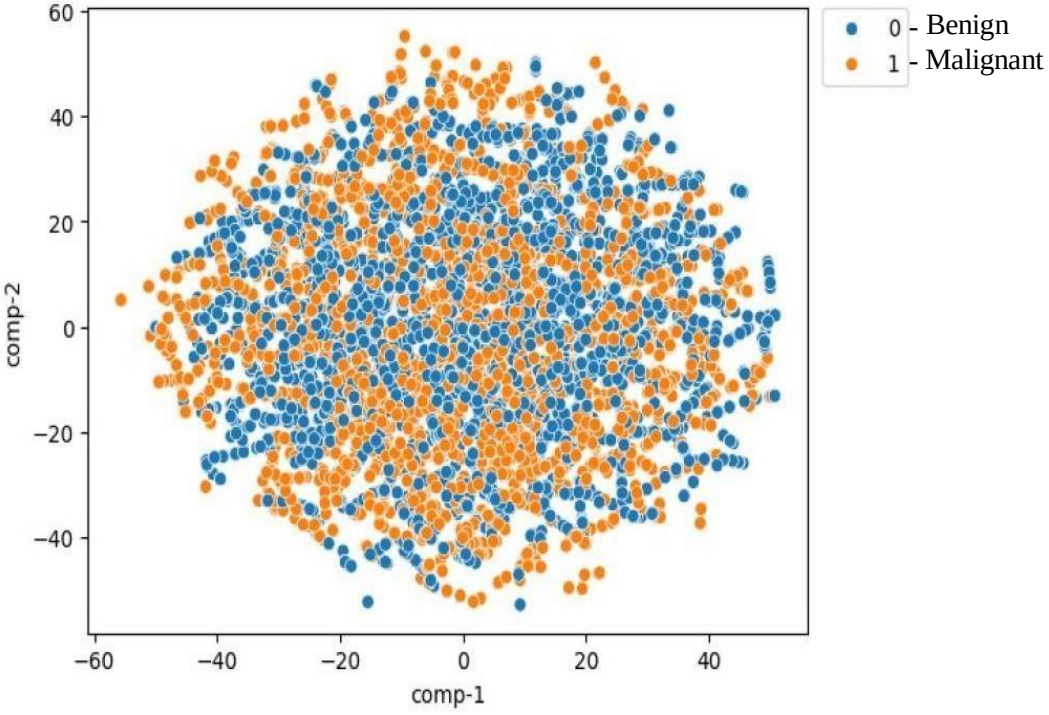


Fig.14 : Proposed Model t-SNE Visualization

CHAPTER 9

OUTPUT SCREENS

Malignant as output:-

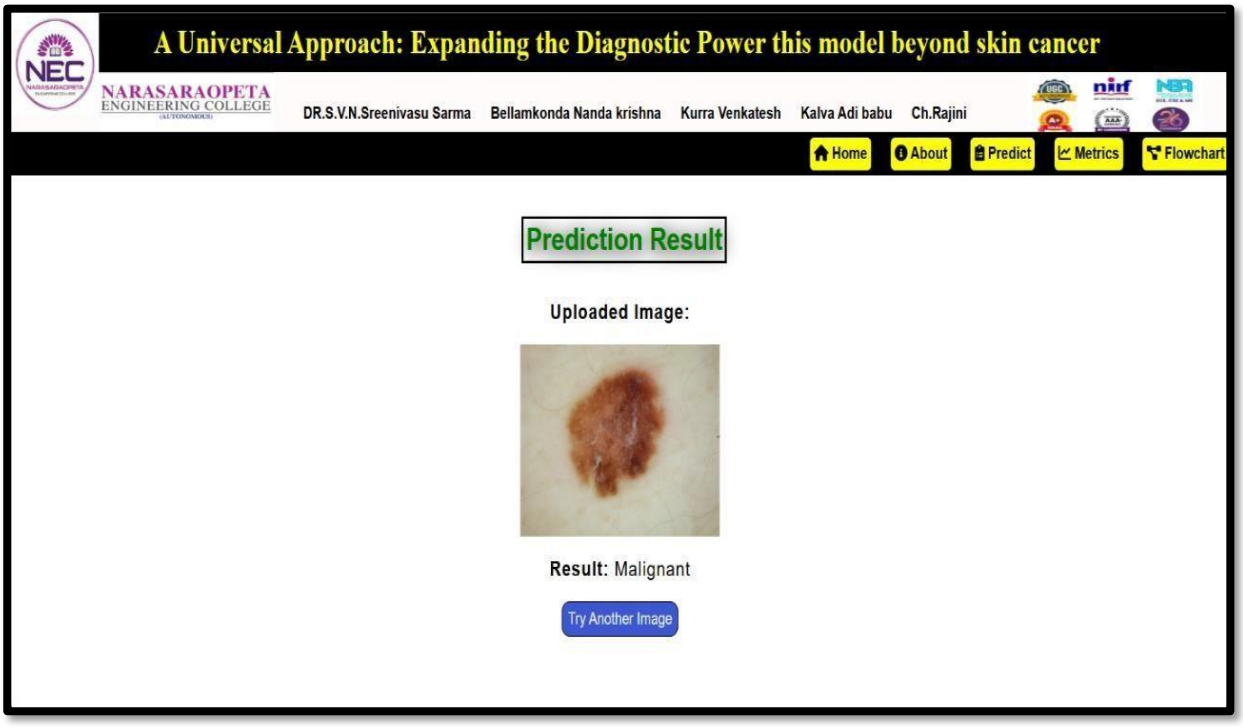


Fig.15 : Predicted as Malignant

Benign as output :-

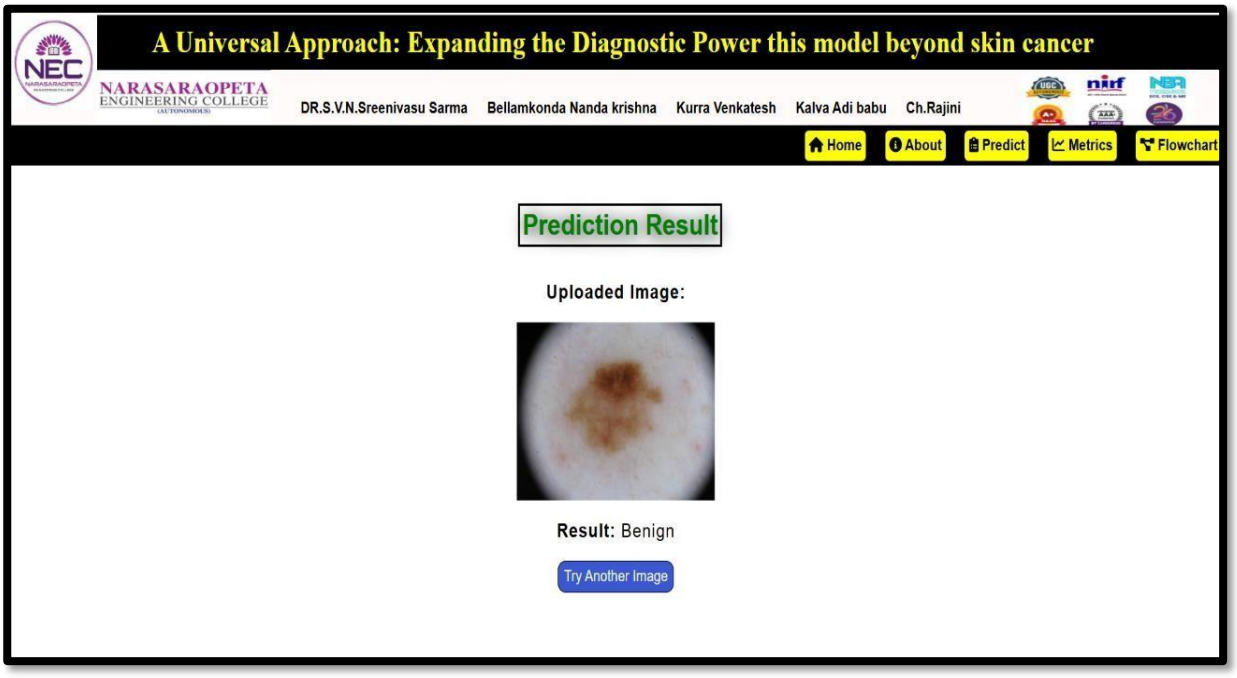


Fig.16 : Predicted as Benign

USER INTERFACE

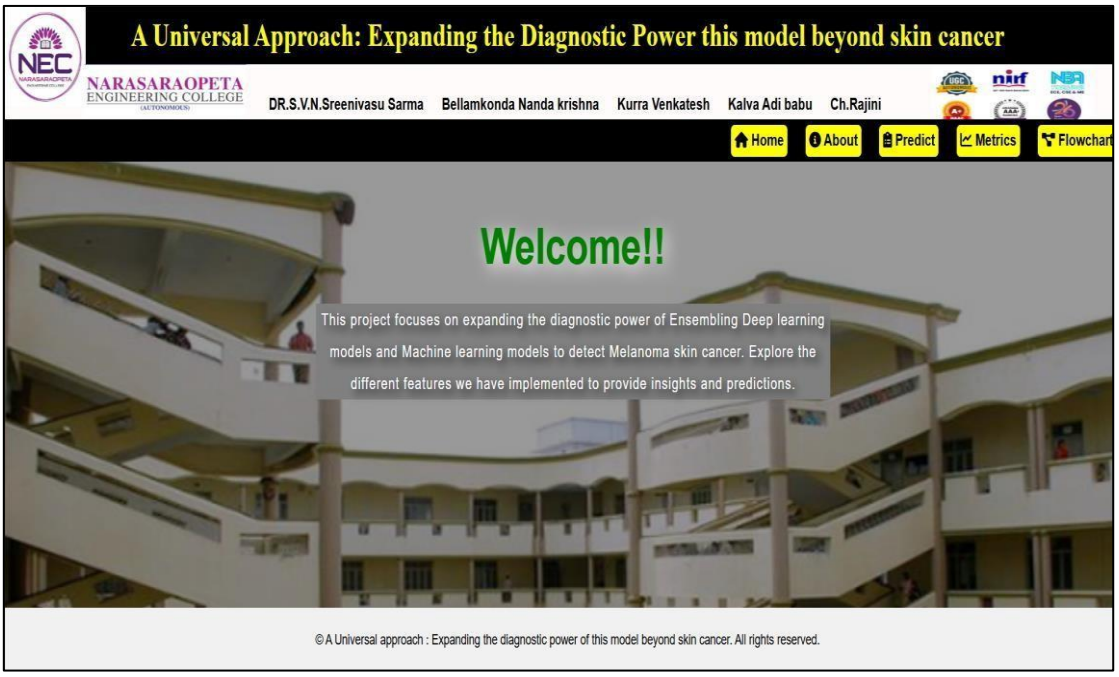


Fig.17 : Home Screen

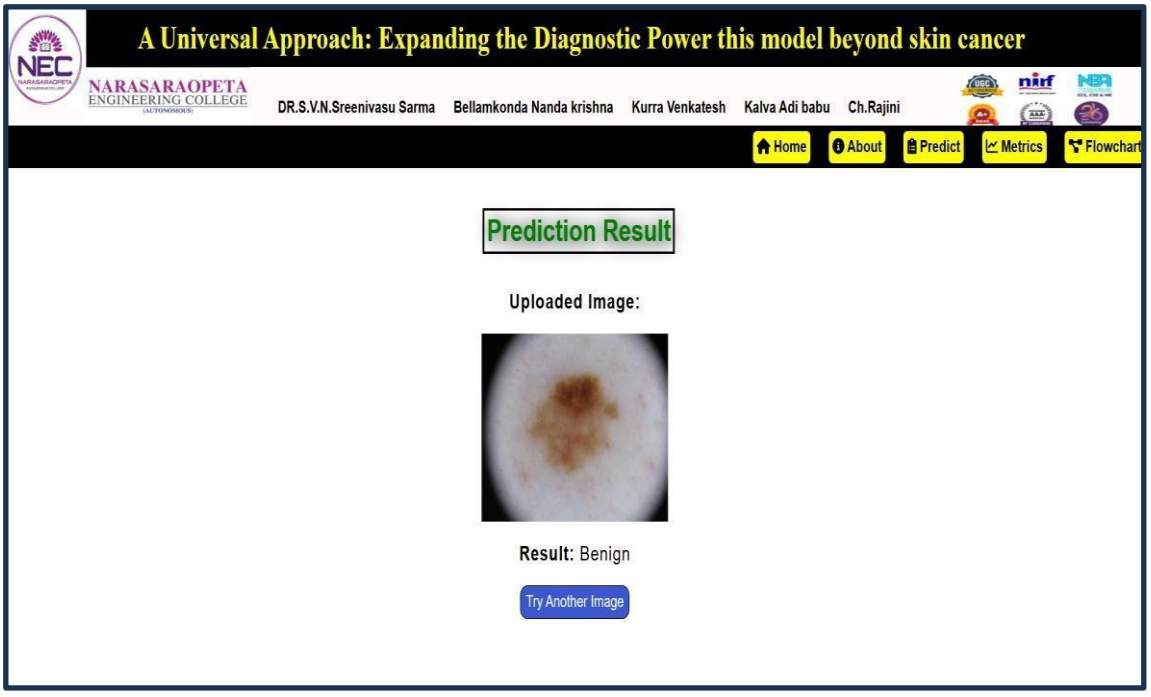


Fig.18 : Output screen

CHAPTER 10
CONCLUSION AND FUTURE WORK

CONCLUSION

Our paper comes with an innovative method that aims to improve melanoma detection by deep learning models VGG19 and VGG16, ResNet50, Vision Transformer (ViT), and CapsNet. Ensemble techniques by combining DL with ML classifiers XGBoost and SVC result in good outcomes, including being one of the top-scoring models from the ensemble ViT model. This study employed the Vision Transformer (ViT) model along with CNN architectures such as VGG19, VGG16, ResNet50, and Capsule Networks to improve melanoma detection accuracy.

Among these, the ViT model achieved the highest accuracy at 92.4%, highlighting its strength in capturing global image features essential for identifying melanoma patterns. ViT's self-attention mechanism enabled it to focus on critical area of images and capture dependencies often missed by CNNs, leading to higher precision, recall, and F1 scores. A notable recall score indicated effective true-positive identification, crucial for early, reliable skin cancer diagnosis. Ensemble method by combining ViT with other models through majority voting improves robustness to 92.3%. This ensemble has balanced local and global feature extraction, which increased the overall classification accuracy. The Vision Transformer performed well in this task, especially as part of an ensemble. Infact, the findings show that ViT, combined with CNNs, has much potential in image-based diagnostics, and using hybrid CNN-ViT models can improve performance even more in the future.

This work not only contributes to melanoma detection but also serves as the foundation for more generalized applications in the clinical sector. More research can be based on these directions in other medical domains by modifying these models to adapt to other medical imaging tasks

and encourage close collaboration with medical specialists as well as proper bedside integration and applicability in real-world medical scenarios. Such results therefore mean new models of performance testing and even working together with doctors open a wide range of huge opportunities in the use of AI for medical purposes beyond the example of merely diagnosing skin cancer.

FUTURE WORK

- Adapt the Model Architecture for Diverse Medical Applications: This objective focuses on modifying the current model architecture to be applicable to different medical domains beyond skin cancer. This could involve identifying transferable features and adapting algorithms for other image-based diagnoses.

