**A DEEP LEARNING APPROACH FOR SKIN DISEASE CLASSIFICATION**

# ABSTRACT

This research introduces an innovative framework for automating the classification of skin diseases utilizing deep learning methodologies, specifically leveraging the DenseNet121 and VGG19 architectures. Conventional manual methods of skin disease classification often suffer from imprecision and inefficiency, necessitating the development of automated systems to enhance accuracy and productivity in the healthcare sector. The primary objective of this study is to construct a robust classification system capable of accurately categorizing skin diseases based on various visual characteristics, such as texture and color. Both DenseNet121 and VGG19 architectures are employed in the classification process to harness the power of convolutional neural networks (CNNs).

The proposed methodology involves optimizing pre-trained DenseNet121 and VGG19 models using a diverse dataset comprising images of various skin diseases. This dataset encompasses a wide range of skin conditions to ensure the model's adaptability and effectiveness. The performance of the developed system is rigorously evaluated through comprehensive experimentation and analysis.

The DenseNet121 model exhibits promising results in classifying skin diseases, achieving an accuracy of 87%. This notable accuracy is attained after hyper-parameter tuning with 25 epochs, a batch size of 64, and a learning rate of 0.001. Similarly, the VGG19 architecture achieves a commendable accuracy of 94% after fine-tuning with 25 epochs, a batch size of 32, and a learning rate of 0.001.

Furthermore, the scalability and computational efficiency of the proposed approach are assessed, demonstrating its ability to handle large datasets and process images in real-time.

Overall, this research contributes to the advancement of automated skin disease classification, offering practical implications for improving diagnostic accuracy and efficiency in healthcare. The outstanding accuracy achieved by the DenseNet121 and VGG19 models underscores their potential as valuable tools for enhancing decision-making processes and patient care in dermatology.

Overall, this research contributes to the advancement of automated skin disease classification, offering practical implications for improving diagnostic accuracy and efficiency in healthcare. The outstanding accuracy achieved by the DenseNet121 and VGG19 models underscores their potential as valuable tools for enhancing decision-making processes and patient care in dermatology.

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
|  | **Page** |
|  |  |
| **ACKNOWLEDGEMENT** | **ii** |
| **APPROVAL AND DECLARATION** | **iii** |
| **BONOFIDE CERTIFICATE** | **iv** |
| **ABSTRACT** | **v** |
| **TABLE OF CONTENTS** | **vii** |
| **CHAPTER 1** |  |
| **INTRODUCTION** | 1 |
| 1.1 Problem Statement | 2 |
| 1.2 Need for Classification | 2 |
| 1.3 Applications of Skin Disease classification | 3 |
| 1.4 Existing System | 4 |
| 1.5 Proposed System | 5 |
| 1.6 Objective of the Proposed System | 6 |
| 1.7 Transfer Learning | 7 |
|  |  |
| **CHAPTER 2** |  |
| **LITERATURE SURVEY** | 9 |
| **PROJECT CHARTER** | 13 |
|  |  |

|  |  |
| --- | --- |
| **CHAPTER 3** |  |
| **REQUIREMENT SPECIFICATION** | 15 |
| 3.1 Software Requirements | 15 |
| 3.2 Hardware Requirements | 15 |
| 3.3 System Tools | 15 |
|  |  |
| **CHAPTER 4** |  |
| **METHODOLOGY** | 16 |
| 4.1 Module Description | 16 |
| 4.2 VGG | 17 |
| 4.3 Fine Tuning of Hyper-Parameter | 18 |
| 4.4 Epoch | 19 |
| 4.5 Batch Size | 19 |
| 4.6 Adam Optimizer | 19 |
|  |  |
| **CHAPTER 5** |  |
| **SYSTEM IMPLEMENTATION** | 21 |
| 5.1 Development of Transfer Learning Model | 21 |
| 5.2 Working Model Algorithm | 23 |
|  |  |
| **CHAPTER 6** |  |
| **RESULTS AND DISCUSSIONS** | 24 |
| 6.1 VGG19’s Performance Architecture | 24 |
| 6.2 Comparison between DenseNet121 and VGG19 | 25 |
| 6.3 DenseNet121 Performance | 26 |
|  |  |

|  |  |
| --- | --- |
| 6.4 VGG19 Performance | 26 |
| 6.5 Comparative Analysis | 26 |
| 6.6 Confusion Matrix | 28 |
| 6.7 Output | 29 |
|  |  |
| **CHAPTER 7** |  |
| **CONCLUSION AND FUTURE ENHANCEMENT** | 31 |
| 7.1 Conclusion | 31 |
| 7.2 Future Enhancement | 32 |
|  |  |
| **REFERENCES** | 33 |
| **APPENDIX** | 36 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| Fig. 1.1 | Working model of transfer learning | 8 |
| Fig. 4.1 | Block Diagram of Architectural design | 17 |
| Fig. 4.2 | Fine tuning of Hyper-Parameter | 18 |
| Fig. 5.1 | Flowchart of the model | 22 |
| Fig. 6.1 | Plot of accuracy and loss of DenseNet121 and VGG19 model | 27 |
| Fig. 6.2 | DenseNet121 – Confusion matrix | 28 |
| Fig. 6.3 | VGG19 – Confusion matrix | 28 |
|  |  |  |
|  |  |  |
|  |  |  |

**CHAPTER 1**

## INTRODUCTION

Skin cancer is one of the most prevalent forms of cancer worldwide, accounting for a significant portion of new cancer diagnoses each year. Early detection and accurate diagnosis are crucial for successful treatment and improved patient outcomes. However, the manual examination of skin lesions by dermatologists is subjective and time-consuming, often leading to delayed diagnosis and potential misdiagnosis.

In recent years, advancements in machine learning and computer vision have enabled the development of automated systems for skin cancer detection and classification. These systems leverage the power of deep learning algorithms to analyze dermatological images and assist healthcare professionals in making accurate diagnoses.

The aim of this project is to develop a robust and reliable system for the prediction of skin cancer using machine learning techniques. By harnessing the capabilities of deep learning models, such as convolutional neural networks (CNNs), the proposed system will analyze images of skin lesions and classify them into different categories, including benign lesions and various types of malignant skin cancers, such as melanoma, basal cell carcinoma, and squamous cell carcinoma.

The motivation behind this project stems from the need to improve the efficiency and accuracy of skin cancer diagnosis, particularly in regions where access to dermatologists may be limited. By providing a tool that can assist healthcare professionals in the early detection of skin cancer, this project aims to reduce the burden on healthcare systems, minimize unnecessary biopsies, and ultimately improve patient outcomes.

The project will utilize a large dataset of annotated dermatological images, comprising both benign and malignant lesions, to train and validate the deep learning models. Various pre-processing techniques will be employed to enhance the quality of the input images and improve the performance of the models. Additionally, transfer learning will be utilized to leverage pre-trained CNN models, such as VGG16, ResNet, and Inception, which have been trained on large-scale image datasets like ImageNet.

The performance of the developed system will be evaluated using metrics such as accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC). Extensive experimentation will be conducted to fine-tune the hyperparameters of the models and optimize their performance on the task of skin cancer prediction.

Overall, this project aims to contribute to the advancement of healthcare technology by providing a reliable and efficient tool for the early detection and classification of skin cancer. By leveraging the power of machine learning and computer vision, this system has the potential to revolutionize the way skin cancer is diagnosed and treated, ultimately saving lives and improving patient outcomes.

