**ABSTRACT**

Early diagnosis is crucial in increasing the likelihood of successful treatment and recovery for fatal diseases like melanoma, which is a dangerous type of skin cancer that is increasing in prevalence. To aid in early detection, an automated system for dermatological disease recognition using lesion images is necessary. This system integrates multiple AI algorithms, including Convolution Neural Network with Multi Layer Perceptron (MLP), on image processing tools to achieve a high accuracy rate. The system is composed of three phases: collecting the data, augmentation, modelling and prediction. This system can be a valuable tool for dermatologists and physicians for early detection. In existing system, clustering of data based on machine learning have been used for diagnosis and resulted in imbalance clustering with low accuracy. In proposed system, deep learning algorithms like convolution neural network with Multi Layer Perceptron(MLP) is been deployed so that automatic detection is carried out with high accuracy. Lesion parameters such as symmetry, color. size, shape, etc, are used to detect skin cancer and to distinguish benign tumors from melanoma. This project presents a detailed systematic review of deep learning techniques for the early detection of melanoma skin cancer.

**CHAPTER 1**

**INTRODUCTION**

**1.1 DEEP LEARNING**

Deep learning is a subset of artificial intelligence (AI) that focuses on building and training neural networks to perform tasks that traditionally require human intelligence. Deep learning models, particularly deep neural networks, have achieved remarkable success in various domains such as computer vision, natural language processing, speech recognition, and medical diagnosis.

Here's how deep learning intersects with AI:

Neural Networks: Deep learning models are typically built using artificial neural networks, which are computational models inspired by the structure and function of the human brain. These networks consist of interconnected layers of nodes (neurons) that process input data and learn to extract relevant features for the task at hand.

Training and Learning: Deep learning models learn from data through a process called training. During training, the model adjusts its internal parameters (weights and biases) based on the input data and the desired output, using optimization algorithms like gradient descent. Through repeated iterations, the model learns to make accurate predictions or classifications.

Representation Learning: Deep learning excels at representation learning, where the model automatically learns to extract meaningful features from raw data. This ability to learn hierarchical representations of data enables deep neural networks to capture complex patterns and relationships in the input data.

Feature Extraction: In tasks such as computer vision and natural language processing, deep learning models can automatically extract features from raw input data, eliminating the need for manual feature engineering. This makes deep learning particularly effective for tasks with large and complex datasets.

Unsupervised Learning: Deep learning includes techniques for unsupervised learning, where the model learns to identify patterns in data without explicit labels or supervision. Examples include autoencoders for dimensionality reduction and generative adversarial networks (GANs) for generating realistic synthetic data.

Transfer Learning: Transfer learning is a common approach in deep learning, where pre-trained models on large datasets are fine-tuned for specific tasks with smaller datasets. This allows leveraging knowledge learned from one task to improve performance on another, accelerating the development of deep learning models for new applications.

Applications of Deep Learning: Deep learning has been successfully applied to a wide range of AI tasks, including image recognition, object detection, machine translation, sentiment analysis, recommendation systems, and autonomous driving. Its versatility and scalability make it a powerful tool for solving complex real-world problems.

Overall, deep learning plays a central role in advancing the capabilities of AI systems, enabling them to perform tasks with human-like intelligence and achieving state-of-the-art performance in various domains. As research in deep learning continues to progress, we can expect further breakthroughs and innovations that push the boundaries of AI capabilities.

**1.2 SKIN CANCER CLASSIFICATION**

Detecting melanoma skin cancer using deep learning is an area of significant research interest and application within the medical field. Melanoma is a type of skin cancer that arises from melanocytes, the cells that produce melanin, which gives skin its color. Early detection of melanoma is crucial for successful treatment and improved patient outcomes.

Deep learning, a subset of machine learning, has shown promising results in automating the process of melanoma detection from dermoscopic images or clinical images. Here's an introduction to how deep learning is applied in melanoma detection:

* Data Collection: Deep learning models require large amounts of labeled data to learn patterns effectively. Dermoscopic images, which are high-resolution images captured with a specialized device called a dermatoscope, are commonly used for training deep learning models for melanoma detection. These images capture detailed features of skin lesions that are not visible to the naked eye.
* Preprocessing: Before training a deep learning model, the dermoscopic images undergo preprocessing steps such as normalization, resizing, and augmentation to ensure consistency and enhance the model's ability to generalize to unseen data. Augmentation techniques like rotation, flipping, and scaling are often used to increase the diversity of the training dataset.
* Model Architecture: Convolutional Neural Networks (CNNs) are the most commonly used deep learning architecture for image-based tasks, including melanoma detection. CNNs are designed to automatically learn hierarchical representations of images by convolving filters across the input image. Transfer learning, where pre-trained CNN models are fine-tuned on dermatoscopic images, is also commonly employed to leverage knowledge learned from large-scale datasets like ImageNet.
* Training: During the training phase, the deep learning model learns to map input dermoscopic images to binary labels (melanoma or non-melanoma) by adjusting its internal parameters through backpropagation and gradient descent optimization. The model's performance is evaluated using metrics such as accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC).
* Validation and Testing: After training, the model's performance is evaluated on a separate validation dataset to assess its generalization ability and tune hyperparameters if necessary. The final model is then evaluated on an independent test dataset to provide an unbiased estimate of its performance.
* Deployment: Once a deep learning model demonstrates robust performance in detecting melanoma from dermoscopic images, it can be deployed in clinical settings to assist dermatologists in diagnosing skin lesions. Automated systems powered by deep learning can provide rapid and accurate assessments, potentially reducing the need for invasive biopsies and improving patient outcomes.

Overall, deep learning has shown great promise in revolutionizing melanoma detection by enabling accurate, efficient, and scalable diagnostic tools that complement the expertise of healthcare professionals. Continued research and development in this field hold the potential to further enhance the accuracy and accessibility of melanoma diagnosis and ultimately improve patient care.

**1.3 IMAGE CLASSIFICATION**

Skin image classification involves using machine learning techniques, including deep learning, to classify images of skin lesions into different categories, such as benign or malignant, or specific types of skin conditions like melanoma, nevus, or dermatitis. This classification is crucial for early detection, diagnosis, and treatment of skin diseases. Here's an overview of the process:

* Data Collection: Large datasets of labeled skin images are collected from medical databases, research institutions, or through collaboration with healthcare professionals. These images may include various types of skin lesions, captured under different conditions and with different imaging modalities such as dermoscopy or clinical photography.
* Data Preprocessing: The collected images undergo preprocessing steps to standardize their size, resolution, and color space. Additionally, techniques such as normalization and augmentation are applied to enhance the robustness and generalization ability of the model. Data augmentation techniques include rotation, flipping, scaling, and adding noise to the images.
* Model Selection: Various machine learning models can be used for skin image classification, ranging from traditional classifiers like Support Vector Machines (SVMs) and Random Forests to deep learning architectures such as Convolutional Neural Networks (CNNs). CNNs are particularly well-suited for image classification tasks due to their ability to automatically learn hierarchical representations of image features.
* Model Training: The selected model is trained using the preprocessed skin image data. During training, the model learns to map input images to their corresponding labels (e.g., benign or malignant) by adjusting its internal parameters through optimization algorithms like stochastic gradient descent (SGD) or Adam. The model's performance is evaluated using metrics such as accuracy, precision, recall, and F1 score on a separate validation dataset.
* Model Evaluation: After training, the model's performance is evaluated on an independent test dataset to assess its generalization ability and effectiveness in classifying unseen skin images. Performance metrics are calculated to measure the model's accuracy and reliability in differentiating between different classes of skin lesions.
* Model Deployment: Once the model demonstrates satisfactory performance, it can be deployed in clinical settings to assist dermatologists in diagnosing skin conditions. Automated systems powered by skin image classification models can provide rapid and accurate assessments, potentially reducing the need for invasive biopsies and improving patient outcomes.

Skin image classification using machine learning holds great promise for improving the early detection and diagnosis of skin diseases, thereby facilitating timely interventions and improving patient care. Continued research and development in this field are essential to further enhance the accuracy, efficiency, and accessibility of skin image analysis tools.

**1.4 OBJECTIVES**

In the context of skin image classification or any machine learning project, setting clear objectives is crucial for guiding the development process and measuring success. Here are some potential objectives for a skin image classification project:

Early Detection: Develop a model capable of accurately identifying skin lesions associated with melanoma and other types of skin cancer at an early stage, enabling timely intervention and treatment.

High Accuracy: Achieve high accuracy in classifying skin images into relevant categories, such as benign and malignant lesions, to minimize false positives and false negatives.

Generalization: Create a model that generalizes well to unseen data, including images captured under different conditions, from various sources, and with different types of skin lesions.