- Evaluate Model Performance in New Medical Contexts: This objective emphasizes testing the adapted model in new medical fields. This might involve collecting specific datasets from other areas and evaluating its accuracy and generalizability for various medical image analysis tasks.

- Collaborate with Medical Specialists for Broader Clinical Integration: This objective highlights the importance of working with medical professionals like radiologists and pathologists for seamless integration of the model into their specific workflows. This collaboration would ensure the model addresses relevant needs and adheres to clinical practices for different medical applications.

REFERENCES

[1] Kassani, S. H., & Kassani, P. H. (2019). A comparative study of deep learning architectures on melanoma detection. Tissue and Cell, 58, 76-83.

[2] Al-Masni, M. A., Al-Antari, M. A., Park, H. M., Park, N. H., & Kim, T. S. (2019, May). A deep learning model integrating FrCN and residual convolutional networks for skin lesion segmentation and classification. In 2019 IEEE Eurasia conference on biomedical engineering, healthcare and sustainability (ECBIOS) (pp. 95-98). IEEE.

[3] Albraikan, A. A., Nemri, N., Alkhonaini, M. A., Hilal, A. M., Yaseen, I., & Motwakel, A. (2023). Automated Deep Learning Based Melanoma Detection and Classification Using Biomedical Dermoscopic Images. Computers, Materials & Continua, 74(2).

[4] Jojoa Acosta, M. F., Caballero Tovar, L. Y., Garcia-Zapirain, M. B., & Percybrooks, W. S. (2021). Melanoma diagnosis using deep learning techniques on dermatoscopic images. BMC Medical Imaging, 21, 1-11.

[5] Daghrir, J., Tlig, L., Bouchouicha, M., & Sayadi, M. (2020, September). Melanoma skin cancer detection using deep learning and classical machine learning techniques: A hybrid approach. In 2020 5th international conference on advanced technologies for signal and image processing (ATSIP) (pp. 1-5). IEEE.

[6] Sanketh, R. S., Bala, M. M., Reddy, P. V. N., & Kumar, G. P. (2020, May). Melanoma disease detection using convolutional neural networks. In 2020 4th international conference on intelligent computing and control systems (ICICCS) (pp. 1031-1037). IEEE.

[7] Lembhe, A., Motarwar, P., Patil, R., & Elias, S. (2023). Enhancement in Skin Cancer Detection using Image Super Resolution and Convolutional Neural Network. Procedia Computer Science, 218, 164-173

[8] Rodrigues, J. F., Brandoli, B., & Amer-Yahia, S. (2020, July). DermaDL: advanced convolutional neural networks for automated melanoma detection. In 2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS) (pp. 504-509). IEEE.

[9] Ghosh, S., Dhar, S., Yoddha, R., Kumar, S., Thakur, A. K., & Jana, N. D. (2024). Melanoma Skin Cancer Detection Using Ensemble of Machine Learning Models Considering Deep Feature Embeddings. Procedia Computer

Science, 235, 3007-3015.

[10] Kavitha, C., Priyanka, S., Kumar, M. P., & Kusuma, V. (2024). Skin Cancer Detection and Classification using Deep Learning Techniques. Procedia Computer Science, 235, 2793-2802.

[11] Kadampur, M. A., & Al Riyaee, S. (2020). Skin cancer detection: Applying a deep learning based model driven architecture in the cloud for classifying dermal cell images. Informatics in Medicine Unlocked, 18, 100282.

[12] Dildar, M., Akram, S., Irfan, M., Khan, H. U., Ramzan, M., Mahmood, A. R., ... & Mahnashi, M. H. (2021). Skin cancer detection: a review using deep learning techniques. International journal of environmental research and public health, 18(10), 5479.

[13] Lee, V., Singh, G., Trasatti, J. P., Bjornsson, C., Xu, X., Tran, T. N., ... & Karande, P. (2014). Design and fabrication of human skin by three-dimensional bioprinting. Tissue Engineering Part C: Methods, 20(6), 473-484.

[14] Mehr, R. A., & Ameri, A. (2022). Skin cancer detection based on deep learning. Journal of biomedical physics & engineering, 12(6), 559.

[15] Murugan, A., Nair, S. A. H., Preethi, A. A. P., & Kumar, K. S. (2021). Diagnosis of skin cancer using machine learning techniques. Microprocessors and Microsystems, 81, 103727.

[16] Shinde, P., & Ingle, Y. (2024). Skin Cancer Detection: A Review Using Machine Learning Techniques. *Asian Journal of Research in Computer Science, 17*(2), 15-26.

[17] Flosdorf, C., Engelker, J., Keller, I., & Mohr, N. (2024). Skin Cancer Detection Utilizing Deep Learning: Classification of Skin Lesion Images Using a Vision Transformer. *arXiv preprint arXiv:2407.18554*.

[18] Inthiyaz, S., Altahan, B. R., Ahammad, S. H., Rajesh, V., Kalangi, R. R., Smirani, L. K., Hossain, M. A., & Rashed, A. N. Z. (2023). Skin Disease Detection Using Deep Learning. *Advances in Engineering Software, 175*, 103361.

[19] Gajera, H. K., Nayak, D. R., & Zaveri, M. A. (2023). A Comprehensive Analysis of Dermoscopy Images for Melanoma Detection via Deep CNN Features. *Biomedical Signal Processing and Control, 79*, 104186.

[20] https://www.kaggle.com/datasets/hasnainjaved/melanoma-skin-cancer-dataset-of-10000-images

A Universal Approach: Expanding the Diagnostic Power of This Model Beyond Skin Cancer

Dr.S.V.N.Srinivasu[1][0000-0002-6049-911X], Bellamkonda Nanda Krishna[2], Kurra Venkatesh[2], Kalva Adi Babu[2], and Ch Rajani[3]

[1] Professor,Dept of CSE, Narasaraopeta Engineering College(Autonomous), Palnadu District, Andhra Pradesh, India. drsvnsrinivasu@gmail.com
[2] Dept of CSE, Narasaraopet Engineering College
nandakrishnabellamkonda@gmail.com, venkateshkurra1010@gmail.com, kalvaadibabu6@gmail.com
[3] Asst.Prof, Dept of CSE, Narasaraopeta Engineering College
rajani.kadiyala@gmail.com

Abstract. Within the fast-paced environment of artificial intelligence, deep learning algorithms have truly played a very special and important role in enhancing skin cancer detection; in fact, they may dramatically alter survival and early diagnosis rates. While most studies have focused on a single model of technique, our research utilizes a multi-framework model to optimize melanoma detection. In this study, we also combine Deep Convolutional Neural Networks VGG19(DCNN), VGG16, ResNet50, Capsule Networks (CapsNet), and vision transformers (ViT) for more profound images' features. Then the embedded features are fed into an ensemble model that involves five machine learning classifiers: Support Vector Classifier (SVC), XGBoost, Random Forest, K-Nearest Neighbors(KNN), and Logistic Regression via majority voting. This classification enhanced the accuracy of classification; ViT had attained its highest accuracy at 92.4%. The ensemble model we developed also performed well overall as it achieved 92.3% when used on a melanoma dataset. These results confirm that our ensemble approach significantly outmatch individual models and contributes more to the efficient detection of skin cancer.

Keywords: Melanoma Detection · CNN · ResNet50 · Vision Transformer(ViT) · CapsNet.