**1.1 Problem Statement**

The project aims to develop a machine learning model for accurately predicting skin cancer based on images of skin lesions. With skin cancer incidence rates on the rise, early detection is crucial for effective treatment and improved patient outcomes. Manual examination of skin lesions can be time-consuming and subjective, highlighting the need for automated systems to aid in diagnosis. Challenges include data variability, imbalanced datasets, identifying interpretable features, and ensuring model generalization. The proposed solution involves curating a comprehensive dataset, experimenting with various convolutional neural network architectures, training and evaluating the model, validating its performance, and visualizing model interpretations. The expected outcomes include accurate predictions, interpretability, model generalization, and potential impact on early detection efforts. Overall, the project aims to contribute to improving skin cancer diagnosis and patient outcomes through automated prediction systems..

**1.2 Need for Classification**

The need for classification in skin cancer prediction arises from several key factors:

1. Early Detection: Skin cancer, if detected early, is highly treatable. Classification models can aid in the early identification of suspicious skin lesions, allowing for timely intervention and potentially life-saving treatment.

2. Diagnostic Accuracy: Manual examination of skin lesions by dermatologists can be subjective and prone to errors. Classification models provide an objective and consistent method for analyzing skin lesions, potentially improving diagnostic accuracy.

3. Resource Optimization: Healthcare resources, including dermatologists' time and expertise, are limited. Classification models can serve as a screening tool, helping prioritize patients for further evaluation by dermatologists based on the likelihood of malignancy.

4. Accessibility: Access to dermatologists may be limited in certain regions or healthcare systems. Classification models can be deployed in various settings, including primary care clinics or even mobile applications, to provide accessible and cost-effective skin cancer screening.

5. Educational Support: Classification models can be used as educational tools to raise awareness about the warning signs of skin cancer and empower individuals to perform self-examinations. By providing information and guidance, these models contribute to early detection and prevention efforts.

Overall, the need for classification in skin cancer prediction stems from the desire to improve early detection, diagnostic accuracy, resource allocation, accessibility to healthcare services, and educational outreach in the fight against skin cancer.

**1.3 Applications of Skin Disease classification**

Skin disease classification has numerous applications across various domains, including healthcare, dermatology, research, and technology. These applications encompass early detection and diagnosis of various skin conditions such as cancerous lesions, dermatitis, psoriasis, eczema, and infections, leading to timely intervention and potentially saving lives. Furthermore, automated classification systems enable remote consultations and telemedicine services, particularly valuable in underserved areas. Dermatologists benefit from these systems as decision support tools, enhancing diagnostic accuracy, and efficiency. Additionally, skin disease classification technologies serve as educational tools for medical professionals and contribute to public health surveillance efforts by providing real-time data on disease prevalence and distribution. They also support clinical trials and research studies, facilitate personalized medicine approaches, and can be integrated into dermatological imaging devices for screening and diagnosis. Overall, skin disease classification plays a pivotal role in revolutionizing dermatological care, improving patient outcomes, and advancing medical research and technology.

**1.4 Existing System**

The existing systems for skin disease classification typically involve a combination of manual examination by dermatologists and computer-aided diagnosis (CAD) systems. Here's an overview of the components typically found in the existing systems:

1. Manual Examination by Dermatologists: Dermatologists visually inspect skin lesions using dermoscopy or clinical photography techniques. They rely on their expertise and experience to identify various characteristics of skin lesions, such as asymmetry, border irregularity, color variation, and diameter.

2. Dermoscopic Imaging: Dermoscopic imaging involves the use of a dermatoscope, a handheld device with a magnifying lens and light source, to capture high-resolution images of skin lesions. These images provide detailed information about the surface and subsurface structures of skin lesions, aiding in diagnosis.

3. Computer-Aided Diagnosis (CAD) Systems: CAD systems analyze dermoscopic images of skin lesions to provide automated assistance to dermatologists in diagnosis. These systems typically employ machine learning algorithms, particularly convolutional neural networks (CNNs), to classify skin lesions into different categories, such as melanoma, basal cell carcinoma, squamous cell carcinoma, or benign lesions.

4. Feature Extraction: CAD systems extract various features from dermoscopic images, such as color histograms, texture descriptors, shape characteristics, and morphological features. These features are used as input to machine learning algorithms for classification.

5. Machine Learning Algorithms: Machine learning algorithms, including supervised learning, unsupervised learning, and deep learning, are used to train models on labeled datasets of dermoscopic images. CNNs, in particular, have shown promising results in skin disease classification due to their ability to automatically learn discriminative features from raw image data.

6. Model Training and Validation: CAD systems train machine learning models on labeled datasets of dermoscopic images, using techniques such as cross-validation to evaluate model performance. Hyperparameter tuning and optimization are performed to improve model accuracy and generalization.

7. Diagnostic Assistance: CAD systems provide diagnostic assistance to dermatologists by automatically classifying skin lesions into different categories based on the analysis of dermoscopic images. Dermatologists can use the predictions generated by CAD systems as a reference to support their clinical decision-making process.

8. Limitations: Despite their potential benefits, existing CAD systems may still have limitations, including reliance on high-quality dermoscopic images, variability in lesion appearance, class imbalance in datasets, and the need for continuous validation and improvement to ensure accuracy and reliability.

In summary, existing systems for skin disease classification combine manual examination by dermatologists with computer-aided diagnosis systems that analyze dermoscopic images using machine learning algorithms. These systems aim to improve diagnostic accuracy and efficiency in the detection and classification of skin lesions, ultimately leading to better patient outcomes.

### 1.5 Proposed System

Building upon the existing machine learning model for ripeness classification, this project focuses on further development and refinement of the system in these key areas:

1. **Machine Learning Model Development**

* **Model Selection:** In the proposed system, we have chosen VGG19, a transfer learning model that has shown promise in image classification tasks. This choice is due to the architecture of VGG19, which combines small convolutional filters with several layers to capture complex characteristics in images efficiently.
* **Performance Evaluation:** Before implementing VGG19, thorough evaluation with the existing DenseNet121 model will be conducted.

1. **User Interface Development**

* The proposed system will have an easy-to-use interface built for easy interaction. No matter how technical user background is, users will find it simple to input tomato images and get accurate estimates of their level of ripeness.
* The interface will put an emphasis on ease of use and ability to make sure users can get around the system with ease. Clear directions and progress

indicators are examples of visualizations that will improve the user experience.

1. **Model Optimization and Retraining**

* To enhance the performance of the VGG19 model, optimization techniques like hyperparameter tuning will be applied. This process involves fine-tuning parameters such as learning rates and batch sizes to achieve the best possible results.
* Additionally, the model will undergo retraining using an expanded dataset. This dataset will incorporate new images of tomatoes across various ripeness stages, enabling the model to learn from a more diverse range of examples. Addressing class imbalances and ensuring dataset quality will be paramount during this phase.

By implementing these strategies, the proposed system aims to improve upon the existing Skin Disease classification framework. The adoption of VGG19 coupled with a user-friendly interface and optimization efforts.