Interpretability: Develop a model that provides insights into the features contributing to its classification decisions, allowing dermatologists to understand and interpret the model's predictions.

Scalability: Build a scalable solution that can handle large volumes of skin image data efficiently, allowing for real-time or near-real-time analysis and classification.

Robustness: Ensure that the model is robust to factors such as variations in skin color, lighting conditions, image quality, and the presence of artifacts or noise in the images.

User-Friendly Interface: Design a user-friendly interface or application that allows healthcare professionals to easily upload, analyze, and interpret skin images, integrating seamlessly into their existing workflows.

Reduced Biopsy Rates: Develop a model that can accurately differentiate between benign and malignant lesions, potentially reducing the need for unnecessary biopsies and minimizing patient discomfort and healthcare costs.

Ethical Considerations: Address ethical considerations related to patient privacy, data security, and bias in the dataset or model predictions, ensuring that the solution adheres to ethical standards and regulatory requirements.

Clinical Validation: Validate the performance of the model through clinical studies or trials involving dermatologists and patients, demonstrating its effectiveness in real-world healthcare settings.

By defining clear objectives at the outset of the project, stakeholders can align their efforts, resources, and expectations towards developing a skin image classification solution that addresses key challenges and delivers tangible benefits to patients and healthcare providers.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 EARLY DETECTION OF SKIN CANCER - SOLUTION FOR IDENTIFYING AND DEFINING SKIN CANCERS USING AI**

**IEEE – 2022**

[**Narayana Darapaneni**](https://ieeexplore.ieee.org/author/37086822760)**;**[**Bhawna Sahni**](https://ieeexplore.ieee.org/author/37089425773)

Skin Cancer is seen as one of the most hazardous forms and common types of cancer in the world. Each year there are approximately more than 10 million new cases of skin cancer recorded globally - this number is alarming. The survival rate is very low if diagnosed in later stages. Artificial Intelligence can play a very important role in using Medical Image Diagnosis to detect this disease in early stages. However, the AI systems for the classification of different skin lesions, are still in the very early stages of clinical application in terms of being ready to aid in the diagnosis of skin cancers. Moreover, there are not many players who are doing research in this direction for conditions specified in the Indian subcontinent. The present paper focusses on advancement in AI solutions in digital image based computer vision for the diagnosis of skin cancer, Some of the challenges and future opportunities to improve the ability to diagnose skin cancer in early stages have also been discussed. Using the HAIS AI tool, we present a computer-aided method, using computer vision and image analysis algorithms for Skin Cancer diagnosis, with improved accuracy. Our solution is focused on the Indian sub-continent and envisions catering to varied business needs that provide flexibility on its adoption and use.

**2.2 PERFORMANCE OF MULTI LAYER PERCEPTRON AND DEEP NEURAL NETWORKS IN SKIN CANCER CLASSIFICATION**

**IEEE – 2021**

[**Yessi Jusman**](https://ieeexplore.ieee.org/author/38252950300)**;**[**Indah Monisa Firdiantika**](https://ieeexplore.ieee.org/author/37088835303)

Skin cancer refers to a condition where there exists abnormal growth of skin cells, mostly occurs on skin exposed to the sun. There are several types of skin cancer, where the most common types include basal cell carcinoma, squamous cell carcinoma, and melanoma. Without proper treatment, skin cancer, particularly in the melanoma form, can lead to deaths. Fortunately, early detection and classification of skin cancer are highly effective in preventing serious damages from skin cancer. In this paper, we train Multi-layer Perceptron, a custom convolutional neural network, and VGG-16 for skin cancer classification on a large skin cancer dataset, HAM10000. The performance of each trained model is subsequently compared and analyzed in terms of classification accuracy and computational time. Our experimental setups reveal that the VGG-16 model can set the best classification accuracy among the compared networks while in terms of testing time, the VGG-16 and custom CNN models are being much faster than the Multi-layer Perceptron. The results of our study are beneficial in providing systematic comparison and analysis of several neural networks in skin cancer classification.

**2.3 SKIN CANCER CLASSIFICATION AND DETECTION USING VGG-19 AND DESNET**

**IEEE- 2023**

[**Ashwinee Barbadekar**](https://ieeexplore.ieee.org/author/37887824700)**;**[**Varad Ashtekar**](https://ieeexplore.ieee.org/author/37089451145)**;**[**Atharva Chaudhari**](https://ieeexplore.ieee.org/author/37089362231)

Skin cancer is one of the most dangerous diseases in the world. Correctly classifying skin lesions at an early stage could aid clinical decision-making by providing an accurate disease diagnosis, potentially increasing the chances of cure before cancer spreads. However, achieving automatic skin cancer classification is difficult because the majority of skin disease images used for training are imbalanced and in short supply; meanwhile, the model’s cross-domain adaptability and robustness are also critical challenges. Recently, many deep learning-based methods have been widely used in skin cancer classification to solve the above issues and achieve satisfactory results. Nonetheless, reviews that include the abovementioned frontier problems in skin cancer classification are still scarce. Therefore, in this article, we provide a comprehensive overview of the latest deep learning-based algorithms for skin cancer classification. We begin with an overview of three types of dermatological images, followed by a list of publicly available datasets relating to skin cancers. After that, we review the successful applications of typical convolutional neural networks for skin cancer classification. As a highlight of this paper, we next summarize several frontier problems, including data imbalance, data limitation, domain adaptation, model robustness, and model efficiency, followed by corresponding solutions in the skin cancer classification task. Finally, by summarizing different deep learning-based methods to solve the frontier challenges in skin cancer classification, we can conclude that the general development direction of these approaches is structured, lightweight, and multimodal. Besides, for readers’ convenience, we have summarized our findings in figures and tables. Considering the growing popularity of deep learning, there are still many issues to overcome as well as chances to pursue in the future.

**2.4 SKIN CANCER DIAGNOSIS AND DETECTION USING DEEP LEARNING**

[**Djaroudib Khamsa**](https://ieeexplore.ieee.org/author/950963718034145)**;**[**Lorenz Pascal**](https://ieeexplore.ieee.org/author/567582619040307)**;**[**Bouidene Zakaria**](https://ieeexplore.ieee.org/author/439574540523839)

**IEEE - 2021**

Skin cancer is one of the most dangerous forms of cancer. Skin cancer is caused by un-repaired deoxyribonucleic acid (DNA) in skin cells, which generate genetic defects or mutations on the skin. Skin cancer tends to gradually spread over other body parts, so it is more curable in initial stages, which is why it is best detected at early stages. The increasing rate of skin cancer cases, high mortality rate, and expensive medical treatment require that its symptoms be diagnosed early. Considering the seriousness of these issues, researchers have developed various early detection techniques for skin cancer. Lesion parameters such as symmetry, color, size, shape, etc. are used to detect skin cancer and to distinguish benign skin cancer from melanoma. This paper presents a detailed systematic review of deep learning techniques for the early detection of skin cancer. Research papers published in well-reputed journals, relevant to the topic of skin cancer diagnosis, were analyzed. Research findings are presented in tools, graphs, tables, techniques, and frameworks for better understanding.

**2.5 SKIN CANCER CLASSIFICATION USING TRANSFER LEARNING-BASED PRE-TRAINED VGG 16 MODEL**

[**Archana Saini**](https://ieeexplore.ieee.org/author/37089827326)**;**[**Kalpna Guleria**](https://ieeexplore.ieee.org/author/37086920477)**;**[**Shagun Sharma**](https://ieeexplore.ieee.org/author/37089374544)

**IEEE - 2024**

Skin cancer is a serious and potentially life-threatening disease that affects millions of people worldwide. Early detection and accurate diagnosis are critical for successful treatment and improved patient outcomes. In recent years, deep learning has emerged as a powerful tool for medical image analysis, including the diagnosis of skin cancer. The importance of using deep learning in diagnosing skin cancer lies in its ability to analyze large amounts of data quickly and accurately. This can help doctors make more informed decisions about patient care and improve overall outcomes. Additionally, deep learning models can be trained to recognize subtle patterns and features that may not be visible to the human eye, leading to earlier detection and more effective treatment. The pre-trained Visual Geometry Group 16 (VGG16) architecture has been used in this study to classification of skin cancer images, and the images have been converted into other color scales, there are named: 1) Hue Saturation Value (HSV), 2) YCbCr, 3) Grayscale for evaluation. Results show that the dataset created with RGB and YCbCr images in field condition was promising with a classification accuracy of 84.242%. The dataset has also been evaluated with other popular architectures and compared. The performance of VGG16 with images of each color scale is analyzed. In addition, feature parameters have been extracted from the different layers. The extracted layers were felt with the VGG16 to evaluate the ability of the feature parameters in classifying the disease.