1 Introduction

Skin cancer, especially melanoma, that also referred to as one of the deadliest and most common cancers these days. It spreads aggressively which makes early diagnosis crucial. Traditional imaging, observational techniques, and biopsy are Consuming and prone to error, latest improvements in Machine Learning and Deep Learning technologies have improved diagnostic accuracy. According to dermoscopy, results show that in melanoma identification, convolutional neural networks work better. Kassani & Hosseinzadeh Kassani [1], 2019 conducted

comparative analysis of the deep architectures for melanoma detection with a focus on clinically efficient architectures. Al-Masni et al.[2], 2019 found that the efficiency of the process in segmentation and classification of the lesion increased with the use of a combination of FrCN and residual networks. Current works focus on the improvement of melanoma detection with hybrid classic and deep learning models. Deep learning-based methods for melanoma diagnosis are shown to be promising by Jojoa Acosta et al. [4] (2021). According to Daghrir et al. [5](2020), a hybrid model of combining classical and Deep Learning techniques shows promising ability to accurately identify skin cancer. Other major contributions in 2020 by Sanketh et al.[9] and Rodrigues et al.[8] also reflect the possible application of CNNs in skin cancer detection, where in CNN based architectures surpassed the existing traditional methods of diagnosis. Inventions like deep-learning model-based image super-resolution methods, much increased the resolution of dermoscopic images and consequently improved diagnosis accuracy greatly, as conceptualized by Lembhe et al. [7] in 2023. Studies highlight machine learning's potential in dermatology for enhanced melanoma detection and improved patient out-comes.

2 Literature Review

Kassani and Hosseinzadeh Kassani [1] executed a comparison of various deep learning approaches for identifying melanoma using dermoscopic images. Among the studied models- AlexNet, VGGNet16, VGGNet19, ResNet50, and Xception-the best performance with an accuracy of 92.08% was shown by ResNet50. This study demonstrated that ResNet50, when combined with augmentation and preprocessing techniques, has significantly improved performance, making it suitable for accurate skin lesion classification. Al-Masni et al. [2] came up with an integrated model using a new combination of full resolution convolution networks (FrCN) for segmentation and residual network systems, namely ResNet-50 for classification. Later, this was extended to dermoscopy images with skin lesions segmented from the rest of the skin with 94.03% accuracy and obtained a Jaccard similarity coefficient of 77.11%. For the ResNet-50 model classification, accuracy attained 81.57%, with an F1-score of 75.75%. Such a combined approach enabled the ResNet50 to extract more number of specific features from segmented lesions and further enhanced skin lesion diagnosis compared to regular models. Albraikan et al. [3] developed an advanced model for melanoma detection that included a number of deep learning methods. Accordingly, the model in view for this research integrates K-means clustering for segmentation, a Capsule Network (CapsNet) with Adagrad optimizer for feature extraction, and Crow Search Optimization technique with the Sparse Autoencoder approach for the classification. A benchmark dataset tested this automated system with great classification accuracy, hence very promising in enhancing melanoma detection. Jojoa Acosta et al. [4] introduced a strategy that makes use of the Mask Region Based CNN (Mask R-CNN) for lesion segmentation and ResNet152 for classification. Their model reported an accuracy of 90.4%,and with 82% of sensi-

tivity, and specificity of 92.5% in distinguishing malignant versus benign lesions. S jong, in essence, this can be a good combination with the current model for enhancing performance, as it will concentrate on the area of importance and deep feature mining for classification. Daghrir et al. [5] in 2020 have proposed one hybrid model for the detection of melanomas where deep learning is combined with classical machine learning methods. This paper used CNN, SVM, and KNN classifiers whose predictions combined through majority voting gave higher performance. Accuracy with CNN reached up to 85.5%, and the hybrid method further enhanced the accuracy to nearly 88.4% using a majority vote, thus justifying the point that integrated hybrid methods do give better results. Ghosh et al. [6] presented a hybrid ensemble model which includes deep learning embedded feature dimensions from the DCNN, Capsule Networks(CapsNet), and Vision Transformers. The next steps were to use machine learning classifiers after the concatenation of the feature vectors, and Logistic Regression, KNN, XGBoost, Support Vector Classifier(SVC), and Random Forest were employed through the mechanism of majority voting, yielding high accuracy of 91.6%. The ViT-based ensemble outperformed standalone DCNN and Caps-Net models by a margin, hence greatly improving melanoma detection classification performance. Lembhe et al. [7] (2023) investigated image super-resolution techniques to enhance the accuracy of deep learning models in skin cancer detection. In this study, low-resolution images were upscaled using InceptionV3, ResNet, and VGG16 models integrated with Image Super Resolution (ISR) and Generative Adversarial Networks (GANs). This was an approach that would enhance the skin lesions classification capability of the model. The use of ISR, therefore, coupled with GANs presents a promising future in enhancing diagnostic performance.Jose F. Rodrigues-Jr et al. [8], in their work entitled "DermaDL: Advanced Convolutional Neural Networks for Automated Melanoma Detection," proposed a new CNN architecture targeted for melanoma detection. This architecture was combined with state-of-the-art techniques such as Aggregated Transformations and Squeeze-and-Excite blocks. The result was that, without using general-purpose architectures, an AUC of 90% was achieved with their specialized network, beating the computational efficiency and melanoma detection accuracy for popular models like ResNet and VGG 5. Ravva Sai Sanketh et al. [9] (2020) proposed, in a research paper titled "Melanoma Disease Detection Using Convolutional Neural Networks," using convolutional neural networks for early detection of melanoma and non-melanoma skin cancers. They used the ISIC dataset and reported an accuracy of 91%. This CNN-based model outperforms the manual detection methodologies and tries to assist physicians with the correct classification of skin cancer without resorting to clinical procedures.

3 Proposed Methodology

3.1 Dataset Collection

This dataset was mined from the Kaggle [11] and is called the ”Melanoma Skin Cancer Dataset of 10,000 Images.” It contains 10,000 high-resolution dermoscopic

images for identifying and diagnosing skin lesions that classified as malignant or benign. It trains an advanced deep learning model capable of differentiating benign conditions from the grave form of skin cancer-the melanoma.Train and Test Folders contain Benign and Malignant.This folder has 9,605 training images: 5,773 benign and 3,832 malignant, to train the model for differentiation.Total 1,000 images, split in equal proportions into 500 benign and malignant samples for model testing.Image enhancementing techniques like the rotation, flipping, and zooming are used. Preprocessing of data was done prior to sending it for training, thus improving the resolution and format to support better feature learning.

3.2 Preprocessing

Resizing These dataset images were uniformly resized to 128x128 pixel resolution. This preprocessing is important because it usually gives uniformity in the size of the images, a requirement that most deep learning models would want, including VGG16 and VGG19. It will reduce computational complexities, hence reducing training time without the loss of the essential features of an image.

Colorspace Conversion The images which are by default loaded through OpenCV are in BGR format. Since the deep learning models, implicitly expect input formatted as RGB, conversion of images to RGB color space was performed using the function cvtColor of the OpenCV library.

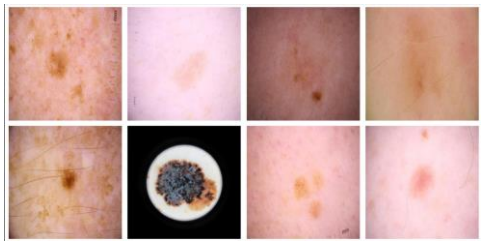


Fig. 1: Images before conversion

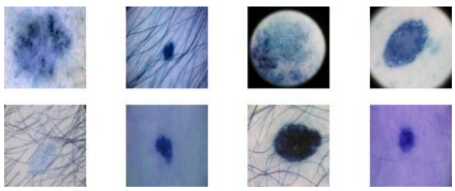


Fig. 2: Images after conversion

Shuffling This randomizes the presentation order of images across the benign and malignant classes. It reduces the amount of bias. Shuffling will make sure that, during training, changing patterns of images from both classes are seen by the model at regular instances, reducing the risk of overfitting on the early-patterned ones.

Normalization The Pixel values have been scaled to a range from 0 to 1 because all the RGB values have been divided by 255 to normalize the data within this range. Hence, model training will be faster because all the magnitudes of data are brought within the same range, and there is no saturation from the neural network's activation functions during training.