A comprehensive comparison will be conducted between the performance of the DenseNet121 and VGG19 models in the task of Skin Disease classification. This comparative analysis will involve evaluating various metrics such as accuracy, precision, recall, and F1-score on both models using the same dataset and experimental setup. By directly comparing the results of DenseNet121 and VGG19, which model offers superior performance in accurately classifying tomatoes into matured, immature, or partially matured categories will be determined.

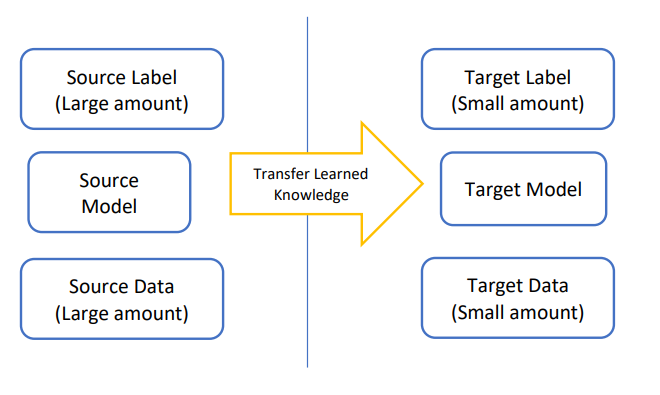
**1.6 Objective of the Proposed System**

The primary objective of this Skin Disease classification project is to develop an accurate system that can distinguish between tomatoes that are immature, partially mature, and mature at different ripeness stages. This project includes gathering different tomato images, preprocessing those images, and extracting specific characteristics from those images. Next, machine learning models will be developed with the help of convolutional neural networks (CNNs) are created and trained to classify the tomatoes according to their level of ripeness. Simple system interaction may be made

by the creation of user-friendly interfaces following a thorough assessment of the models' performance. By taking these aspects into account, the Skin Disease classification project ensures the development of an accurate, user-friendly system that can be adapted to a range of applications. It makes the processes of harvesting, storing, and distributing tomatoes more effective, which in turn improves product quality.

**1.7 Transfer Learning**

Transfer learning is a machine learning technique in which a pre-trained model created for one task is used as the foundation for another task that is related. In transfer learning, a pre-trained model is utilised as a starting point before being adjusted on a fresh dataset or task with comparatively fewer data points. Compared to training a model from scratch, this method can result in models that are trained faster and more accurately. A systematic representation of transfer learning is shown in fig. 1.1.



**Fig. 1.1 Working model of Transfer Learning**

**Types of Transfer Learning:**

**Inductive Transfer Learning:**

The pre-trained model is used as a starting point for a new task that is related but different from the original task. The pre-trained model is usually fine-tuned on the new task using less labelled data. This method is commonly used in computer vision and natural language processing (NLP) tasks.

**Transductive Transfer Learning:**

The pre-trained model is utilized as a starting point for a new task that has the

same input and output as the original task, but the data distribution is changed. This sort of transfer learning is known as transductive. Without altering the model's output, the pre-trained model is adjusted to the new data distribution.

**Unsupervised Transfer Learning:**

Similar to inductive transfer learning is unsupervised transfer learning. This algorithm use of unlabelled datasets in both the source and target tasks, as well as their focus on unsupervised tasks, is the only variation.

**CHAPTER 2**

**LITERATURE SURVEY**

## 1. Chinapat Sakunrasrisyuay et al, “Tomato Maturity Classification: A Transfer Learning Approach”, Conference: 2021 25th International Computer Science and Engineering Conference, Nov. 2021, doi: [10.1109/ICSEC53205.2021.968454](http://dx.doi.org/10.1109/ICSEC53205.2021.9684584).

Implementation of the Transfer Learning technique to classify the tomatoes based on their maturity state. The performance of tomato classification using various state-of-the-art CNNs with TL classifiers InceptionResnetV2, ResNet152V2, MobileNetV2 and AlexNet. Nearly 200 color images were been tested. Color richness was determined by maturity and rotten tomatoes were evaluated. Using tomato color as the primary characteristic, the image datasets were divided into two classes: mature and rotten. There were 233 training image datasets with 40 testing image datasets. Their comparative results showed that ResNet15V2 yielded the highest classification precision; with the best training accuracy of 92%, the best testing accuracy of 99.46% and the lowest training loss, compared to the other CNN-based models.

## 2. Haichun Zuo, “Analysis and Detection of Tomatoes Quality Using Machine Learning Algorithm and Image Processing”, Nanfang College Guangzhou, Sep. 2022, doi: 10.21203/rs.3.rs-2016895/v1.

A comprehensive overview of the machine learning models in the image processing is provided. An in-depth overview of various methods such as preprocessing, segmentation, feature extraction, and classification that focus on fruit and vegetable quality based on color, texture, size, shape, and defects were provided. Using the appearance characteristics of tomatoes such as color, shape, size, and texture, image processing operations have been performed with MATLAB software to extract the mentioned features. In the future, a system can be set up that reads images on the rails simultaneously and processes them in video images. Algorithms such as deep learning are introduced that can be more efficient in real-time processing. Thus, the study method can be developed for future work with deep learning algorithms.

## 3. Ninja Begum, Manuj Kumar Hazarika, “Maturity Detection of Tomatoes Using Transfer Learning”, in: ScienceDirect Measurement: Food 7 (2002) 100038.

Deep transfer learning which is a sub category of artificial intelligence had been applied to acquire good accuracy. Here transfer learning is used to classify tomatoes into their maturity classes. Three approaches of transfer learning viz. VGG, Inception and ResNet are implemented. Tomato dataset is used during the experiment which consisted of all the three classes of tomatoes. The models were evaluated by training them iteratively with varying epoch number and batch size. From the results it is seen that VGG 19 performed best at epoch 50 and batch size 32. The other models also showed good results proving transfer learning to be a viable solution in solving food related problems.

## 4. Prasenjit Das et al, “An Automated Tomato Maturity Grading System Using Transfer Learning based AlexNet”, Article  in  Ingénierie des systèmes d information · April 2021, doi: : 10.18280/isi.260206.

A Transfer Learning approach is introduced and utilized it to develop an automated Tomato Maturity Grading system. the proposed method used a well-performed pretrained CNN model, namely AlexNet, to take advantage of the Transfer Learning. Transfer learning applied at the last three layers to customize the AlexNet as per our proposed method's requirement. Hence Alexnet is customized with a fully connected layer of three neurons, a softmax layer, and a classification layer of only three classes to classify tomato images into three distinct maturity stages: Red, Yellow, and Green, based on their color feature analysis. The dataset was prepared using the augmentation process applied on the camera captured tomato images from a local market. The augmentation process increased the number of images in the dataset that prevented the model from overfitting issues during training with fewer image samples captured. The experimental results showed that the proposed method achieved promising results with an overall accuracy of 94%. Their proposed method has less computation of parameters and better precision as compared to other available standard architecture.

**5. Taehyeong Kim et al, “Tomato Maturity Estimation Using Deep Neural Network”, Special Issue Data-Driven Agricultural Innovation Technology for Digital Architecture, Dec. 2022, doi: 10.3390/app13010412.**

A tomato maturity estimation approach based on a deep neural network has been proposed. Tomato images were obtained using an RGB camera installed on a monitoring robot and samples were cropped to generate a dataset with which to train the classification model. The classification model is trained using cross-entropy loss and mean-variance loss, which can implicitly provide label distribution knowledge. For continuous maturity estimation in the test stage, the output probability distribution of four maturity classes is calculated as an expected value. Their results demonstrate that F1 score was approximately 0.91 on average, From the overall results we found that their approach cannot only classify the discrete maturation stages of tomatoes but can also continuously estimate their maturity. Furthermore, it is expected that with higher accuracy data labelling, more precise classification and higher accuracy may be achieved.

