**SKIN DISEASE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK WITH VISION TRANSFORMER**

**Abstract**

Skin disease diagnosis and classification are crucial tasks in dermatology that can benefit significantly from advancements in deep learning and image processing techniques. In this study, we propose a comprehensive approach that combines convolutional neural networks (CNNs) with vision transformer models for skin disease classification, adaptive Gaussian thresholding for denoising, and the Sobel algorithm for segmentation. The first step involves denoising the input images using adaptive Gaussian thresholding, which helps remove noise and enhance the clarity of important features in the images. Next, we employ the Sobel algorithm for edge detection and segmentation, which allows us to focus on the relevant regions within the images for further analysis. For the classification task, we leverage the power of CNNs combined with vision transformer models. CNNs excel at capturing spatial patterns and local features, while vision transformers are effective in modeling long-range dependencies and global context in the images. By integrating these two architectures, our model can effectively learn hierarchical representations of skin disease images, leading to accurate and robust classification results. Experimental results demonstrate the effectiveness of our proposed approach compared to traditional methods. The combination of denoising, segmentation, and classification techniques yields superior performance in terms of accuracy, precision, recall, and F1 score, showcasing the potential of deep learning and image processing fusion in advancing skin disease diagnosis and classification systems.

**SYSTEM REQUIREMENTS**

**Hardware Requirements**

* SYSTEM : I3 2.4 GHz
* HARD DISK : 40 GB
* MONITOR : 15 VGA colour
* MOUSE : Logitech.
* RAM : 4 GB
* KEYBOARD : 110 keys enhanced.

**Software Requirements**

* Operating system :- Windows 10
* Environment :- Anaconda
* Coding Language :- Python
* IDE :- Spyder

**Introduction**

Skin cancer is the uncontrolled growth of abnormal skin cells. It appears during the process of unrepaired DNA damage to the skin cells mutations that is triggers, defects occurs in genes which are leading the skin cells for multiplying gradually and also framed the malignant type of tumours. Moreover, skin cancer is categorized into three different types of skin cancers such as basal cell, squamous and melanomas skin cancers. Even though, the major classifications of skin cancer are melanoma and non-melanoma. Here, the Melanoma is a malignancy of the cells which provide the melanocytes (skin colour) and able to invade the neighbouring tissues. In addition, it is also able to spread throughout the human body and may lead to cause of death. On the other hand, the non-melanoma spreads in to other parts of the body.

Malignant melanoma is a most fatal form of human skin cancers that led to a raised mortality value. Recently, the incidence of melanoma is increasing reasonably, particularly the white skinned people. In North America, the melanoma type of cancer diseases became the fifth important cancer among male genders and the sixth important cancer among female genders in Australia. In addition, based on reports that are released from the country where the people living western countries of the world. Moreover, the melanoma type of cancer is a seventh most important malignancy in female genders and also the male genders in the sixth position. Otherwise, it is a curable type of skin cancer disease only when identified the disease in early stage. In this direction, the early stage of melanoma‘s skin cancer disease can be diagnosed with a simple excision which is able to reduce the mortality rate.

Dermoscopy is a tool which is used for diagnosing the melanoma and it is also used by dermatologists due to its value in detecting melanoma in its early stages. Moreover, it achieves good visualization of many pigmented structures like dots, pigment networks and streaks and blue with white areas that are not visible to the naked eye. Hence, need more than two features for identifying the given lesion image as melanoma. Thus, dermoscopy images are used to become more confident over the process of distinguish the categories of lesions.

In the process of melanoma detection in cancer diagnosis system, the dermatologists are adopting the ABCD rule which is used for analysing the four standard parameters such as Diameter, Asymmetry, Colours and Border for diagnosing the melanoma at starting stage. Moreover, the melanoma images with hair and reflection and the invisible of melanoma borders that creates a visual identification very complex task to the experts who are involving the process of skin cancer diagnosis. In addition, the elucidation of the melanoma images is efficient and subjective even for the trained dermatologists. The Computer Aided Diagnosis (CAD) became essential for managing all these challenges to assist the physicians, interpret the images clearly and also to take the correct decision over the diagnosis process in melanoma detection. Moreover, this kinds of diagnosis systems used for reducing the time taken for diagnosing the disease and it also used for improving the detection accuracy.