Label Encoding Categorical labels of the dataset, that is, 'benign' and 'malignant' were encoded into their numerical form in the first place. Label encoding hanges categorical data into integers-0 for 'benign' and 1 for 'malignant'. By doing so, all the algorithms understood what exactly these labels mean while training.

Train-Test Split Further, the preprocessed dataset was subsequently divided into two distinct subsets-testing 20% and training 80%. About 80% of the total data were put into the training set, while the remaining percentage of about 20% was kept for testing. This guarantees that a significant portion of the data is allocated for model training while it is being tested with unseen data so as to assess its ability for generalization and hence give good results for new inputs. This prevents the overfitting of data and thus gives a good measure of the performance of the model.

3.3 Models

This work follows the model approach hybrid deep learning with machine learning to classify melanoma images. In the proposed methodology, the deep learning model that has been pre-trained will be used for the feature extraction of images, while the actual classification is to be done through the machine learning model. Along with this, the approach also will employ an ensemble method to give better accuracy to the system. In detail, this will be briefed below:

Deep Learning architectures for Extracting Features

VGG16 and VGG19: We used pre-trained VGG16 and VGG19 models to extract image features. The network learns hierarchically; hence we could use them directly to extract good features for melanoma without retraining the whole deep network.

ResNet50: A pre-trained model learns the residual functions really well on deep structures by using skip connection. It performs well in the deep feature extraction for low and high level features of images.

Capsule Network: CapsNet strengthens spatial relations and recognizes complicated patterns. Unlike most convolution networks, its ability will be of recognizing part- whole relations, which has much significance in the detection of melanomas in skin lesions.

Transformer Vision ViT: The model uses the Vision Transformer, where self-attention captures global dependencies in input data. ViT is particularly efficient for image classification, this contribution well captures long- range dependencies, making it an ideal choice for this work.

Machine Learning algorithms for Classification tasks The following machine learning models are used for classification tasks after feature extraction

Logistic Regression: Logistic regression employs a linear model that sorts the images in some order by using extracted features.

Random Forest: Random Forest is an ensemble decision tree-based model that integrates multiple decision trees to provide useful classification for the target images.

K-Nearest Neighbors (KNN): KNN is a non-parametric distance-based classifier that compares test samples against their k-nearest neighbors within feature space.

Support Vector Classifier (SVC): SVC classifies the images using a kernel-based approach that seeks the maximum margin separation between classes, adopting a Radial Basis Function (RBF) kernel.

XGBoost: XGBoost is a gradient boosting machine learning model known for its efficiency and performance in classification tasks.

Ensemble Voting Classifier A number of model predictions are combined into a single prediction for better results using an ensemble voting classifier. This is achieved as: Individual predictions of VGG16, VGG19, ResNet50, CapsNet, and Vision Transformer are aggregated. The hard voting strategy was used; thus, the classification result depends on the majority vote of all the models. Hence, one advantage of this ensembling technique is that the inherent variance and bias of single models are reduced in an attempt to give better overall accuracy. It is a model that has been trained by the proposed approach on the melanoma dataset. Some of the metrics for the assessment of the performance of the model includes F1 score, recall, precision, and the accuracy. In that regard, higher results were

allowed to the classifier with ensemble voting in comparison with individual deep learning and machine learning models that proved the validity of the approach presented.

4 Proposed Model

The proposed Vision Transformer (ViT) model used attention mechanisms for long-range dependencies in image classification. The input shape is (128, 128, 3) with four transformer blocks, an embedding dimension of 32, and eight attention heads. A feed-forward and drop-out of 0.1 prevent overfitting and enhance classification based on feature points.

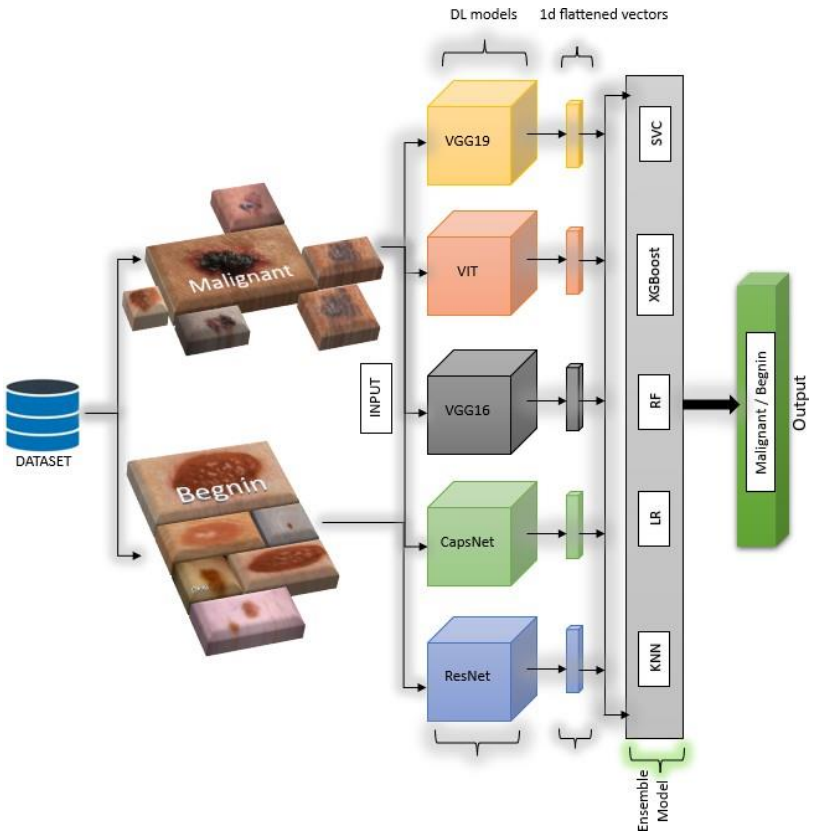


Fig. 3: Proposed Architecture for Melanoma Detection Using Vision Transformer.

Utilizing Adam-Adaptive moment estimation algorithm, learning a rate of 0.0001 has been set for multi-class classification with sparse categorical cross-entropy loss function is used ,with a Batch size of 32 per epoch and is tunable for best performance. Regulation at 0.0005 and max norm of 5.0 avoids overfitting and enhances generalization. The accuracy in ViT was 0.924, and recall, precision and F1 scores were the same in testing. A fair metric indicates that an accurate model balances both true positives and negatives with the least amount of bias. It classifies test images based on the actual ground truth labels. The metrics are computed and the correspondences in t-SNE coupled with the confusion matrices analyze classification strengths as well as improvement areas.

Table 1: Transformer Model Architecture Parameters.

Parameter	Value
Input Shape	(128, 128, 3)
Number of Transformer Blocks	4
Embedding Dimension	32
Number of Attention Heads	8
Feed-Forward Dimension	32
Dropout Rate	0.1

Table 2: Training Setup.

Parameter	Value
Optimizer	Adam (learning rate: 0.0001)
Loss Function	Sparse Categorical Cross-Entropy
Batch Size	32
Regularization Parameters	0.0005
Max Norm	5.0

5 Analytical Comparison

Various machine learning models and deep learning models were analyzed in this work to detect melanoma skin cancer. The different models that have been tried in this work include KNN, Logistic Regression, Random Forest, XGBoost, SVC, Ensemble models, and deep learning models includes VGG 19, VGG 16, ViT, ResNet50, and CapsNet. All these models are assessed based on their performance by considering the precision, recall score, accuracy and F1 score.

5.1 Performance Table

Table 3 gives all models performance. Vision Transformer ViT stands the best among all models with 92.4% accuracy. Following ViT, VGG 19 with 92.2% then VGG16 with 91.7% and at last ResNet50 and CapsNet with 90.6% and 90.3% respectively.

Table 3: Accuracy Comparison of Various Models.