**6.** **Mayuri Sharma et al, “Ensemble averaging of Transfer Learning models for identification of nutritional deficiency in Rice Plant”   
Department of Computer Science, The University of Missouri, St. Louis, MO 63121, USA**

A framework for hosting sophisticated systems on the cloud, where processing may take place and farmers can communicate with the system, is proposed. In order to carry out the duty of deficiency detection in rice plants, six TL architectures namely, Inceptionv3, Resnet152v2, Exception, Densenet201, Inceptionresnetv2, and VGG19 are taken into consideration in this study. The two publicly accessible datasets from Mendeley and Kaggle were used. In the Mendeley dataset, the ensemble-based architecture increased the maximum classification accuracy to 95% from 91.17%, while in the Kaggle dataset, it increased to 92% from 90%.

**7. Thani Jintasuttisak, “Deep neural network-based date palm tree detection in drone imagery”, Computers and Electronics in Agriculture Volume 192, Jan 2022, 106560, ScienceDirect.**

The demonstration of the successful application of a cutting-edge CNN, YOLO-V5, in recognizing date palm trees in photos acquired by a camera aboard a drone flying 122 meters over farmlands in the Northern Emirates of the United Arab Emirates (UAE) was done. During the dataset preparation procedure, they randomly picked 125 recorded photos and separated them into three datasets: training (60%), validation (20%), and testing (20%).

**8. Vrunda Kusanur, “Using Transfer Learning for nutrient deficiency prediction and classification in Tomato** **Plant**”, **International Journal of Advanced Computer Science and Applications(IJACSA), Volume 12 Issue 10, 2021.**

They employed the Transfer Learning model, a component of a pre-trained deep learning model, to identify nutrient stress in plants. To increase classification accuracy, they evaluated three alternative architectures, including Inception-V3, ResNet50, and VGG16, with two classifiers: RF and SVM.

**9.** [**Zahraa Al Sahili**](https://arxiv.org/search/cs?searchtype=author&query=Sahili,+Z+A)**,**[**Mariette Awad**](https://arxiv.org/search/cs?searchtype=author&query=Awad,+M)**, “The Power of Transfer Learning in Agricultural Applications” Maroun Semaan Faculty of Engineering, American University of Beirut, Beirut, Lebanon.**

AgriNet is constructed in this research with a dataset collection of 160k agricultural photographs from over 19 geographical regions, multiple image captioning devices, and over 423 plant types and illnesses. AgriNet model is a collection of pretrained models based on five ImageNet architectures: VGG16, VGG19, Inception-v3, InceptionResNet-v2, and Xception. AgriNet-VGG19 had the best classification accuracy (94%), as well as the highest F1-score (92%). Moreover, all suggested models were found to successfully categorize the 423 classes of plant species, diseases, pests, and weeds, with the Inception-v3 model having a minimum accuracy of 87%. Lastly, tests were carried out on two external datasets to assess the superiority of AgriNet models over ImageNet models: a pest and plant diseases dataset from Bangladesh and a plant diseases dataset from Kashmir.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1. General Project Information** | | | | | |
| **Project Name:** | | **A DEEP LEARNING APPROACH FOR SKIN DISEASE CLASSIFICATION** | | | |
| **Executive Sponsors:** | | **-** | | | |
| **Department Sponsor:** | | **-** | | | |
| **Impact of project:** | | Using Transfer Learning to predict the ripeness of the tomato. | | | |
| **2. Project Team** | | | | | |
|  | **Name** | | **Department** | **Telephone** | **E-mail** |
| **Project Mentor:** | **Dr. R. Nagendran** | | IT | 9486087812 | nagendran.it@srit.org |
| **Team Members:** | **Gopinath T** | | IT | 6379568342 | gopinath.2006@srit.org |
|  | **Inbhatamil S** | | IT | 9444762600 | inbhatamil.2006@srit.org |
|  | **Prabakaran C** | | IT | 6382190286 | prabakaran.2006@srit.org |
| **3. Stakeholders** | | | | | |  |
|  | | | | | |
| **4. Project Scope Statement** | | | | | |
| **Project Purpose / Business Justification** | | | | | |
| To classify tomatoes based on their maturity state.  To minimize the tomato loss due to its unripe condition. | | | | | |
| **Objectives (in business terms)** | | | | | |
| **Deliverables** | | | | | |
| The model that classifies the tomatoes as ripen and unripe. | | | | | |
| **Scope:** | | | | | |
| Scope of this project is to classify the tomatoes based on their maturity state as matured, immature and partially matured. | | | | | |

**PROJECT CHARTER**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Milestones** | | | | | | | | |
|  | **Phases** | | | **Start date (dd-mm-yyyy)** | | **End date (dd-mm-yyyy)** | |  |
|  | Model development | | | 26-12-2023 | | 10-01-2024 | |  |
|  | Implementation and parameter-tuning | | | 11-01-2024 | | 24-01-2024 | |  |
|  | Development of UI | | | 26-01-2024 | | 20-02-2024 | |  |
|  | Implementation | | | 21-02-2024 | | 15-03-2024 | |  |
|  | Testing and Documentation | | | 16-03-2024 | | 23-03-2024 | |  |
| **Major Known Risks (including significant Assumptions)** | | | | | | | | |
|  | **Risk** | | | | | **Risk Rating (High, Medium, Low)** | |  |
|  | We have difficulties in learning new concept like Machine learning so this maybe lead us to schedule overruns | | | | | Medium | |  |
| **Constraints** | | | | | | | | |
|  | Time | 3 months | | | | | |  |
|  | Budget |  | | | | | |  |
|  | Quality | The project will be of high quality with assured data accuracy. | | | | | |  |
|  | Scope | The scope shall be defined previously in this document section titled ‘scope’ and no more | | | | | |  |
| **5. Communication Strategy** | | | | | | | | |
| Meetings have been conducted  Communicating through mobile phones. | | | | | | | | |
| **6. Sign-off** | | | | | | | | |
|  | | | **Name** | | **Signature** | | **Date**  **(DD/MM/YYYY)** | |
| **Executive Sponsor** | | |  | |  | |  | |
| **Department Sponsor** | | |  | |  | |  | |
| **Project Mentor** | | | **Dr. R. Nagendran** | |  | |  | |

**CHAPTER 3**

## REQUIREMENT SPECIFICATIONS

### 3.1 Software Requirements

|  |  |  |
| --- | --- | --- |
| Operating System | : | Windows 10 & above |
| Simulator Tool | : | Jupyter notebook, Visual Studio Code |
| Programming Package | : | Python 3 |

* 1. **Hardware Requirements**

|  |  |  |
| --- | --- | --- |
| Processor | : | Any Intel or AMD x84-64 processor |
| RAM | : | Minimum 4 GB |
| Hard Disk | : | 24 GB to accommodate the project files, datasets, and software tools |
| Input Device | : | Standard Keyboard and Mouse |
| Output Device | : | Standard Monitor |

**3.3 System Tools**

Using the Jupyter Notebook, an open-source web application, documents with text, equations, live code, and visualizations can be created. The Jupyter Notebook team is in charge of maintaining Jupyter Notebook.