There are many research works have been done in the application areas of temporal data mining techniques where the time at which an event has been happened is stored efficiently in the database with time constraints. Even though, most of the data mining techniques that are handled the data in temporal databases at best as data series in chronological order and avoid the time intervals. A new Temporal Query and Mining Language (TQML) to perform mining the temporal patterns in the databases with temporal nature that provides a special consideration for the temporal attributes so that the prediction is become an accurate and the easier. In all the existing works that are focused over the temporal database management or the temporal mining. Even though, security problem is a big problem which is to be addressed in the current applications and also it is necessary for providing a temporal database management system with a right query language that is able to carry out the temporal reasoning through temporal data mining to provide effective security systems.

**Existing system**

The research gap in skin disease diagnosis and classification lies in the need for comprehensive and automated systems that integrate advanced deep learning and image processing techniques to address the challenges of variability in skin lesions, subjectivity in diagnosis, and time-intensive manual analysis. While existing studies have explored individual aspects such as CNN-based classification or image segmentation algorithms, there is a lack of research that combines these techniques synergistically to create a holistic approach. Specifically, the integration of vision transformer models with CNNs for learning hierarchical representations, coupled with denoising techniques like adaptive Gaussian thresholding and segmentation using the Sobel algorithm, remains largely unexplored. Additionally, there is a scarcity of studies that rigorously evaluate such integrated systems using diverse datasets and benchmark metrics, hindering the validation of their effectiveness in real-world clinical settings. Bridging this research gap is essential to develop robust and reliable automated systems for skin disease diagnosis, ultimately improving patient outcomes and healthcare efficiency.

Existing Drawbacks:

1. **Limited Contextual Understanding**: Traditional CNNs may struggle with understanding global context due to their local receptive fields.
2. **Dependency on Manual Feature Engineering**: Traditional methods often rely on handcrafted features, which might not capture all relevant information effectively.
3. **Sensitivity to Noise and Variability**: Without robust preprocessing steps, models can be adversely affected by noise and variations in image quality.

**Proposed System:**

The proposed system integrates several advanced techniques to enhance skin disease classification. Firstly, adaptive Gaussian thresholding is employed for denoising, effectively removing noise and enhancing the clarity of important features in input images. This preprocessing step is crucial for improving the quality of the input data and facilitating more accurate classification. Next, the Sobel algorithm is utilized for edge detection and segmentation, enabling focusing on the relevant regions within the images for further analysis. By isolating these regions, meaningful features essential for accurate classification can be extracted.

For the classification task itself, the strengths of both convolutional neural networks (CNNs) and vision transformer models are leveraged. CNNs excel at capturing spatial patterns and local features, while vision transformers are effective in modeling long-range dependencies and global context in the images. Integrating these two architectures enables the model to effectively learn hierarchical representations of skin disease images, leading to accurate and robust classification results. This fusion of CNNs and vision transformers allows for capturing both local and global features, enabling a more comprehensive understanding of the images.

Advantages:

1. **Improved Accuracy**: By combining denoising, segmentation, and classification techniques, the proposed system achieves higher accuracy in skin disease classification compared to traditional methods.
2. **Enhanced Robustness**: The fusion of CNNs and vision transformers, along with robust preprocessing steps, improves the model's robustness to noise and variability in input images.
3. **Efficient Global Context Understanding**: Vision transformers provide a mechanism for understanding global context, complementing the local feature extraction capabilities of CNNs.
4. **Reduced Dependency on Manual Feature Engineering**: The proposed system reduces reliance on manual feature engineering by learning representations directly from raw data, potentially improving adaptability to diverse skin disease types and image variations.

**Background study**

Adegun, A., & Viriri, S. [1]these authors research offers a comprehensive review of the methods currently used to assess skin lesion pictures. In order to diagnose melanoma, pictures of skin lesions are analyzed, and this page gives a detailed examination of the techniques and algorithms utilized to do so. These approaches range from initial preparation through feature extraction, segmentation algorithms, and classification strategies. There has been an examination of both traditional and cutting-edge methods. The highest-rated algorithms were dissected for their inner workings.

Ali, M. et al. [2] In terms of classification accuracy, the DCNN model shown here beats other transfer learning methods. The model's efforts to enhance accuracy were further hampered by the imbalanced dataset and the scarcity of training images. For this reason, we ensured that the dataset had an equal number of cases from each class. We also trained a number other transfer learning models on the same dataset, however their performance lagged below that of our recommended DCNN model.

Dascalu, A., & David, E. O. [3] a computer-aided diagnosis Superimposing a sonification method on a complex neural network, a crude dermoscope was put to the test. The dermoscopy's sensory components, time constraints, physical discomfort of obtaining photos, and physicians' capacity to overcome or contribute to a "God complex" may all be lessened or relieved by technology breakthroughs in the identification of skin cancer. The study's use of teledermatology aided in preventive medicine and might aid in healthcare systems' ongoing efforts to save costs.