Model	Accuracy
VGG19	92.2%
VGG16	91.7%
CapsNet	90.3%
ViT	92.4%
ResNet50	90.6%

5.2 Training and Testing Accuracy Graphs

Comparing the training and testing accuracy graphs of VGG19, VGG16, CapsNet, ViT, and ResNet50. VGG19 reached 92.2 training accuracy with strong feature extraction and low overfitting, while VGG16 lagged at 91.7% with lower effectiveness and more loss. It achieved 90.3% accuracy, surpassing VGG models due to CapsNet's better spatial hierarchy capture, though it required more epochs to stabilize. ViT reached 92.3% accuracy with long dependency attention and was initially slow but matched others in generalization and robustness. ResNet50 scored 90.6% accuracy, providing stability and outperforming ViT and VGG19.

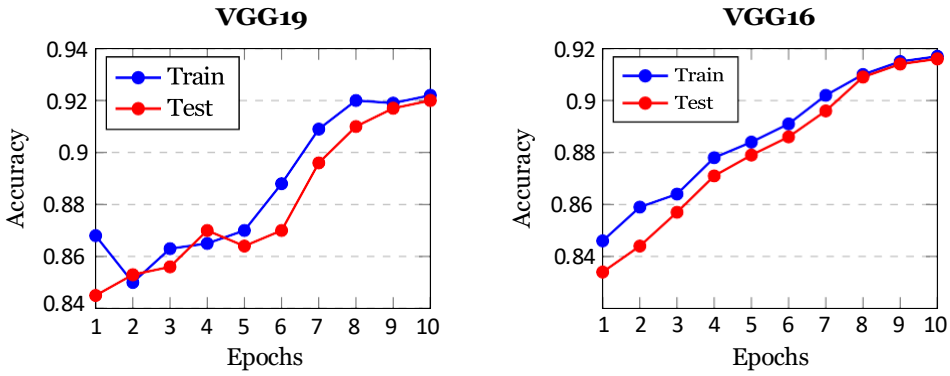


Fig. 4: Training and Testing Accuracy graphs of all models

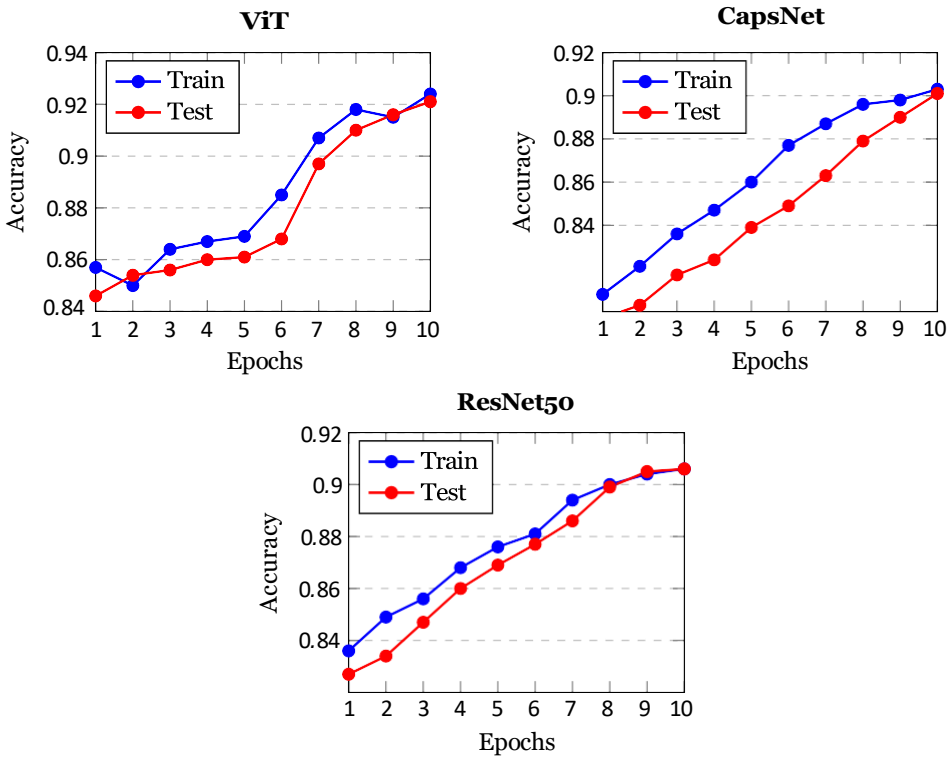


Fig. 5: Training and Testing Accuracy graphs of all models

5.3 Evaluation Metrics

Below Table 4 is the evaluation matrix for all models: comparing all the models across the metrics precision,recall,f1 score and accuracy. In Table 4, the most critical metrics include accuracy, precision, recall, and F1 score, which emphasizes the ability of each model to accurately detect melanoma. Accuracy measures overall correct classifications; hence, both ViT (92.4%) and Ensemble (92.3%) have robust performance in correctly classifying benign and malignant cases. Precision indicates each model's ability to avoid false positives and thus avoid unnecessary treatments, where ViT is strong with high precision. Recall (sensitivity) means fewer false negatives, that is fewer missed melanomas, and ViT does well with a recall of 92.4%. The F1 score, a measure of finding the balance between precision and recall, is also in favor of ViT and Ensemble to classify with a good accuracy and balance. CapsNet and ResNet50 promise to do better in complex feature detection but slightly lower F1 scores indicate some compromise between false positives and false negatives. The comparison once again establishes ViT for high-stakes medical diagnosis.

Table 4: Comparison of Models Across Metrics

MODEL	Metric	Vgg19	Vgg16	CapsNet	ViT	ResNet
KNN	ACC	91.7	91.1	90.4	91.9	90.5
	PRE	91.7	91.1	90.4	91.9	90.5
	REC	91.7	91.1	90.4	91.9	90.5
	F1	91.7	91.1	90.4	91.9	90.5
LR	ACC	90.7	91.2	90.3	92.2	90.9
	PRE	90.7	91.2	90.3	92.2	90.9
	REC	90.7	91.2	90.3	92.2	90.9
	F1	90.7	91.2	90.3	92.2	90.9
RF	ACC	91.3	91.2	90.4	89.5	90.8
	PRE	91.3	91.2	90.4	89.5	90.8
	REC	91.3	91.2	90.4	89.5	90.8
	F1	91.3	91.2	90.4	89.5	90.8
XGBOOST	ACC	92.3	91.5	90.2	91.9	90.5
	PRE	92.3	91.5	90.2	91.9	90.5
	REC	92.3	91.5	90.2	91.9	90.5
	F1	92.3	91.5	90.2	91.9	90.5
SVC	ACC	91.4	91.0	90.3	92.4	90.6
	PRE	91.4	91.0	90.3	92.4	90.6
	REC	91.4	91.0	90.3	92.4	90.6
	F1	91.4	91.0	90.3	92.4	90.6
ENSEMBLE	ACC	92.2	91.7	90.3	92.3	90.6
	PRE	92.2	91.7	90.3	92.3	90.6
	REC	92.2	91.7	90.3	92.3	90.6
	F1	92.2	91.7	90.3	92.3	90.6

5.4 Model Summary

The Table 5 provides an overview of the models applied for melanoma detection, highlighting their parameters and computational complexities. VGG19 and ResNet50 stand out for their high parameter counts (20 million and 23.5 million, respectively), indicating robust architectures with significant feature extraction capability. VGG16 is simpler with 14.7 million parameters, making it slightly faster but with a marginally reduced feature set. CapsNet, with approximately 7 million parameters, utilizes capsule layers for improved spatial awareness, beneficial for detailed lesion detection. ViT, with only 218,946 parameters, is highly efficient; it leverages self-attention mechanisms to capture image dependencies, achieving high accuracy with minimal computational cost.

Table 5: Model Parameters Comparison.