The IPython project, which formerly had its own IPython Notebook project, gave rise to the Jupyter Notebooks project. The core programming languages that are supported by Jupyter are: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that can also be used.

**CHAPTER 4**

**METHODOLOGY**

**4.1 Module Description**

On considering the proposed scheme, the project is divided into the following modules.

1. Development of transfer learning-based model to classify the different maturity stages of the tomato.
2. Classification of different maturity stages of tomato

**4.1.1. Development of Transfer Learning-Based Model to Classify the Different Maturity Stages of the Tomato**

For training the model the various images of tomato have been collected from internet and these images are done some data preprocessing technique such as image cropping, resizing, noise removal and image enhancement to train the model.

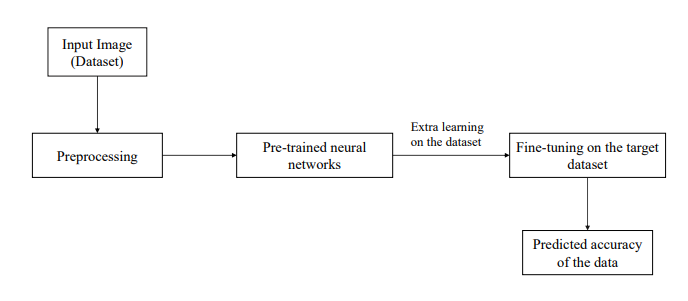
The dataset is labelled based on the maturity stages of the tomato. The dataset has been split into three groups for the purpose of training, validating, and testing the data.

* Training (75%)
* Validation (20%)
* Testing (5%)

Feature engineering concept is used where the raw data is processed and transformed into features based on RGB colors and a predictive model is build using machine learning and a statistical modelling architecture. For the model building, the VGG19 Architecture is used, and it is also compared with the other model DenseNet121.

**4.1.2 Classification of Maturity Stage of Tomato**

In order to achieve better accuracy, the model is trained to classify the maturitystate further in the level of immature, partially matured and matured. For each state immature, partially matured, and matured, the dataset images are labelled respectively. Therefore, by using this strategy to train the model, high levels of accuracy can be obtained.



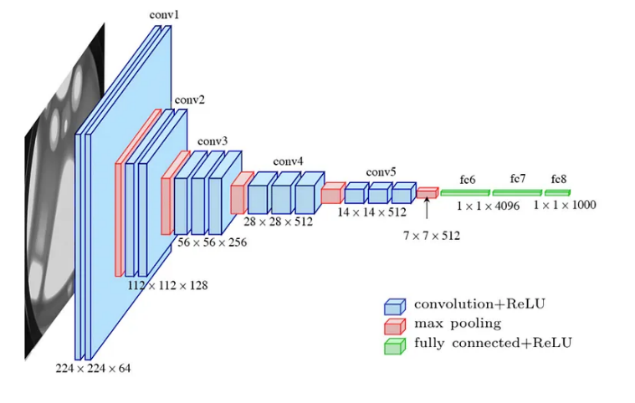
**Fig. 4.1 Block Diagram of Architectural Design**

**4.2 VGG**

The VGG – (Visual Geometry Group) is one among the standard deep Convolutional Neural Network (CNN) architecture with several layers. VGG 16 or VGG 19 describes the number of convolutional layers i.e, 16 and 19 convolutional layers respectively.

VGG19 is a convolutional neural network architecture known for its deep structure, comprising 19 layers, including convolutional and pooling layers followed by fully connected layers. Its design simplicity, characterized by stacking multiple layers with small receptive fields, has made it a popular choice in various computer vision applications, including image classification.

VGG19 is used in the project to categorize images into many classes or categories according to their visual characteristics. The architecture is particularly suitable for this task since it can capture complex patterns and hierarchical representations inside images. VGG19 gradually learns to represent complicated visual information by extracting features at various levels of abstraction through the processing of images through its convolutional layers.



**Fig 4.2 VGG Net Architecture**

In conclusion, VGG19 is essential to the project's image classification

processing since it can reliably classify images by using its deep architecture and hierarchical feature extraction skills. Even though it performed well overall, more research and adjustment might be required to maximize its effectiveness in various training setups and guarantee reliable outcomes.

**4.3 Fine Tuning of Hyper-Parameter**

The working model's learning process is controlled by a set of parameters known as the hyper-parameters. To fix the hyperparameter, both the working model and working dataset are needed. Every model looks for the optimal combination of hyper-parameters. Therefore, hyper-parameters like the number of epochs and the batch size are adjusted across the training set in order to maximize the model's classification performance during the experiment. The hyper-parameter values that produce the maximum classification accuracy determine how effective the model is. An epoch is the total number of training iterations across the whole dataset. By determining the optimal collection of hyper-parameters, we are able to maximize the performance of the architectures.

### Additionally, an image data generator is used to train the model's input images in order to minimize overfitting and underfitting problems. In the model mentioned above, the image data generator flips, zooms, and rescales the given input images. Once more, the loss function "categorical cross entropy" and the optimizer "adam" are used to train the models. The default value of 0.001 is set by the Keras library for the learning rate. It's important to keep in mind that the model's accuracy was the only reason these hyper-parameters were set during training.

**4.4 Epoch**

An epoch refers to a single flow or pass through the entire training dataset during the training the phase of a model. During each epoch, the model iterates through the entire dataset, processing each sample to make predictions and updates its parameters. The number of epochs is a hyperparameter that defines how many times the model will cycle through the entire dataset. Choosing the appropriate number of epochs is crucial for achieving good performance without overfitting or underfitting the data.

**4.5 Batch Size**

When training a model, the number of training samples processed in a single iteration is referred to as the batch size in machine learning. While bigger batch sizes improve computational efficiency, they may slow down convergence and impact generalization. Smaller batch sizes allow more frequent but noisier updates to model parameters, potentially aiding in convergence. For best results, the batch size selection must be carefully considered and empirically experimented with, since it affects both training dynamics and computational efficiency.

**4.6 Adam Optimizer**

The Adam optimizer, short for Adaptive Moment Estimation, is a widely employed optimization algorithm in the deep learning algorithms. It merges attributes from two other optimization methods, namely RMSprop and Momentum, to dynamically fine-tune learning rates throughout the training process. Adam's flexibility and efficiency

in adjusting to varying data conditions have propelled its popularity, establishing it as a preferred choice for optimizing deep neural networks, showcasing robust convergence and superior performance across diverse tasks.

Setting the learning rate to 0.001 implies that the model parameters are updated in small increments during each training step, which can help stabilize the training process and prevent overshooting the optimal solution.