**CHAPTER 3**

**SYSTEM ANALYSIS**

* 1. **EXISTING SYSTEM**

In later days, skin cancer is seen as one of the foremost Dangerous frame of the Cancers found in People. Skin cancer is found in different sorts such as Melanoma, Basal and Squamous cell Carcinoma among which Melanoma is the foremost eccentric. The discovery of Melanoma cancer in early organize can be accommodating to remedy it. Computer vision can play critical part in Therapeutic Picture Determination and it has been demonstrated by numerous existing frameworks. In this paper, we show a computer helped strategy for the discovery of Melanoma Skin Cancer utilizing Picture Handling devices. The input to the system is the skin injury picture and after that by applying novel picture preparing methods, investigations it to conclude almost the nearness of skin cancer. The Injury Picture examination apparatuses checks for the different Melanoma parameters Like Asymmetry, Border, Colour, Diameter,(ABCD) etc. by surface, estimate and shape examination for picture division and include stages. The extricated highlight parameters are utilized to classify the picture as Ordinary skin and Melanoma cancer injury.

**3.1.1 Disadvantages of Existing System**

* Detection of the skin cancer with low variation is problematic
* Complexity of image based detection is identified
* The level variation is meant with different processing
  1. **PROPOSED SYSTEM**

There are many types of the skin cancer, each type has a different color, size and features. Many skin features may have impact on digital images like hair and color, and other impacts such as lightness, and type of the scanner or digital camera. In the first stage, we have obtained the features related with images using discrete wavelet transformation. In the second stage, the features of skin images have been reduced using principle component analysis to the more essential features. In the classification stage, two classifiers based on supervised machine learning have been developed. The first classifier based on feed forward back-propagation artificial neural network and the second classifier based on k-nearest neighbor. The classifiers have been used to classify subjects as normal or abnormal skin cancer images. A classification with a success of 95% and 97.5% has been obtained by the two proposed classifiers and respectively. This result shows that the proposed hybrid techniques are robust and effective.

**3.2.1 Advantages of the Proposed System**

* The skin cancer can be detected with a high accuracy
* The system implements with a good level of system
* The implementation work carried with a wide variety of system
  1. **FEASIBILITY STUDY**

A feasibility study for a skin cancer detection system would assess the viability and potential success of implementing such a system within the context of healthcare, technology, and societal factors. Here's an outline of key components to consider:

**Market Analysis:**

Identify the target market for the skin cancer detection system, including healthcare providers, medical institutions, and potentially consumers.

Analyze market trends related to skin cancer diagnosis and treatment, including the prevalence of skin cancer cases, current detection methods, and emerging technologies in dermatology.

**Clinical Needs and Requirements:**

Consult with dermatologists, oncologists, and other healthcare professionals to understand the clinical needs and challenges associated with skin cancer detection.

Identify specific requirements for the detection system, such as accuracy, sensitivity, specificity, ease of use, and integration with existing healthcare workflows.

**Technology Assessment:**

Evaluate existing technologies for skin cancer detection, including image analysis algorithms, machine learning models, and diagnostic tools.

Assess the technical feasibility of implementing automated or semi-automated systems for analyzing skin lesions, including image capture devices and software platforms.

**Data Collection and Analysis:**

Determine the availability and quality of data needed to train and validate the skin cancer detection algorithms.

Consider sources of data, such as clinical images, patient records, pathology reports, and demographic information, and evaluate data privacy and security concerns.

Regulatory and Compliance Considerations:

Identify regulatory requirements applicable to medical devices and diagnostic tools for skin cancer detection, such as FDA approval in the United States or CE marking in Europe.

Ensure compliance with standards for data privacy, patient confidentiality, and medical device safety.

**Risk Assessment:**

Evaluate potential risks associated with the skin cancer detection system, including false positives, false negatives, misdiagnosis, and patient safety concerns.

Develop strategies to mitigate risks and ensure the safe and effective use of the technology in clinical practice.

**Financial Analysis:**

Estimate the costs associated with developing, testing, and deploying the skin cancer detection system, including research and development expenses, regulatory compliance costs, and marketing expenses.

Conduct a cost-benefit analysis to assess the potential return on investment (ROI) and economic impact of the system, considering factors such as healthcare cost savings, improved patient outcomes, and market demand.

**Stakeholder Engagement:**

Engage key stakeholders, including healthcare providers, patients, insurers, regulators, and technology partners, to gather input, build support, and address concerns related to the skin cancer detection system.

Collaborate with healthcare organizations and research institutions to validate the performance and clinical utility of the system through clinical trials and real-world studies.

Implementation Plan:

Develop a detailed implementation plan outlining the steps required to develop, test, validate, and deploy the skin cancer detection system in clinical settings.

Consider factors such as pilot testing, user training, scalability, and sustainability to ensure successful adoption and integration into routine clinical practice.

**Ethical and Social Considerations**:

Consider ethical implications related to patient consent, privacy, equity, and access to healthcare services.

Address social factors such as healthcare disparities, patient education, and public awareness campaigns to promote early detection and prevention of skin cancer.

Based on the findings of the feasibility study, decisions can be made regarding the development, commercialization, and implementation of the skin cancer detection system. By addressing clinical, technological, regulatory, financial, and social considerations, the feasibility study helps to ensure the successful translation of innovative technologies into impactful solutions for improving healthcare outcomes.

* 1. **SYSTEM ARCHITECTURE**

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system.

There have been efforts to formalize languages to describe system architecture; collectively these are called architecture description languages (ADLs). Various organizations define systems architecture in different ways, including:

* An allocated arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirements baseline.
* Architecture comprises the most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales about the overall structure (i.e., essential elements and their relationships) and associated characteristics and behavior.
* If documented, it may include information such as a detailed inventory of current hardware, software and networking capabilities; a description of long-range plans and priorities for future purchases, and a plan for upgrading and/or replacing dated equipment and software.

An architecture diagram is a graphical representation of a set of concepts that are part of architecture, including their principles, elements and components. Architecture diagram can help system designers and developers visualize the high-level, overall structure of their system or application, in order to ensure the system meets their users' needs. Using architecture diagram, you can also describe patterns that are used throughout the design. It's somewhat like a blueprint that you use as a guide, so that you and your colleagues can discuss, improve and follow.

**Fig 3.1 System architecture**

**CHAPTER 4**

**PROJECT DESCRIPTION**

**4.1 PROBLEM DEFINITION**

Skin cancer could be a dangerous infection. Skin has three (3) fundamental layers. Skin cancer starts in furthest layer, which is made up of first layer squamous cells, moment layer basal cells, and deepest or third layer melanocytes cell. Squamous cell and basal cell are in some cases called non-melanoma cancers. Non-melanoma skin cancer continuously reacts to treatment and rarely spreads to other skin tissues. Melanoma is more perilous than most other types of skin cancer [3]. In case it isn't recognized at beginning organize, it is rapidly attack adjacent tissues and spread to other parts of the body. Formal diagnosis method to skin cancer discovery is Biopsy method. A biopsy may be a strategy to evacuate a bit of tissue or a test of cells from quiet body so that it can be dissected in a research facility. It is awkward strategy. Biopsy Strategy is time expending for understanding as well as specialist since it takes parcel of time for testing.

Biopsy is done by evacuating skin tissues (skin cells) which sample undergoes arrangement of laboratory testing [1]. There's plausibility of spreading of illness into other portion of body. It is more risky. Considering all the cases said over, So Skin cancer discovery utilizing svm is proposed. This strategy uses digital picture handling strategy and SVM for classification. This procedure has motivated the early location of skin cancers, and requires no oil to be connected to your skin to realize clear sharp pictures of your moles. In this way, it's quicker and cleaner approach. But, most imperatively, due to its higher amplification, Skin Cancer Location Utilizing SVM can prevent the superfluous extraction of flawlessly safe moles and skin injuries.

**4.2 OVERVIEW OF THE PROJECT**

Skin cancer can be defined as skin growths with differing causes and various degrees of malignancy. Skin cancer can also be referred to as Skin Neoplasm’s. Skin cancer develops on skin and so can be seen. The main cause of Skin cancer all over the world is UV radiations coming from the sun and it is estimated that Americans are greatly affected by skin cancers than the Africans and Asians. This is due to the fair complexion of their skin and so less melanin. Whereas Africans and Asians due to the high melanin content in the skin is far resistant to skin cancer [1]. It has been statistically proven that fairer skin toned people are much prone to tanning and so is prone to skin cancer. Cancer is the general name for a group of more than 100 diseases. Although there are different kinds of cancer, all cancers occur because abnormal cells grow out of control. Untreated cancers can cause serious illness and death.