Model	Parameters
VGG19	20,024,384
VGG16	14,714,688
CapsNet	7 million (approx)
ViT	218,946
ResNet50	23,587,712

5.5 Model Parameter Tables

Table 6 provides the parameters of the VGG19 and VGG16 models outline their design and functional differences. Both models accept input images of shape (128, 128, 3) and include fully trainable layers, allowing for comprehensive feature extraction from dermoscopic images. The base models, VGG19 and VGG16, are similar in architecture, but VGG19 has additional convolutional layers, resulting in a total of 20 million parameters compared to 14.7 million in VGG16. This added complexity allows VGG19 to capture more detailed patterns but also increases computational demand. For feature extraction, both models output a standardized shape of (9605, 512), which aids in efficient downstream classification.

Table 6: Comparison of VGG19 and VGG16 Models.

Parameters	Value(VGG19)	Value(VGG16)
Input shape	(128,128,3)	(128,128,3)
Base Model	VGG19	VGG16
Trainable Layers	True	True
Feature Extraction Shape	(9605,512)	(9605,512)
Standardize Features Shape	(9605,512)	(9605,512)
Number of Classes	2	2
KNN Neighbors	10	10
Random Forest Estimators	50	50
XG Boost Tree Method	Auto	Auto
SVC Kernel	RBF	RBF
SVC C	0.65	0.65
Voting Classifier	Hard	Hard

The ResNet50 model in Table 7, with 23.5 million parameters and trained on ImageNet weights, utilizes skip connections to manage deep layers efficiently. Its robust architecture excels in extracting both high-level and low-level image features, making it well-suited for detailed melanoma analysis.

Table 7: Parameters for the ResNet50 Model.

Parameter	Value
Input_shape	(128,128,3)
Num_classes	3
Number of parameters	23,587,712
Optimizer	adam
Loss Function	categorical crossentropy
Epochs	10
Batch_size	32

CapsNet in Table 8 employs approximately 7 million parameters with a unique capsule structure that captures spatial hierarchies, improving pattern recognition in skin lesions. Its architecture includes dynamic routing, which enhances the model's accuracy in distinguishing subtle features within dermoscopic images.

Table 8: Table of Parameters for CapsNet.

Parameters	Value
Input_shape	(128,128,3)
Num_classes	3
Conv1_filters	256
Conv1_kernel_size	9
Conv1_padding	Valid
Primary_caps_dim	8
Primary_caps_n_caps	32
Primary_caps_strides	2
Conv_caps_dim	16
Conv_caps_n_caps	32
Conv_caps_kernel_size	9
Conv_caps_padding	Valid
Dense_caps_dim	16
Dense_caps_n_caps	3
Decoder_network	[512,1024,784]
Optimizer	Adam
Learning_rate	0.0001
Epochs	10
Batch_size	32

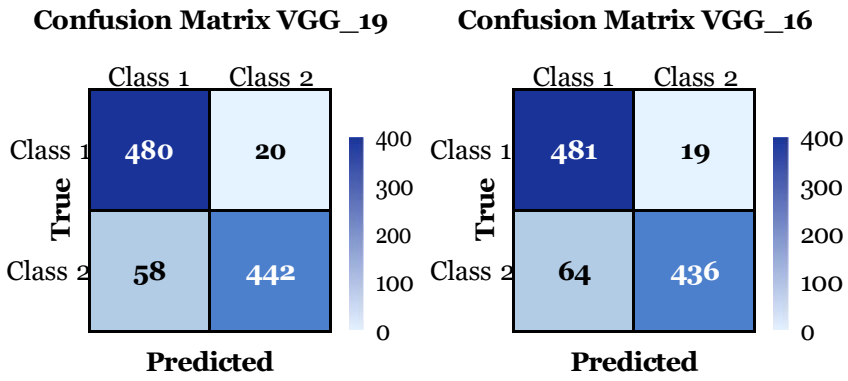
The Vision Transformer (ViT) model in Table 9, with 218,946 parameters and four transformer blocks, is highly efficient in processing images through self-attention mechanisms. Despite its compact size, ViT's ability to capture long-range dependencies enables it to achieve high accuracy with minimal computational cost.

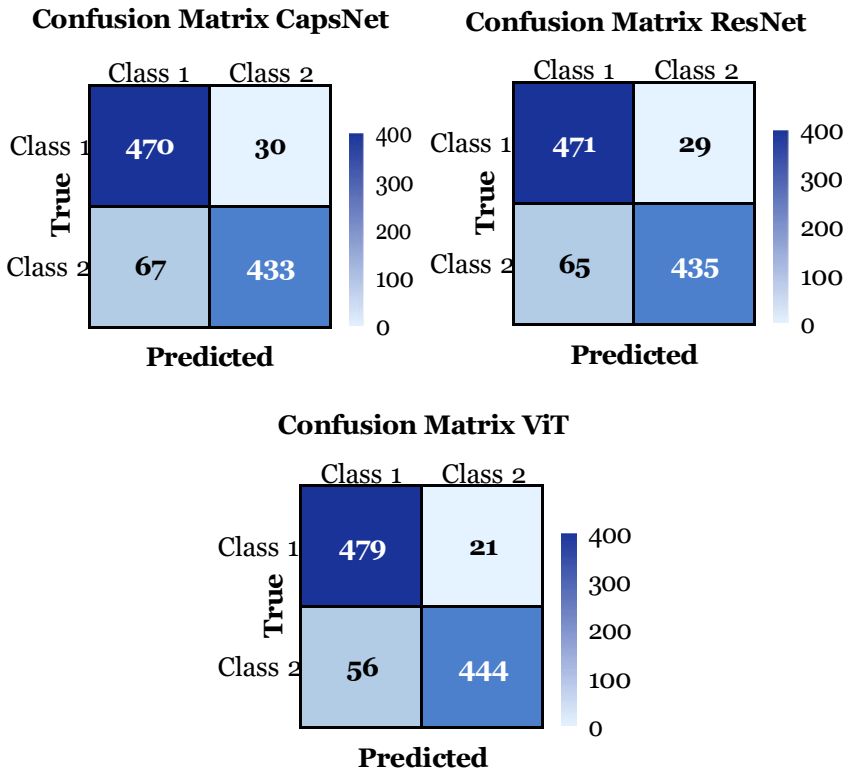
Table 9: Parameters for the ViT Model.

Parameter	Value
Input_shape	(128,128,3)
Num_classes	2
Num_transformer_blocks	4
Embed_dim	32
Num_heads	8
Ff_dim	32
Dropout	0.1
Optimizer	Adam
Learning_rate	0.0001
Epochs	10
Batch_size	32
Regularization_scale	0.0005

5.6 Confusion Matrix

We compared the confusion matrixes of deep models such as VGG19, VGG16, CapsNet, ViT, and ResNet50 and visually identify the true positives, true negatives, false positives, and false negatives. VGG19 balances TPs and TNs well but has occasional FPs due to benign misclassifications. VGG16 shows a similar pattern but with more FNs, indicating potential melanoma misses. CapsNet's feature hierarchy boosts TPs but may increase FPs due to spatial emphasis. ResNet50 maintains low FPs and balanced TPs/TNs with its skip connections, reducing misdiagnoses. ViT achieves the highest accuracy, minimizing both FPs and FNs through its self-attention for complex dependencies, making it highly effective for melanoma detection.