**CHAPTER 5**

**SYSTEM IMPLEMENTATION**

**5.1 Development of Transfer Learning based model**

A number of images of tomatoes had been taken and gathered from the web. The images have been classified as mature, immature, or partially matured based on the color of the various maturity stags. Table 5.1 illustrates that images of ripe tomatoes are reddish, those of partially matured tomatoes are pale yellowish-green, and those of immature tomatoes are green. The images are split in the ratio 75:20:5 for the model's train, test, and validation stages. Here, the Densenet121 has been used to forecast maturity state.

|  |  |
| --- | --- |
| Maturity Stage | Color of the Tomato/ Image |
| Matured | Red |
| Immature | Green |
| Partially Matured | Pale yellowish-green |

**Table 5.1 Color of the tomato at different maturity stages**

Following the image collection, it undergoes a number of phases, as seen in fig. 5.1. The images in the dataset have been pre-processed by certain techniques such as cropping, enhancement and resizing take place and then followed by data augmentation process. The dataset can be amplified by making small adjustments to the data or by creating new data points using machine learning models to explore the latent space of the original data. In the ratio of 75:20:5, the data is then classified into the following groups: training data, validation data, and testing data. Densenet121 model architectures is used to carry out the transfer learning. The model is then assessed using performance indicators including the F1 score, recall, accuracy, confusion matrix, and precision.

**Image**

**acquisition**

**Image**

**processing**

**Image**

**cropping**

**Image**

**enhancement**

**Image**

**resizing**

**Data**

**augmentation**

**Testing**

**data**

**Transfer**

**Learning**

**Validation**

**data**

**Data**

**splitting**

**Training**

**data**

**Preference**

**evaluation**

**Fig. 5.1 Flowchart of the Model**

**5.2 Working Model Algorithm**

Step 1: Collect a dataset of tomato images with corresponding maturity classes.

Step 2: Split the dataset into training, validation, and testing sets in the ratio of 75:20:5. Step 3: Normalize the pixel values of the grayscale images to a range between 0 and 1. Step 4: To make the dataset more variable, use data augmentation techniques like rotation,

scaling and flipping.

Step 5: Use a pretrained CNN model, VGG19 with multiple convolutional layers

followed by pooling layers to extract the features from the greyscale images.

Step 6: Add several fully connected layers at the end to map the extracted features to the

maturity content labels.

Step 7: To determine class probabilities, use a Softmax activation function in the last layer. Step 8: Define a loss function such as categorical cross-entropy to measure the difference

between the predicted and actual maturity content labels.

Step 9: Use an optimization algorithm such as Adam to minimize the loss function and

update the model parameters.

Step 10: Train the model on the training set for a specified number of epochs, adjusting

the learning rate and other hyperparameters as needed.

Step 11: Monitor the model's performance on the validation set to detect overfitting.

Step 12: Test the model on the testing set and compute performance metrics such as

accuracy, precision, recall, and F1 score.

Step 13: DenseNet121 and VGG19 models metrices are compared.

Step 13: Use the best model for further prediction of maturity stage of the tomato.

**CHAPTER 6**

## RESULT AND DISCUSSION

**6.1. VGG19 Model’s Performance Metrices**

VGG19 model yields an accuracy of about 94% with the least amount of loss at a learning rate of 0.001 value furthermore epoch of 25 and batch size of 32.

The performance metrics such as precision, recall, accuracy and f1 score of the model are calculated using the equations described below

Precision = TP

(TP + FP)

Recall = TP

(TP+FN)

F1 score = 2 ∗ (precision ∗ recall)

Precision + recall

Accuracy = TP+TN

TP+TN+FP+FN

where TP -> number of true positive instances,

TN -> number of true negative instances,

FP -> number of false positive instances,

FN -> number of false negative instances

**6.3 DenseNet121 Performance**

Across various epochs and batch sizes, DenseNet121 consistently demonstrated robust performance. At epoch 25 with a batch size of 64, DenseNet121 achieved remarkable precision, recall, F1-score, and accuracy, with values of 1.00, 0.92, 0.96, and 87%, respectively. This suggests that DenseNet121 effectively minimized false positives while maximizing true positives, resulting in high overall classification accuracy.

The achieved F1-score of 0.96 indicates a balanced trade-off between precision and recall, which is essential for handling imbalanced datasets commonly encountered in real-world applications.

**6.4 VGG19 Performance**

Although VGG19 also demonstrated competitive performance, compared to DenseNet121, its outcomes shown more variation across various configurations. VGG19 obtained excellent recall, accuracy, F1-score, and precision values of 0.9, 1.00, 0.85, and 94%, respectively, at epoch 25 with a batch size of 32.

The difference implies that although VGG19 performed exceptionally well at reducing false positives, it might have had difficulties maximizing actual positives, which resulted in a little lower F1-score.

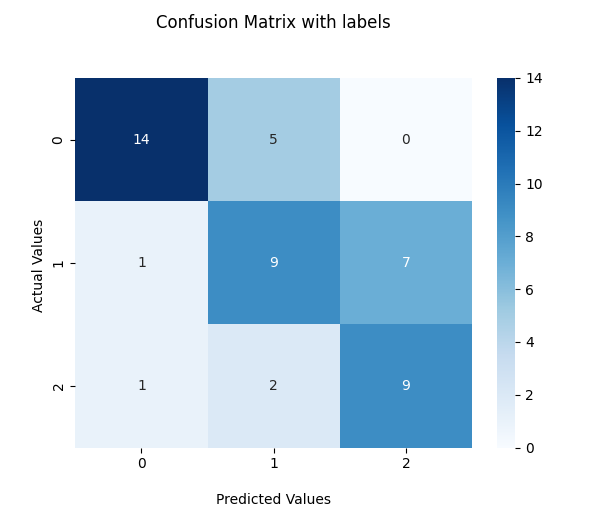
**6.5 Comparative Analysis**

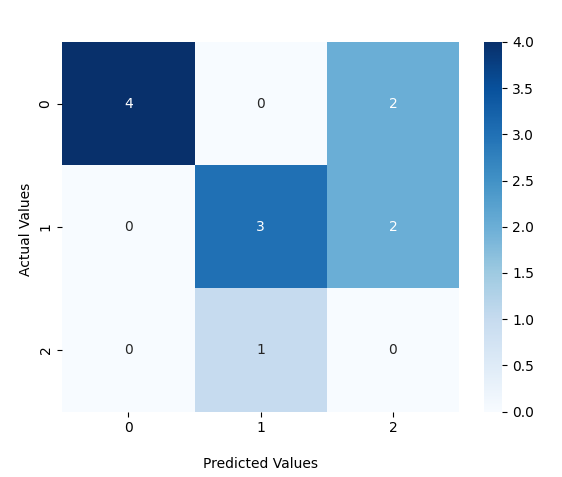
It is clear from comparing DenseNet121 and VGG19's performance that both architectures can achieve high accuracy when it comes to image classification tasks. DenseNet121, on the other hand, performed more consistently across configurations, performing especially well in situations involving larger batch sizes and longer training epochs. With its consistent performance, DenseNet121 would be an excellent option for applications involving complicated feature handling and strong modification.

On the other hand, VGG19's performance, while still impressive, exhibited more sensitivity to variations in training parameters. This variation could be explained by the deeper architecture of VGG19, which under some circumstances may make it more prone to overfitting or divergence issues.

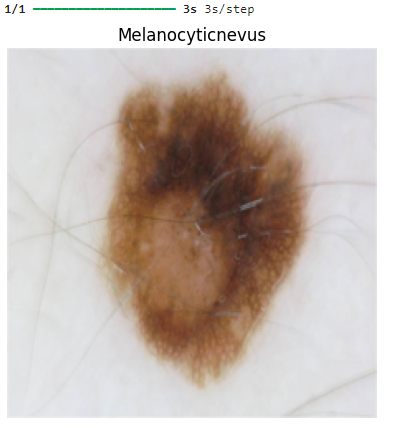
In summary, our results highlight the effectiveness of DenseNet121 and VGG19 in image classification tasks. DenseNet121, with its densely connected architecture, demonstrated consistent and reliable performance across different training configurations. VGG19, while competitive, exhibited more variability in performance, suggesting potential sensitivity to training parameters.