Han, S. et al. [4] the author show that deep learning algorithms trained with large samples of both Asian and Caucasian populations may be used to cancer diagnosis, therapy recommendation, and illness categorization with results that are competitive with those of human specialists.

Hasan, M. et al. [5] The goal of this research was to propose using Convolutional Neural Networks to classify melanomas. It's now possible to distinguish benign from malignant skin cancers thanks to a newly created classification system. Experimental and assessment results show the model has promise as a reference for helping dermatologists make diagnoses of skin cancer. Even if any doctor may get adequate findings by randomly choosing photographs, the standard method of patient diagnosis takes a long time.

Hekler, A. et al. [6] this is the first research that the author are aware of to integrate dermatologists' clinical judgment with AI for digital skin diagnostics. Using 300 test photos from a diverse data set known for its excellent external validity, we found that the'man with machine' strategy was better.

Mijwil, M. [11] To determine whether photographed cases of skin cancer are malignant or benign, we use a convolutional neural network model trained with a range of parameters in this research. The results show that the classifier is reliable, with no controversial results. This completely generalized cancer detection in pictures is achieved using the InceptionV3 architecture, which has a very steady success rate across both classes. This framework has a high rate of correct diagnoses (86.7%) and excellent performance.

Monika, M. et al. [12] The incidence of skin cancer is rising dramatically over the world for several reasons. In light of this, this study proposes an MSVM-based classification strategy that employs two powerful algorithms, ABCD and MSVM, to extract features. The level of accuracy is almost 96.25 percent. The suggested approach classifies eight different skin malignancies with a high degree of accuracy and precision.

**Problem statement**

Skin diseases present a formidable challenge in dermatology due to their diverse visual manifestations and the inherent subjectivity in manual diagnosis, leading to potential errors and delays in treatment. Addressing these challenges requires advanced computational techniques that can automate and improve the accuracy of skin disease diagnosis and classification. The proposed study integrates convolutional neural networks (CNNs) with vision transformer models to effectively learn hierarchical representations of skin disease images, complemented by adaptive Gaussian thresholding for denoising and the Sobel algorithm for segmentation, enabling focused analysis on relevant regions. By leveraging the strengths of deep learning and image processing fusion, the system aims to provide accurate and robust classification results, addressing the variability, subjectivity, and time-intensive nature of traditional diagnosis methods and showcasing the potential for advancing skin disease diagnosis and classification systems in healthcare.

**Research contribution**

* Introduces a novel denoising technique using adaptive Gaussian thresholding.
* Removes noise from skin disease images, improving the clarity of important features and enhancing the quality of input data for the classification model.
* Incorporates the Sobel algorithm for edge detection and segmentation of skin disease images.
* Enables focused analysis on relevant regions within the images, reducing computational complexity and ensuring the classification process targets informative parts of the image.
* Combines the strengths of CNNs in capturing spatial patterns and local features with vision transformers' ability to model long-range dependencies and global context.
* Enhances the classification accuracy and robustness by learning hierarchical representations from skin disease images.