6 Conclusion

Our paper comes with an innovative method that aims to improve melanoma detection by deep learning models VGG19 and VGG16, ResNet50, Vision Transformer (ViT), and CapsNet. Ensemble techniques by combining DL with ML classifiers XGBoost and SVC result in good outcomes, including being one of the top-scoring models from the ensemble ViT model. This study employed the Vision Transformer (ViT) model along with CNN architectures such as VGG19, VGG16, ResNet50, and Capsule Networks to improve melanoma detection accuracy. Among these, the ViT model achieved the highest accuracy at 92.4%, highlighting its strength in capturing global image features essential for identifying melanoma patterns. ViT's self-attention mechanism enabled it to focus on critical areas of images and capture dependencies often missed by CNNs, leading to higher precision, recall, and F1 scores. A notable recall score indicated effective true-positive identification, crucial for early, reliable skin cancer diagnosis. Ensemble method by combining ViT with other models through majority voting improves robustness to 92.3%. This ensemble has balanced local and global feature extraction, which increased the overall classification accuracy. The Vision Transformer performed well in this task, especially as part of an ensemble. In fact, the findings show that ViT, combined with CNNs, has much potential in

image-based diagnostics, and using hybrid CNN-ViT models can improve performance even more in the future. This work not only contributes to melanoma detection but also serves as the foundation for more generalized applications in the clinical sector. More research can be based on these directions in other medical domains by modifying these models to adapt to other medical imaging tasks and encourage close collaboration with medical specialists as well as proper bedside integration and applicability in real-world medical scenarios. Such results therefore mean new models of performance testing and even working together with doctors open a wide range of huge opportunities in the use of AI for medical purposes beyond the example of merely diagnosing skin cancer.

References

1. Kassani, S.H., Kassani, P.H.: A comparative study of deep learning architectures on melanoma detection. Tissue and Cell **58**, 76–83 (2019)
2. Al-Masni, M.A., Al-Antari, M.A., Park, H.M., Park, N.H., Kim, T.S.: A deep learning model integrating FrCN and residual convolutional networks for skin lesion segmentation and classification. In: 2019 IEEE Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS), pp. 95–98. IEEE (2019)
3. Albraikan, A.A., Nemri, N., Alkhonaini, M.A., Hilal, A.M., Yaseen, I., Motwakel, A.: Automated deep learning-based melanoma detection and classification using biomedical dermoscopic images. Computers, Materials & Continua **74**(2) (2023)
4. Jojoa Acosta, M.F., Caballero Tovar, L.Y., Garcia-Zapirain, M.B., Percybrooks, W.S.: Melanoma diagnosis using deep learning techniques on dermatoscopic images. BMC Medical Imaging **21**, 1–11 (2021)
5. Daghrir, J., Tlig, L., Bouchouicha, M., Sayadi, M.: Melanoma skin cancer detection using deep learning and classical machine learning techniques: A hybrid approach. In: 2020 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), pp. 1–5. IEEE (2020)
6. Ghosh, S., Dhar, S., Yoddha, R., Kumar, S., Thakur, A.K., Jana, N.D.: Melanoma skin cancer detection using ensemble of machine learning models considering deep feature embeddings. Procedia Computer Science **235**, 3007–3015 (2024)
7. Lembhe, A., Motarwar, P., Patil, R., Elias, S.: Enhancement in skin cancer detection using image super resolution and convolutional neural network. Procedia Computer Science **218**, 164–173 (2023)
8. Rodrigues, J.F., Brandoli, B., Amer-Yahia, S.: DermaDL: Advanced convolutional neural networks for automated melanoma detection. In: 2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS), pp. 504–509. IEEE (2020)
9. Sanketh, R.S., Bala, M.M., Reddy, P.V.N., Kumar, G.P.: Melanoma disease detection using convolutional neural networks. In: 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 1031–1037. IEEE (2020)
10. Kavitha, C., Priyanka, S., Kumar, M.P., Kusuma, V.: Skin cancer detection and classification using deep learning techniques. Procedia Computer Science **235**, 2793–2802 (2024)
11. Melanoma Skin Cancer Dataset of 10000 Images, https://www.kaggle.com/datasets/hasnainjaved/melanoma-skin-cancer-dataset-of-10000-images, last accessed 2023/10/25

ORIGINALITY REPORT

6%	4%	6%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Dinesh Goyal, Bhanu Pratap, Sandeep Gupta, Saurabh Raj, Rekha Rani Agrawal, Indra Kishor. "Recent Advances in Sciences, Engineering, Information Technology & Management - Proceedings of the 6th International Conference “Convergence2024” Recent Advances in Sciences, Engineering, Information Technology & Management, April 24–25, 2024, Jaipur, India", CRC Press, 2025 Publication	1%
2	pdffox.com Internet Source	1%
3	H.L. Gururaj, Francesco Flammini, S. Srividhya, M.L. Chayadevi, Sheba Selvam. "Computer Science Engineering", CRC Press, 2024 Publication	<1%
4	"Advances in Artificial Intelligence and Machine Learning in Big Data Processing", Springer Science and Business Media LLC, 2025 Publication	<1%

5	www.frontiersin.org Internet Source	<1%
6	www.sctce.ac.in Internet Source	<1%
7	usermanual.wiki Internet Source	<1%
8	Subhayu Ghosh, Sandipan Dhar, Raktim Yoddha, Shivam Kumar, Abhinav Kumar Thakur, Nanda Dulal Jana. "Melanoma Skin Cancer Detection Using Ensemble of Machine Learning Models Considering Deep Feature Embeddings", Procedia Computer Science, 2024 Publication	<1%
9	Sukhpreet Kaur, Sushil Kamboj, Manish Kumar, Arvind Dagur, Dhirendra Kumar Shukla. "Computational Methods in Science and Technology", CRC Press, 2024 Publication	<1%
10	Surajit Sarma, Nabankur Pathak. "Chapter 37 An Emerging Area of Research: Building an Assamese AI Chatbot for Educational Institutions Using Bi-LSTM Model", Springer Science and Business Media LLC, 2024 Publication	<1%
11	www.yumpu.com Internet Source	<1%

12	export.arxiv.org Internet Source	<1%
13	repositori.upf.edu Internet Source	<1%
14	Mohammed Rakeibul Hasan, Mohammed Ishraaf Fatemi, Mohammad Monirujjaman Khan, Manjit Kaur, Atef Zaguia. "Comparative Analysis of Skin Cancer (Benign vs. Malignant) Detection Using Convolutional Neural Networks", Journal of Healthcare Engineering, 2021 Publication	<1%
15	mdpi-res.com Internet Source	<1%
16	peerj.com Internet Source	<1%
17	www.igi-global.com Internet Source	<1%
18	Shivinder Nijjer, Sahil Raj. "Predictive Analytics in Human Resource Management - A Hands-on Approach", Routledge, 2020 Publication	<1%
19	link.springer.com Internet Source	<1%
20	ebin.pub Internet Source	<1%

21	www.mdpi.com Internet Source	<1%
22	Thompson Stephan. "Artificial Intelligence in Medicine", CRC Press, 2024 Publication	<1%

Exclude quotes	On
Exclude bibliography	On
Exclude matches	Off

CERTIFICATE
OF PRESENTATION

Paper ID: 174

Paper Title: A Universal Approach: Expanding the Diagnostic Power of This Model Beyond Skin Cancer

Authors: Dr.S.V.N.Srinivasu, Bellamkonda Nanda krishna , Kurra Venkatesh , Kalva Adi Babu, Ch Rajani

*This is to certify that **BELLAMKONDA NANDA KRISHNA** of **Narasaraopeta Engineering College** has presented the paper at the **Second International Conference on Advanced Computing, Machine Learning, Robotics and Internet Technologies (AMRIT-2024)**, organised in hybrid mode by the Department of Computer Science & Engineering and Department of Electrical Engineering, National Institute of Technology Agartala, Tripura, jointly organised by the Department of Computer Science, Assam University, Silchar, Assam, during November 8-9, 2024.*

Dr. Prodipto Das
Program Chair, AMRIT-2024
Dept. of CS, AU, Silchar

Dr. Sadhan Gope
Organizing Chair, AMRIT-2024
Dept. of EE, NIT Agartala

Dr. Ranjita Das
Organizing Chair, AMRIT-2024
Dept. of CSE, NIT Agartala