Skin cancer is the most commonly occurring cancer. Skin cancer develops on skin and therefore from skin cells. Based on the type of skin cells, from which cancer arise, is classified into; Basal cell cancer [BCC], Squamous cell cancer [SCC], Melanoma Basal cell cancer Basal cell cancer is the most common skin cancer occurs in sun exposed areas. It rarely causes death as it rarely spreads. It is easily treated with surgery or radiation. Symptoms for basal cell cancer are: Raised, smooth, pearly bump on sun exposed skin (head, neck or shoulders).Small blood vessels are seen sometimes. Crusting and bleeding in the centre of the tumor. Squamous cell cancer It is less common than Basal cell cance cancer are red, scaling, thickened patch, ulceration and bleeding may occur and it develops into large mass if not treated.

**CHAPTER 5**

**MODULE DESCRIPTION**

**5.1 IMAGE ACQUISITION**

Input to proposed system is dermoscopic images, dermoscopic images are images taken by dermatoscope. It is kind of magnifier used to take pictures of skin lesions (body part). It is hand held instrument make it very easier to diagnose skin disease. The skin image acquisition is the process where the image is given to the system. The image can be in any size to be uploaded by the trainer or tester.Here the skin image are prepared for the further analysis of the cancer detection system

**Fig 5.1 Image acquisition**

**5.2 PRE-PROCESSING**

Goal of pre-processing is an improvement of image data that reduces unwanted distortions and enhances some image features important for further image processing. Image pre-processing involves three main things

1) Gray scale conversion

2) Noise removal.

**1) Grayscale conversion**

Grayscale image contains only brightness information. Each pixel value in grayscale image corresponds to an amount or quantity of light. The brightness graduation can be differentiated in grayscale image. Grayscale image measures only light intensity. 8 bit image will have brightness variation from 0 to 255 where ‘0’ represents black and ‘255’ represents white. In grayscale conversion colour image is converted into grayscale image shows in fig (6.2.1). Grayscale images are easier and faster to process than coloured images. All image processing technique are applied on grayscale image [4]. In our proposed system coloured or RBG image is converted into grayscale image by using weighted sum method by using following equations

Grayscale intensity= 0.299 R + 0.587 G + 0.114 B (6.1)

2) **Noise Removal**

The objective of noise removal is to detect and removed unwanted noise from digital image. The difficulty is in deciding which features of an image are real and which are caused by noise. Noise is random variations in pixel values. In our proposed system we are using median filter to remove unwanted noise shows in fig (4). Median filter is nonlinear filter, it leaves edges invariant. Median filter is implemented by sliding window of odd length [4]. Each sample value is sorted by magnitude, the centre most value is median of sample within the window, is a filter output.

**Fig 5.2 Image processing**

**5.3 IMAGE SHARPENING**

The objective of image sharpening is to process an image to increase visibility of feature of interest. Here contrast enhancement is used to get better quality result. Image sharpening is an effect applied to digital images to give them a sharper appearance. Almost all lenses can benefit from at least a small amount of sharpening. Here the image sharpening is applied with the verification of the pre-processed image to get a vital pixel.

**5.4 SEGMENTATION**

Segmentation is process of removing region of interest from given image. Region of interest containing each pixel similar attributes. Here we are using maximum entropy thresholding for segmentation [5]. First of all we have to take gray level of original image then calculate histogram of gray scale image then by using maximum entropy separate foreground from background. After maximum entropy we obtained binary image that is black and white image shows in fig 6.2.2. Image segmentation is performed by using our proposed automatic thresholding and masking operation in R, G and B planes. First, automatic thresholding proposed by Otsu12 is applied in each plane. Binary masks for each plane are obtained and then combined to produce a final lesion mask. We use 3-plane masking procedure to increase segmentation accuracy. Then edge detection is applied to further segmentation. The main prerequisite for extracting the features is that the lesion must be separated from the surrounding normal skin. But the segmented image may contain other smaller blobs which are not the skin lesion. To overcome this, we find the biggest blob in the segmented image. The segmented image obtained contains only the skin lesion.

**Fig 5.3 Image segmentation**

The skin-images on which we are going to work contain the cancerous mole along with the skin part, so it is very important to remove the region of interest. So for this purpose we are going to use contour segmentation method.

**5.5 FEATURE EXTRACTION**

To create a GLCM, the gray co-matrix function is used. The gray-level co-occurrence matrix (GLCM) is created by gray co matrix function. This is done by determining how often a pixel with the intensity (gray-level) value i occurs in a specific spatial relationship to a pixel with the value j. Each element (i, j) in GLCM is found by the sum of the number of times that the pixel with value i occurred in the specified spatial relationship to a pixel with value j in the input image. Because the processing is required to calculate a GLCM for the full dynamic range of an image is prohibitive, gray co-matrix scales the input image. The scaling is used by gray co-matrix for reducing the number of intensity values in grayscale image from 256 to eight. The size of the GLCM is determined by the number of gray levels. The number of gray levels the matrix called GLCM and intensity value scaling can be controlled by the Num Levels and the Gray Limits parameters. Certain properties of the spatial distribution of the gray scale image can be revealed by gray-level co-occurrence matrix.

For instance, when most of the values in the GLCM are clustered along the diagonal, the texture is coarse with respect to the specified offset. Several statistical measures can be derived from the GLCM. Segmentation is followed by feature extraction. No machine learning algorithm can work without predefined features set. The type of features can be broadly divided into following categories.

**Fig 5.4 Feature extraction**

The main features of the Melanoma skin Lesion are its Geometric Feature. Hence, we propose to extract the Geometric Features of segmented skin lesion.

Here, we used some standard geometry features (Area, Perimeter, Greatest Diameter, Circularity Index, and Irregularity Index) adopted. From the Segmented image containing only skin lesion, the image blob of the skin lesion is analyzed to extract the it’s geometrical features. The Different Features extracted are as follows: Area (A): Number of pixels of the lesion. Perimeter (P): Number of edge pixels. Major Axis Length or Greatest Diameter (GD): The length of the line passing through lesion centroid and connecting the two farthest boundary points.

* 1. **FURIA CLASSIFICATION**

Using the FURIA rules for the melanoma skin cancer, we use some pre-defined thresholds in classification stage. The Feature Values Extracted in the Feature Extraction stage is compared and the skin lesion is classified as Melanoma Skin Cancer or normal skin or Mole. This classification method proves to be efficient for most of the skin images. FURIA is often referred to as technique for reducing the number of variables in a data set without loss of information and as a possible process for identifying new variables in to another smaller set the newly created variables are not usually easy to interpret. FURIA has been most successful in applications such as image compression where data reduction and not interpretation is of primary importance. FURIA allows one to identify the uncorrelated components of an ensemble of data. FURIA is used for classification, to classify the skin cancer. FURIA uses a method of analysis which involves finding the linear combination of a set of variables that has maximum variance and removing its effect and then testing and training is done. With the results of testing and training, FURIA will find whether the given values are benign or malignant. If the values are below 1 then it is benign. In case, the values are above 1, it’s a malignant. FURIA extends the well-known RIPPER algorithm, a state-of-the-art rule learner, while preserving its advantages, such as simple and comprehensible rule sets. With the generative rule set the detection of the algorithm can be done with the verified pattern which is used/ the possible feature identification with FURIA will tends a fuzzy classification with the generative rule scheme. The detection of the full training features and tested with the extracted values

**Fig 5.5 FURIA Segmentation**

Classifier is used to classify cancerous image from other skin diseases. For simplicity FURIA classifier is used here. FURIA takes set of images and predicts for each input image belongs to which of the two categories of cancerous and non-cancerous classes. The purpose of FURIA is create hyper plane that separates two classes with maximum gap between them

**CHAPTER 6**

**REQUIREMENTS ENGINEERING**

**6.1 HARDWARE SPECIFICATION**

* Processor : Intel core processor 2.6.0 GHz
* RAM : 8GB
* Hard disk : 256 GB

Keyboard : Standard keyboard

* Monitor : 15 inch color monitor

**6.2 SOFTWARE SPECIFICATION**

* Front End : PYTHON
* IDE : Pycharm
* Platform : Windows 10

**CHAPTER 7**

**SYSTEM DESIGN**

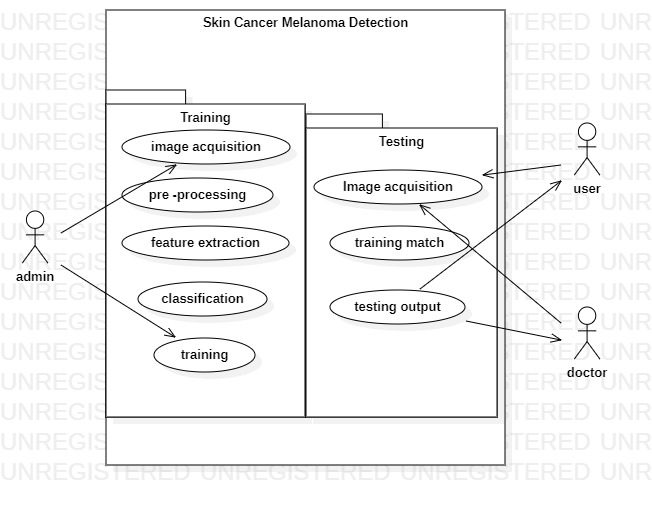
A Unified Modeling Language (UML) diagram is a visual representation of a system's structure and behavior using standardized symbols and notations. For a skin cancer detection system, we can create several types of UML diagrams to illustrate different aspects of the system. Here are a few examples:

**7.1 USE CASE DIAGRAM:**

Use case diagrams represent the interactions between users (actors) and the system, depicting the system's functionalities from a user's perspective.