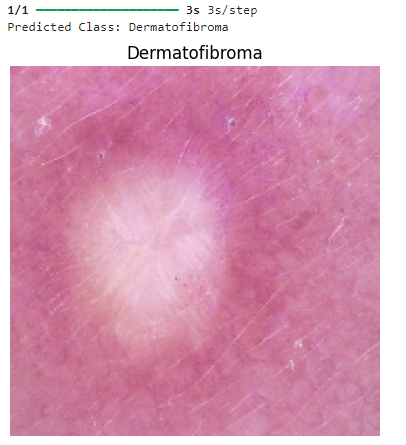
**6.6 Confusion Matrix**





**6.7 Output:**

****

****

****

**CHAPTER 7**

## CONCLUSION

**7.1 Conclusion**

This study developed an automated tomato maturity grading system by introducing and utilizing the Transfer Learning technique. The recommended method made use of Transfer Learning by utilizing DenseNet, and VGG, a successful pretrained CNN model. The tomato images that were collected from the internet were subjected to the augmentation method in order to prepare the dataset. The augmentation process helped the model avoid overfitting issues that arise when training with fewer image samples by increasing the number of images in the dataset. The method can accurately determine the tomato images' stages of maturity.

DenseNet121 consistently demonstrated robust performance, particularly with larger batch sizes, showcasing its effectiveness in learning complex patterns within the image data. Its dense connectivity pattern and efficient feature reuse likely contributed to its superior performance compared to VGG19 in most configurations.

On the other hand, VGG19 also performed flawlessly, although with some variability across different epochs and batch sizes. Its deep architecture allowed it to capture complex features within the images, however with slightly less consistency compared to DenseNet121.

Overall, this project underscores the importance of systematically evaluating different architectures and optimization strategies to identify the most suitable approach for achieving desired performance goals in image classification tasks.

**7.2 Future Enhancement**

The Deep Learning model generates a lot of calculations, the model performs a little slowly. In the future, a lightweight Deep Learning model that has been specially trained to categorize stages will be implemented from scratch. An application will be developed so that the user can take the live photos and the model predicts it’s maturity.

## REFERENCES

[1] Ninja Begum, Manuj Kumar Hazarika, “Maturity detection of tomatoes using transfer learning”, Measurement: Food 7 (2022) 100038, ScienceDirect.

[2] T. Wang, J. Huan, M. Zhu, “Instance-based deep transfer learning”, in: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2019, pp. 367–375.

[3] K. Prasad, S. Jacob, M.W. Siddiqui, “Fruit maturity, harvesting, and quality standards, in: Preharvest Modulation of Postharvest Fruit and Vegetable Quality”, Academic Press, 2018, pp. 41–69.

[4] S. Dargan, M. Kumar, M.R. Ayyagari, G. Kumar, “A survey of deep learning and its applications: a new paradigm to machine learning”, Arch. Comput. Meth. Eng. 27 (4) (2020) 1071–1092.

[5] I.N.C.E. Ahmet, M.Y. Çevik, K.K. Vursavuş, “Effects of maturity stages on textural mechanical properties of tomato”, Int. J. Agric. Biol. Eng. 9 (6) (2016) 200–206.

[6] P. Wan, A. Toudeshki, H. Tan, R. Ehsani, “A methodology for fresh tomato maturity detection using computer vision”, Comput. Electron. Agric. 146 (2018) 43–50.

[7] A. Rafiq, H.A. Makroo, M.K. Hazarika, “Artificial neural network-based image analysis for evaluation of quality attributes of agricultural produce”, J. Food Process. Preserv. 40 (5) (2016) 1010–1019.

[8] N. El-Bendary, E. El Hariri, A.E. Hassanien, A. Badr, “Using machine learning techniques for evaluating Skin Disease”, Expert Syst. Appl. 42 (4) (2015) 1892–1905.

[9] J. O’Donoghue, M. Roantree, “A framework for selecting deep learning hyper-parameters”, in: British International Conference on Databases, Springer, Cham, 2015, pp. 120–132.

[10] R. Poojary, R. Raina, A.K. Mondal, “Effect of data-augmentation on fine-tuned CNN model performance”, IAES Int. J. Artific. Intell. 10 (1) (2021) 84.

[11] Simonyan, K., & Zisserman, A. (2014). “Very deep convolutional networks for largescale image recognition”. arXiv preprint arXiv:1409.1556.

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, “Rethinking the inception architecture for computer vision”, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.

[13] K. He, X. Zhang, S. Ren, J. Sun, “Deep residual learning for image recognition”, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[14] Y. Altuntaş, Z. Cömert, A.F. Kocamaz, “Identification of haploid and diploid maize seeds using convolutional neural networks and a transfer learning approach”, Comput. Electron. Agric. 163 (2019) 104874.

[15] G. Vrbančič, V. Podgorelec, “Transfer learning with adaptive fine-tuning”, IEEE Access 8 (2020) 196187-196211

[16] H.J.G. Opeña, J.P.T. Yusiong, “Automated tomato maturity grading using ABC– trained artificial neural networks”, Malaysian J. Comput. Sci. 30 (1) (2017) 12–26.

[17] D.S. Prabha, J.S. Kumar, “Assessment of banana fruit maturity by image processing technique”, J. Food Sci. Technol. 52 (3) (2015) 1316–1327.

[18] A. Noor, Y. Zhao, A. Koubâa, L. Wu, R. Khan, F.Y. Abdalla, “Automated sheep facial expression classification using deep transfer learning”, Comput. Electron. Agric. 175 (2020) 105528.

[19] J. Chen, J. Chen, D. Zhang, Y. Sun, Y.A. Nanehkaran,” Using deep transfer learning for image-based plant disease identification”, Comput. Electron. Agric. 173 (2020) 105393.

[20] Haichun Zuo, “Analysis and Detection of Tomatoes Quality using Machine Learning Algorithm and Image Processing”, Sptember 7 th 2022, Research Square.