Actors in a skin cancer detection system might include dermatologists, patients, and administrators.

Use cases could include "Capture Skin Image," "Analyze Skin Image," "View Diagnosis Result," etc.



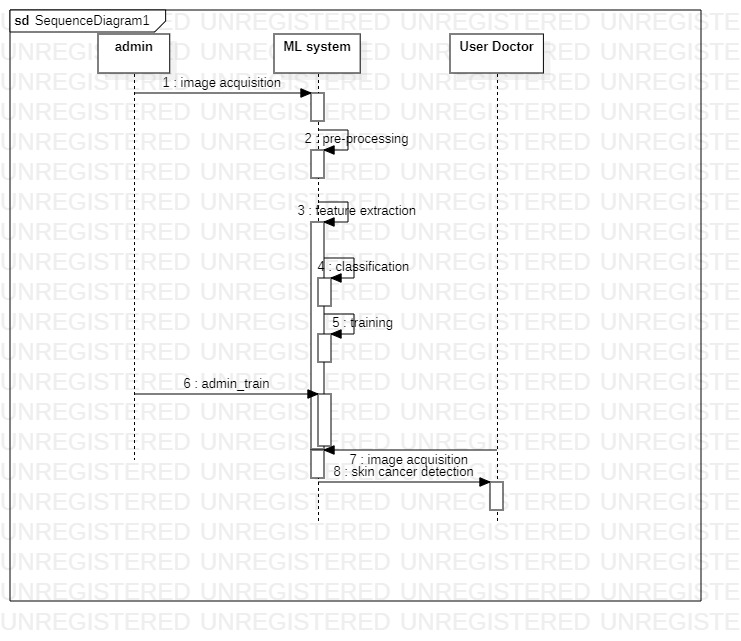
**Fig 7.1 Use Case diagram**

**7.2 SEQUENCE DIAGRAM:**

Sequence diagrams illustrate the interactions between objects or components over time, showing the sequence of messages exchanged.

A sequence diagram for the skin cancer detection system could depict the flow of interactions between the user, image capture device, diagnostic algorithm, and database.

It could show steps such as "User captures skin image," "Image is sent to diagnostic algorithm," "Diagnostic result is returned to user."

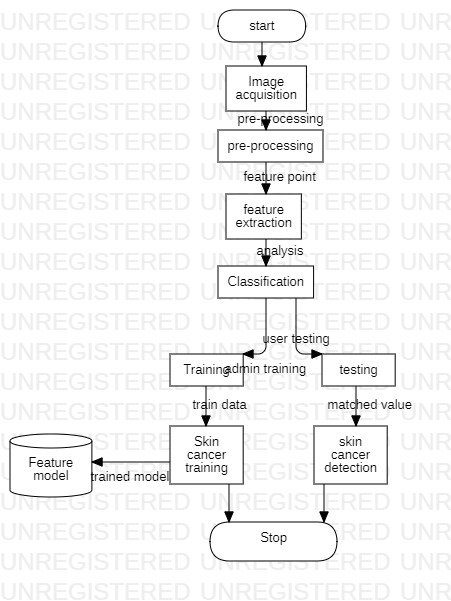


**Fig 7.2 Sequence Diagram**

**7.3 ACTIVITY DIAGRAM:**

Activity diagrams depict the workflow or process flow within the system, showing activities, decisions, and transitions.

An activity diagram for the skin cancer detection system might show the steps involved in analyzing a skin image, including preprocessing, feature extraction, classification, and result presentation.



**Fig 7.3 Activity Diagram**

**7.4 DATA FLOW DIAGRAM**

A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in.

**Data flow Symbols:**

|  |  |
| --- | --- |
| **Symbol** | **Description** |
| http://cpanel.stpaulsscience.org/gceict/specifications/ocr/unit3/sdlc/dfd/entity.jpg | An **entity**. A source of data or a destination for data. |
| http://cpanel.stpaulsscience.org/gceict/specifications/ocr/unit3/sdlc/dfd/process.jpg | A **process** or task that is performed by the system. |
| http://cpanel.stpaulsscience.org/gceict/specifications/ocr/unit3/sdlc/dfd/store.jpg | A **data store**, a place where data is held between processes. |
| http://cpanel.stpaulsscience.org/gceict/specifications/ocr/unit3/sdlc/dfd/flow.jpg | A **data flow**. |

This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

**LEVEL 0**

DFD Level 0 is also called a Context Diagram. It’s a basic overview of the whole system or process being analyzed or modeled. It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and

Image

Skin cancer detection

**Fig 7.4 level 0-DFD**

**LEVEL 1**

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub – processes. Level 1 - interaction between 2 different business applications. This is primarily used to explain the process to business and tech leads, QA leads. As described previously, context diagrams (level 0 DFDs) are diagrams where the whole system is represented as a single process. A level 1 DFD notates each of the main sub-processes that together form the complete system. We can think of a level 1 DFD as an “exploded view” of the context diagram.

Image

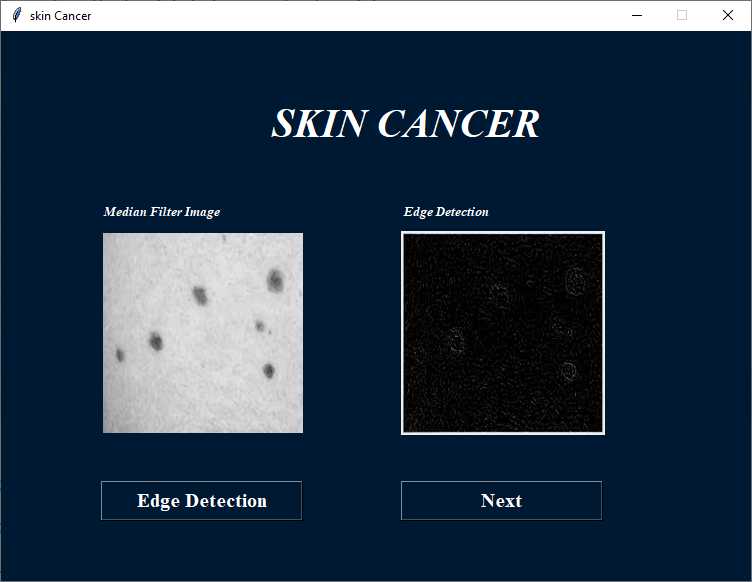
**Fig 7.5 level 0-DFD**

**CHAPTER 8**

**SYSTEM TESTING AND IMPLEMENTATION**

**8.1 SYSTEM TESTING**

System Testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer-based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.



**Fig 8.1 System testing**

**Two Category of System Testing**

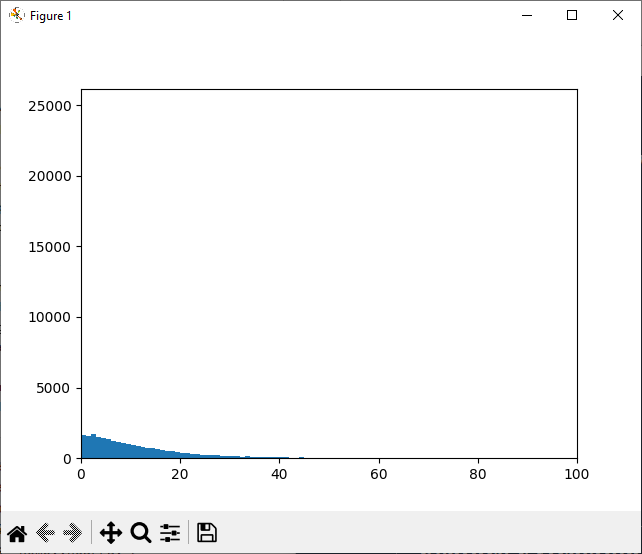
* **Black Box Testing**
* **White Box Testing**

**System test falls under the black box testing category of software testing**.

* **White box testing** is the testing of the internal workings or code of a software application. In contrast,
* Black box or System Testing is the opposite. System test involves the external workings of the software from the user's perspective.

**System Testing involves testing the software code for following**

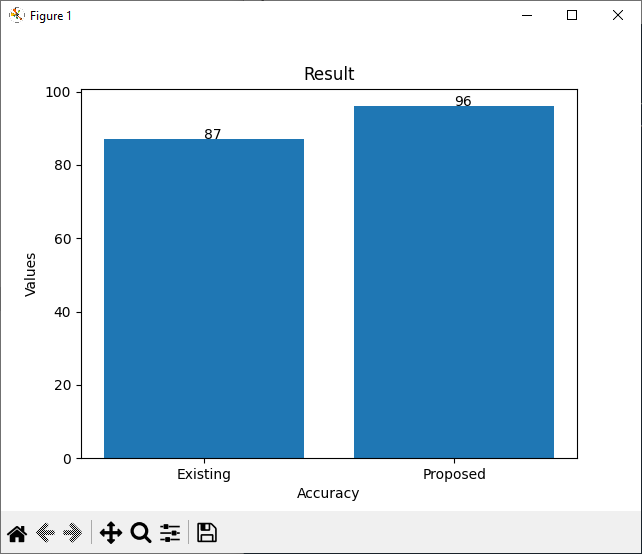
* Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called End to End testing scenario.
* Verify thorough testing of every input in the application to check for desired outputs.
* Testing of the user's experience with the application. That is a very basic description of what is involved in system testing. You need to build detailed test cases and test suites that test each aspect of the application as seen from the outside without looking at the actual source code.



**Fig 8.2 White and Black box testing**

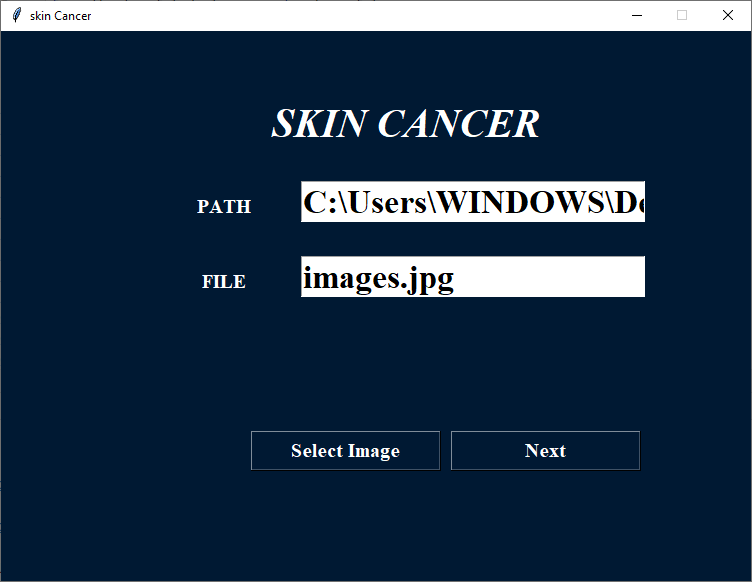
**These are the steps taken to fully test new software in preparation for marketing it:**

* **Unit testing -** testing performed on each module or block of code during development. Unit Testing is normally done by the programmer who writes the code.
* **Integration testing -** testing done before, during and after integration of a new module into the main software package. This involves testing of each individual code module. One piece of software can contain several modules which are often created by several different programmers. It is crucial to test each module's effect on the entire program model.



**Fig 8.3 Integration testing**

* **Validation Testing -** Validation testing is the Quality Assurance (QA) process of verifying that a software application meets the needs and requirements of its stakeholders. The main goal of validation testing is to verify whether a software product meets its acceptance criteria
* **System testing -** testing done by a professional testing agent on the completed software product before it is introduced to the market.



**Fig 8.4 Validation Testing**

**Acceptance testing -** beta testing of the product done by the actual end users

**CHAPTER 9**

**CONCLUSION AND FUTURE ENHANCEMENT**

**9.1 CONCLUSION**

It can be easily concluded that the proposed system of skin cancer detection can be implemented using gray level co-occurrence matrix and support vector machine to classify easily whether image is cancerous or non-cancerous. Accuracy of proposed system is 95%. It is painless and timeless process than biopsy method. It is more advantageous to patients. This project we have discussed a computer-aided diagnosis system for melanoma skin cancer. It can be concluded from the results that the proposed system can be effectively used by patients and physicians to diagnose the skin cancer more accurately. This tool is more useful for the rural areas where the experts in the medical field may not be available. Since the tool is made more users friendly and robust for images acquired in any conditions, it can serve the purpose of automatic diagnostics of the Skin Cancer.

**9.2 FUTURE ENHANCEMENT**

In future system, by comparing the first two techniques it is found that FURIA and takes less processing time, whereas ABCD method gives an accuracy of 90%. Third classification technique is the method called AIS (Artificial Immune System) using clonal selection method which is the future work.

**APPENDIX**

* + 1. **SOURCECODE**

Ar\_master

import pymysql

class master\_flask\_code:

def \_\_init\_\_(self):

self.user = 'root'

self.password = ''

self.host = 'localhost'

self.database = 'python\_intelligent\_attendance'

def find\_max\_id(self,table):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

cursor.execute("SELECT id FROM "+table)

data = cursor.fetchall()

maxin = len(data)

if maxin == 0:

maxin = 1

else:

maxin += 1

return maxin

def insert\_query(self,qry):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

result=cursor.execute(qry)

conn.commit()

conn.close()

return result

def select\_login(self,qry):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

cursor.execute(qry)

data = cursor.fetchall()

check = len(data)

if check == 0:

return 'no'

else:

return 'yes'

def select\_single\_colum(self,table,colum):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

qry1=("select "+colum+" from "+table)

cursor = conn.cursor()

cursor.execute(qry1)

data = cursor.fetchall()

return data

def admin\_login(self,username,password):

if username == 'admin' and password == 'admin':

return 'yes'

else:

return 'no'

Median Filter

import shutil

from tkinter import \*

import os

import cv2

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

# image = Image.open(r"median.png")

# image = image.convert("L")

# image = image.filter(ImageFilter.FIND\_EDGES)

# image.save('img\_blur.png')

root.destroy()

import edge\_detection

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

im1 = Image.open(r"greyscale.png")

im2 = im1.filter(ImageFilter.MedianFilter(size=3))

im2.save('median.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Grayscal Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Median Filter Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open('greyscale.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('median.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Median Filter",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

Main.py

from tkinter import \*

from PIL import ImageTk,Image

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

dd=0

root.destroy()

import median\_filter

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

filapath=sample\_data.student.file\_path

img = Image.open(filapath).convert('L')

img.save('greyscale.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="RGB Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Grayscal Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open(sample\_data.student.file\_path)

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('greyscale.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Grayscale",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

Feature Extraction

import shutil

from tkinter import \*

import os

from tkinter import messagebox

import random

import cv2

import imagehash as imagehash

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.pyplot as plt1

import cv2

import accuracy\_value

import sample\_data

def image\_matching(a,b):

i1 = Image.open(a)

i2 = Image.open(b)

i1 = i1.resize((230, 200), Image.ANTIALIAS)

i2 = i2.resize((230, 200), Image.ANTIALIAS)

assert i1.mode == i2.mode, "Different kinds of images."

assert i1.size == i2.size, "Different sizes."

pairs = zip(i1.getdata(), i2.getdata())

if len(i1.getbands()) == 1:

# for gray-scale jpegs

dif = sum(abs(p1-p2) for p1,p2 in pairs)

else:

dif = sum(abs(c1-c2) for p1,p2 in pairs for c1,c2 in zip(p1,p2))

ncomponents = i1.size[0] \* i1.size[1] \* 3

xx= (dif / 255.0 \* 100) / ncomponents

return xx

def testing():

name = []

values = []

input\_image=sample\_data.student.file\_path

entries = os.listdir('train/')

for x in entries:

val=100

directory=x

name.append(x)

x1="train/"+x

arr = os.listdir(x1)

for x2 in arr:

path=x1+"/"+str(x2)

find=image\_matching(path,input\_image)

if(find<val):

val=find

values.append(val)

values\_lenght= len(values)

pos=0;

pos\_val=100

result="unknown"

for x in range(0, values\_lenght):

if values[x]<pos\_val:

pos=x

pos\_val=values[x]

print(pos\_val)

if(pos\_val<20):

result=name[pos]

messagebox.showinfo(title="info", message=str(result))

plt.close()

nn = accuracy\_value.sample()

nn.demo()