**APPENDIX**

**SOURCE CODE**

**1.Importing Libraries**

import os

import pathlib

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

import shutil

import random

import tensorflow as tf

from tensorflow import keras

import tensorflow\_hub as hub

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout, BatchNormalization, GlobalAveragePooling2D, Activation, GlobalMaxPool2D, BatchNormalization

from tensorflow.keras.optimizers import Adam

from tensorflow.keras import Input, Model

from tensorflow.keras.utils import plot\_model

from sklearn.metrics import classification\_report, confusion\_matrix, ConfusionMatrixDisplay

**2.Datapreprocessing**

splitfolders.ratio(r"C:/Users/ajayk/Desktop/skincancer/databse", output="output",

seed=1337, ratio=(.75, .2, .05), group\_prefix=None, move=False)

**3.Exploratory Data Analysis**

data\_dir = pathlib.Path('output/')

train\_dir = 'output/train'

val\_dir = 'output/val'

test\_dir = 'output/test'

class\_name = np.array(sorted([item.name for item in data\_dir.glob("\*")]))

print(class\_name)

for dirpath, dirnames, filenames in os.walk('output'):

print(f"{len(dirnames)} folder and {len(filenames)} images in {dirpath}")

def view\_random\_image(target\_dir, target\_class):

target\_folder = target\_dir + target\_class

random\_image = random.sample(os.listdir(target\_folder), 1)

img = mpimg.imread(target\_folder + "/" + random\_image[0])

plt.imshow(img)

plt.title(target\_class)

plt.axis("off")

print(f"Image Shape : {img.shape}")

plt.show()

for i in range(1,2):

img\_n = view\_random\_image(target\_dir='output/train/', target\_class= Melanocyticnevus)

for i in range(1,2):

img\_n = view\_random\_image(target\_dir='output/train/', target\_class= Dermatofibroma)

for i in range(1,2):

img\_n = view\_random\_image(target\_dir='output/train/', target\_class=’ Melanoma’)

**Feature Engineering**

tf.random.set\_seed(46)

train\_datagen = ImageDataGenerator(rescale=1/255.0,

rotation\_range=0.2,

zoom\_range=0.2,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

vertical\_flip=True,

horizontal\_flip=True)

valid\_datagen = ImageDataGenerator(rescale=1/255.0)

test\_datagen = ImageDataGenerator(rescale=1/255.0)

**Pipeline**

train\_data = train\_datagen.flow\_from\_directory(train\_dir,

batch\_size=16,

target\_size=(224, 224),

class\_mode='categorical',

shuffle=True,

seed=46)

valid\_data = valid\_datagen.flow\_from\_directory(val\_dir,

batch\_size=16,

target\_size=(224, 224),

class\_mode='categorical',

shuffle=False,

seed=46)

test\_data = test\_datagen.flow\_from\_directory(test\_dir,

batch\_size=16,

target\_size=(224, 224),

class\_mode='categorical',

shuffle=False,

seed=46)

train\_data.class\_indices

train\_y=train\_data.classes

val\_y=valid\_data.classes

test\_y=test\_data.classes

print("train\_y.shape: ", train\_y.shape)

print("val\_y.shape: ", val\_y.shape)

print("test\_y.shape: ", test\_y.shape)

**Modelling**

base\_model = tf.keras.applications.DenseNet121(input\_shape=(224, 224, 3),

include\_top=False, weights='imagenet')

base\_model.trainable = False

global\_average\_layer = GlobalAveragePooling2D()

prediction\_layer = Dense(3)

softmax = Activation('softmax')

inputs = Input(shape=(224, 224, 3))

x = base\_model(inputs, training=False)

x = Dropout(0.25)(x)

x = global\_average\_layer(x)

outputs = prediction\_layer(x)

outputs = softmax(outputs)

model = Model(inputs, outputs)

model.summary()

plot\_model(model, show\_shapes=True)

model.compile(loss='categorical\_crossentropy',

optimizer=Adam(learning\_rate=0.001),

metrics=['accuracy'])

history = model.fit(train\_data, epochs=25, validation\_data=valid\_data)

**Model Evaluation**

def plot\_loss\_curves(history):

loss = history.history['loss']

val\_loss = history.history['val\_loss']

accuracy = history.history['accuracy']

val\_accuracy = history.history['val\_accuracy']

epochs = range(len(loss))

plt.plot(epochs, loss, label='training\_loss')

plt.plot(epochs, val\_loss, label='val\_loss')

plt.title("Loss Curves")

plt.xlabel("Epochs")

plt.legend()

plt.figure()

plt.plot(epochs, accuracy, label='training\_accuracy')

plt.plot(epochs, val\_accuracy, label='val\_accuracy')

plt.title("Accuracy Curves")

plt.xlabel("Epochs")

plt.legend()

plot\_loss\_curves(history)

val\_pred = model.predict(valid\_data)

val\_pred = val\_pred.argmax(axis=1)

print(classification\_report(val\_pred, val\_y))

cfm\_val = confusion\_matrix(val\_pred, val\_y)

ax = sns.heatmap(cfm\_val, annot=True, cmap='Blues')

ax.set\_title('Confusion Matrix with labels\n\n');

ax.set\_xlabel('\nPredicted Values')

ax.set\_ylabel('Actual Values ');

plt.show()

**Model Inference**

test\_pred = model.predict(test\_data)

test\_pred = test\_pred.argmax(axis=1)

print(classification\_report(test\_pred, test\_y))

cfm\_test = confusion\_matrix(test\_pred, test\_y)

ax = sns.heatmap(cfm\_test, annot=True, cmap='Blues')

ax.set\_title('Confusion Matrix with labels\n\n');

ax.set\_xlabel('\nPredicted Values')

ax.set\_ylabel('Actual Values ');

plt.show()

**Model Saving**

model.save('model\_densenet\_121.h5')

from keras.models import load\_model

from tensorflow.keras.preprocessing import image

from keras.applications.vgg16 import preprocess\_input

from keras.applications.vgg16 import decode\_predictions

from keras.applications.vgg16 import VGG16

import matplotlib.pyplot as plt

from keras.models import load\_model

from tensorflow.keras.preprocessing import image

import numpy as np

# Load the trained model

model = load\_model('my\_model.keras')

# Load the image

image\_path = r"C:\Users\ajayk\Desktop\skincancer\databse\Dermatofibroma\ISIC\_0024386.jpg"

image = image.load\_img(image\_path, target\_size=(224, 224))

# Preprocess the image

img = np.array(image)

img = img / 255.0

img = img.reshape(1, 224, 224, 3)

# Predict the label

label = model.predict(img)

# Determine the predicted class

class\_names = ['Dermatofibroma', 'Melanocyticnevus', 'Melanoma']

predicted\_class\_index = np.argmax(label)

predicted\_class = class\_names[predicted\_class\_index]

# Print the predicted class

print("Predicted Class:", predicted\_class)

# Display the image with the predicted label

plt.imshow(image)

plt.title(predicted\_class)

plt.axis('off')

plt.show()

**Tinter Code for GUI**

import tkinter as tk

from tkinter import filedialog

from PIL import Image, ImageTk

from keras.models import load\_model

from tensorflow.keras.preprocessing import image

import numpy as np

import matplotlib.pyplot as plt

# Function to classify an image

def classify\_image():

# Load the trained model

model = load\_model('my\_model.keras')

# Open a dialog box to select an image file

file\_path = filedialog.askopenfilename()

# Load the selected image

image = Image.open(file\_path)

image = image.resize((224, 224)) # Resize the image to match model input size

# Convert image to numpy array and preprocess

img = np.array(image)

img = img / 255.0

img = img.reshape(1, 224, 224, 3)

# Predict the label

label = model.predict(img)

# Determine the predicted class

class\_names = ['Dermatofibroma', 'Melanocyticnevus', 'Melanoma']

predicted\_class\_index = np.argmax(label)

predicted\_class = class\_names[predicted\_class\_index]

# Display the image with the predicted label

plt.imshow(image)

plt.title(predicted\_class)

plt.axis('off')

plt.show()

# Create a Tkinter window

root = tk.Tk()

root.title("Skin Cancer Classifier")

# Create a button to trigger image classification

classify\_button = tk.Button(root, text="Classify Image", command=classify\_image)

classify\_button.pack(pady=10)

# Run the Tkinter event loop

root.mainloop()