################################################################## read dataset

def read\_first\_data():

img\_file = 'img\_blur.png'

img = cv2.imread(img\_file, cv2.IMREAD\_COLOR) # rgb

alpha\_img = cv2.imread(img\_file, cv2.IMREAD\_UNCHANGED) # rgba

gray\_img = cv2.imread(img\_file, cv2.IMREAD\_GRAYSCALE) # grayscale

print(type(img))

print('RGB shape: ', img.shape) # Rows, cols, channels

print('ARGB shape:', alpha\_img.shape)

print('Gray shape:', gray\_img.shape)

print('img.dtype: ', img.dtype)

print('img.size: ', img.size)

hash0 = imagehash.average\_hash(Image.open(img\_file))

im = cv2.imread(img\_file)

vals = im.mean(axis=2).flatten()

counts, bins = np.histogram(vals, range(257))

plt.bar(bins[:-1] - 0.5, counts, width=1, edgecolor='none')

plt.xlim([0, 100])

plt.show()

################################################################## Next page

def next\_page():

name = "train"

if os.path.exists(name):

h = 0;

else:

os.mkdir(name)

name=txt1.get()

name1 = txt2.get()

if (name == ""):

messagebox.showinfo(title="info", message="Enter Disease Details")

elif(name1==""):

messagebox.showinfo(title="info", message="Enter Description")

else:

name1 = "train\\" + name

if os.path.exists(name1):

j = 0;

else:

os.mkdir(name1)

ri2 = Image.open(sample\_data.student.file\_path)

ri2 = ri2.resize((230, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

x = random.randint(10000000, 100000000)

str = name1 + '\\%d.jpg' % x

shutil.copy(sample\_data.student.file\_path, str)

messagebox.showinfo(title="info", message="Success")

################################################################## main loop

image = Image.open(sample\_data.student.file\_path)

width, height = image.size

hash0 = imagehash.average\_hash(Image.open(sample\_data.student.file\_path))

print(hash0)

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Feature Extraction",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

# message = Label(root, text="Feature Extraction",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

# message.place(x=400, y=170)

ri2 = Image.open('img\_blur.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

# txt=Entry(root,width=10)

# txt.place(x=400, y=200)

# lbl2 = Label(root)

# a1 = Image.open('img\_blur.png')

# a123 = a1.resize((200, 200), Image.ANTIALIAS)

# a12 = ImageTk.PhotoImage(a123)

#

# lbl2.configure(image=a12)

message = Label(root, text="Disease",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

txt1 = Entry(root, width=15, font=('times', 25, ' bold '))

txt1.place(x=400, y=190)

message = Label(root, text="Description",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=240)

txt2 = Entry(root, width=15, font=('times', 25, ' bold '))

txt2.place(x=400, y=260)

#

#

# txt2 = Entry(root, width=15, font=('times', 25, ' bold '))

# txt2.place(x=400, y=280)

######## button with command function

compare\_dataset = Button(root, text="Feature Extraction",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 20, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Training",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 20, ' bold '))

resust\_dataset.place(x=400, y=310)

resust\_dataset = Button(root, text="Testing",width=16 ,height=1,command=testing,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 20, ' bold '))

resust\_dataset.place(x=400, y=360)

root.mainloop()

Edge Detection

import shutil

from tkinter import \*

import os

import cv2

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

# image = Image.open(r"median.png")

# image = image.convert("L")

# image = image.filter(ImageFilter.FIND\_EDGES)

# image.save('img\_blur.png')

root.destroy()

import feature\_extraction

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

image = Image.open(r"median.png")

image = image.convert("L")

image = image.filter(ImageFilter.FIND\_EDGES)

image.save('img\_blur.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Median Filter Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Edge Detection",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open('median.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('img\_blur.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Edge Detection",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

Ar\_master

import pymysql

class master\_flask\_code:

def \_\_init\_\_(self):

self.user = 'root'

self.password = ''

self.host = 'localhost'

self.database = 'python\_intelligent\_attendance'

def find\_max\_id(self,table):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

cursor.execute("SELECT id FROM "+table)

data = cursor.fetchall()

maxin = len(data)

if maxin == 0:

maxin = 1

else:

maxin += 1

return maxin

def insert\_query(self,qry):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

result=cursor.execute(qry)

conn.commit()

conn.close()

return result

def select\_login(self,qry):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

cursor.execute(qry)

data = cursor.fetchall()

check = len(data)

if check == 0:

return 'no'

else:

return 'yes'

def select\_single\_colum(self,table,colum):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

qry1=("select "+colum+" from "+table)

cursor = conn.cursor()

cursor.execute(qry1)

data = cursor.fetchall()

return data

def admin\_login(self,username,password):

if username == 'admin' and password == 'admin':

return 'yes'

else:

return 'no'

Median Filter

import shutil

from tkinter import \*

import os

import cv2

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

# image = Image.open(r"median.png")

# image = image.convert("L")

# image = image.filter(ImageFilter.FIND\_EDGES)

# image.save('img\_blur.png')

root.destroy()

import edge\_detection

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

im1 = Image.open(r"greyscale.png")

im2 = im1.filter(ImageFilter.MedianFilter(size=3))

im2.save('median.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Grayscal Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Median Filter Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open('greyscale.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('median.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Median Filter",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

Main.py

from tkinter import \*

from PIL import ImageTk,Image

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

dd=0

root.destroy()

import median\_filter

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

filapath=sample\_data.student.file\_path

img = Image.open(filapath).convert('L')

img.save('greyscale.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="RGB Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Grayscal Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open(sample\_data.student.file\_path)

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('greyscale.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Grayscale",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

Feature Extraction

import shutil

from tkinter import \*

import os

from tkinter import messagebox

import random

import cv2

import imagehash as imagehash

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.pyplot as plt1

import cv2

import accuracy\_value

import sample\_data

def image\_matching(a,b):

i1 = Image.open(a)

i2 = Image.open(b)

i1 = i1.resize((230, 200), Image.ANTIALIAS)

i2 = i2.resize((230, 200), Image.ANTIALIAS)

assert i1.mode == i2.mode, "Different kinds of images."

assert i1.size == i2.size, "Different sizes."

pairs = zip(i1.getdata(), i2.getdata())

if len(i1.getbands()) == 1:

# for gray-scale jpegs

dif = sum(abs(p1-p2) for p1,p2 in pairs)

else:

dif = sum(abs(c1-c2) for p1,p2 in pairs for c1,c2 in zip(p1,p2))

ncomponents = i1.size[0] \* i1.size[1] \* 3

xx= (dif / 255.0 \* 100) / ncomponents

return xx

def testing():

name = []

values = []

input\_image=sample\_data.student.file\_path

entries = os.listdir('train/')

for x in entries:

val=100

directory=x

name.append(x)

x1="train/"+x

arr = os.listdir(x1)

for x2 in arr:

path=x1+"/"+str(x2)

find=image\_matching(path,input\_image)

if(find<val):

val=find

values.append(val)

values\_lenght= len(values)

pos=0;

pos\_val=100

result="unknown"

for x in range(0, values\_lenght):

if values[x]<pos\_val:

pos=x

pos\_val=values[x]

print(pos\_val)

if(pos\_val<20):

result=name[pos]

messagebox.showinfo(title="info", message=str(result))

plt.close()

nn = accuracy\_value.sample()

nn.demo()

################################################################## read dataset

def read\_first\_data():

img\_file = 'img\_blur.png'

img = cv2.imread(img\_file, cv2.IMREAD\_COLOR) # rgb

alpha\_img = cv2.imread(img\_file, cv2.IMREAD\_UNCHANGED) # rgba

gray\_img = cv2.imread(img\_file, cv2.IMREAD\_GRAYSCALE) # grayscale

print(type(img))

print('RGB shape: ', img.shape) # Rows, cols, channels

print('ARGB shape:', alpha\_img.shape)

print('Gray shape:', gray\_img.shape)

print('img.dtype: ', img.dtype)

print('img.size: ', img.size)

hash0 = imagehash.average\_hash(Image.open(img\_file))

im = cv2.imread(img\_file)

vals = im.mean(axis=2).flatten()

counts, bins = np.histogram(vals, range(257))

plt.bar(bins[:-1] - 0.5, counts, width=1, edgecolor='none')

plt.xlim([0, 100])

plt.show()

################################################################## Next page

def next\_page():

name = "train"

if os.path.exists(name):

h = 0;

else:

os.mkdir(name)

name=txt1.get()

name1 = txt2.get()

if (name == ""):

messagebox.showinfo(title="info", message="Enter Disease Details")

elif(name1==""):

messagebox.showinfo(title="info", message="Enter Description")

else:

name1 = "train\\" + name

if os.path.exists(name1):

j = 0;

else:

os.mkdir(name1)

ri2 = Image.open(sample\_data.student.file\_path)

ri2 = ri2.resize((230, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

x = random.randint(10000000, 100000000)

str = name1 + '\\%d.jpg' % x

shutil.copy(sample\_data.student.file\_path, str)

messagebox.showinfo(title="info", message="Success")

################################################################## main loop

image = Image.open(sample\_data.student.file\_path)

width, height = image.size

hash0 = imagehash.average\_hash(Image.open(sample\_data.student.file\_path))

print(hash0)

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Feature Extraction",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

# message = Label(root, text="Feature Extraction",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

# message.place(x=400, y=170)

ri2 = Image.open('img\_blur.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

# txt=Entry(root,width=10)

# txt.place(x=400, y=200)

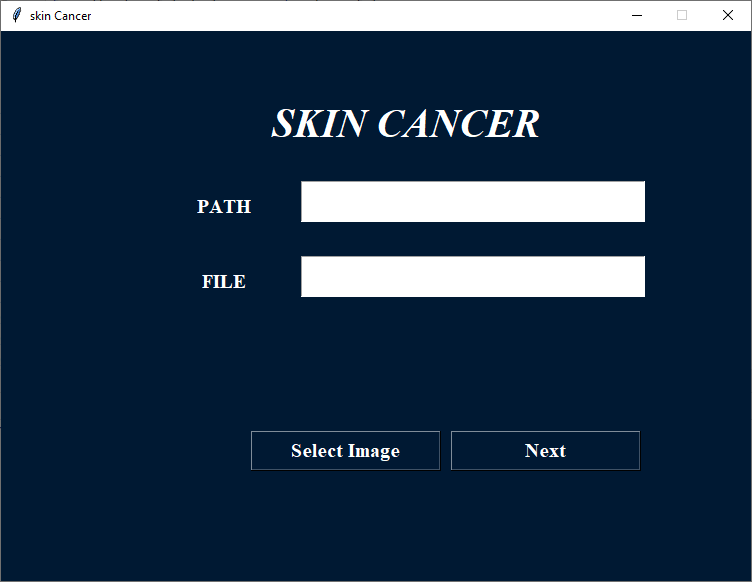
# lbl2 = Label(root)

# a1 = Image.open('img\_blur.png')

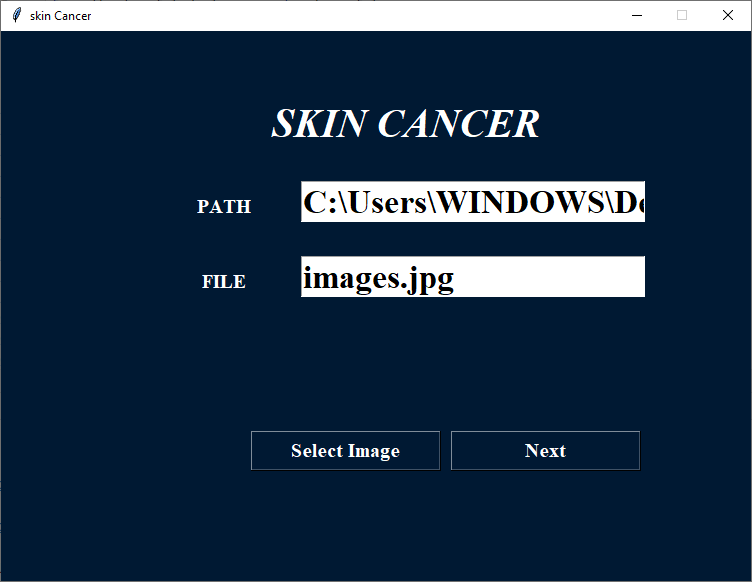
# a123 = a1.resize((200, 200), Image.ANTIALIAS)

# a12 = ImageTk.PhotoImage(a123)

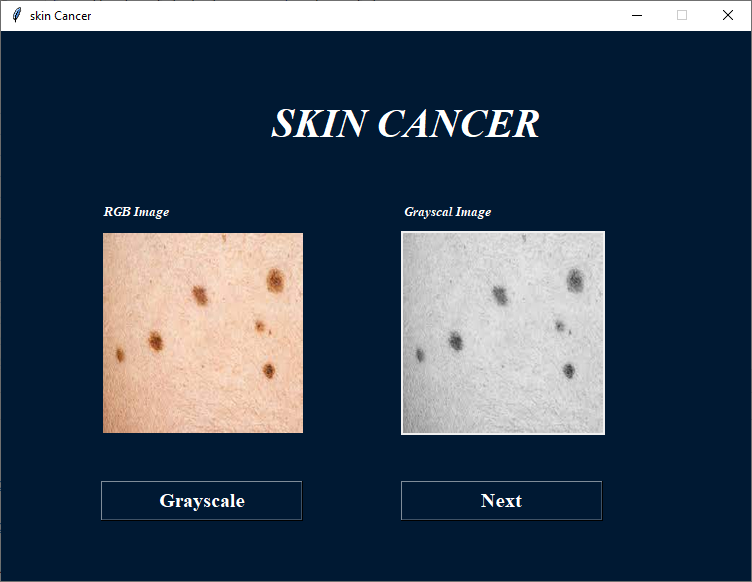
* + 1. **SCREENSHOT**

****

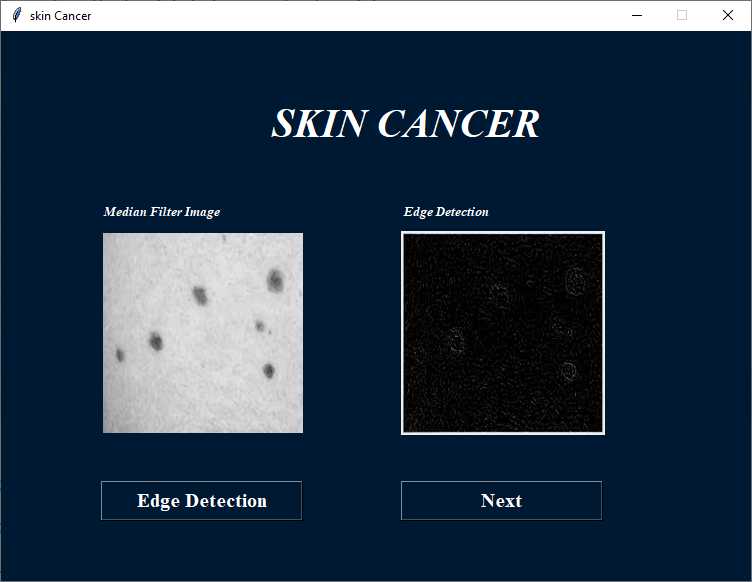
**Image loading of the input**



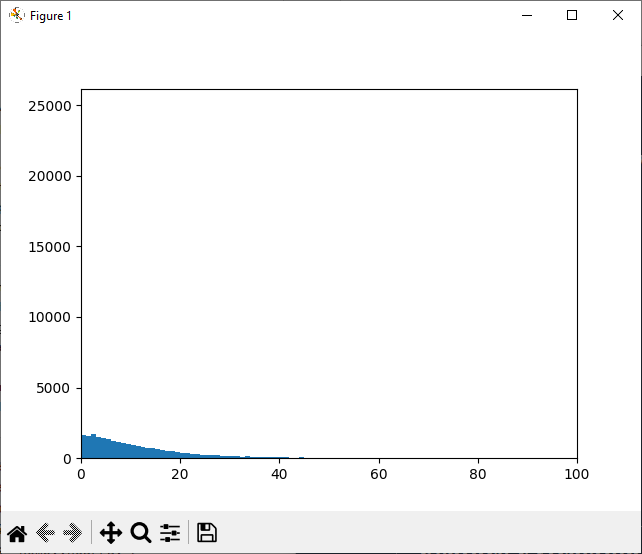
**Image loading**



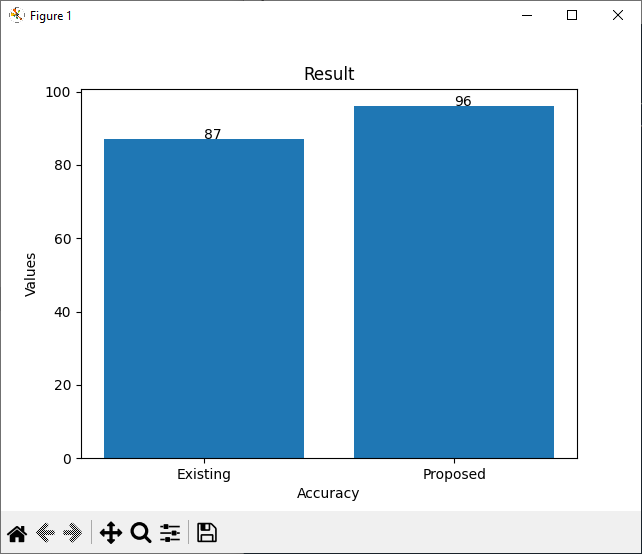
**Image view**

****

**Edge detection**

****

**Graph analysis**

****

**Comparison**