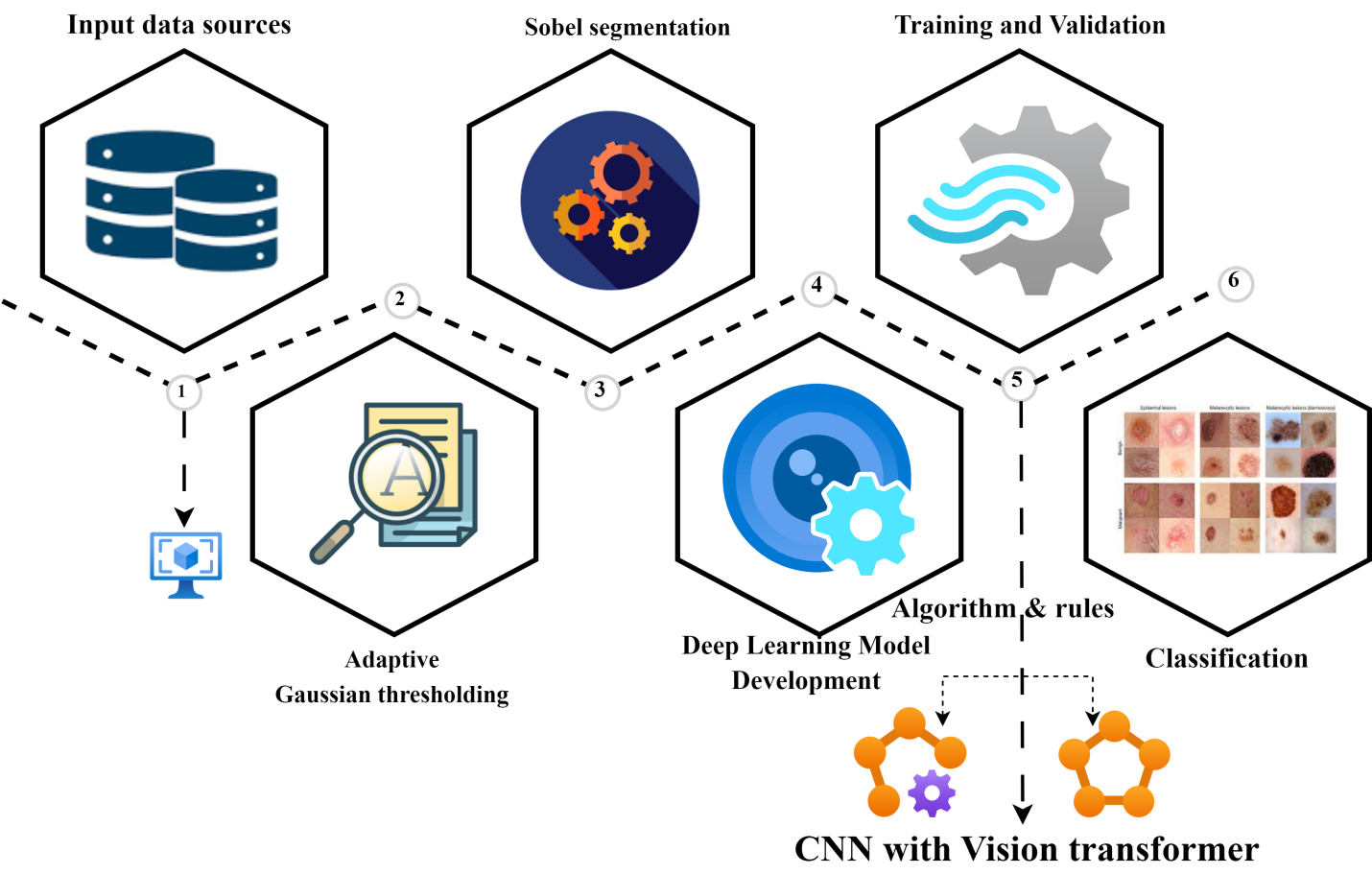
**Research objective**

1. **Develop a Comprehensive Classification System**:
   * Design and implement a holistic approach that integrates CNNs with vision transformer models for skin disease classification.
   * Create a framework that combines denoising techniques like adaptive Gaussian thresholding and segmentation using the Sobel algorithm to enhance classification accuracy.
2. **Enhance Image Preprocessing Techniques**:
   * Investigate and optimize adaptive Gaussian thresholding as a denoising method for skin disease images, focusing on improving feature clarity and noise reduction.
   * Explore the effectiveness of the Sobel algorithm for edge detection and segmentation, aiming to identify and isolate relevant regions within the images.
3. **Evaluate Model Performance and Robustness**:
   * Conduct extensive experiments to evaluate the proposed classification system's performance in terms of accuracy, precision, recall, and F1 score.
   * Compare the performance of the integrated CNN-vision transformer model with traditional methods and assess its robustness across different skin disease datasets.
4. **Demonstrate the Advantages of Fusion Techniques**:
   * Showcase the advantages of combining CNNs with vision transformer models for learning hierarchical representations and capturing both local and global features in skin disease images.
   * Highlight how the fusion of deep learning and image processing techniques improves the diagnostic capabilities of automated systems in dermatology.
5. **Contribute to Advancements in Skin Disease Diagnosis**:
   * Contribute to the advancement of automated systems for skin disease diagnosis by addressing key challenges such as variability in skin lesions, subjectivity in diagnosis, and time-intensive analysis.
   * Provide insights and recommendations for further research and development in
   * the field of computer-aided diagnosis for dermatological conditions.

**Modules**

1. **Data Collection and Preprocessing Module**:
   * Collect a diverse dataset of skin disease images, including various types of lesions and conditions.
   * Preprocess the images by standardizing the resolution, adjusting brightness and contrast, and applying data augmentation techniques to enhance the dataset's diversity.
2. **Adaptive Gaussian Thresholding for Denoising Module**:
   * Implement the adaptive Gaussian thresholding algorithm to remove noise and enhance the clarity of skin disease images.
   * Tune the thresholding parameters adaptively based on local image characteristics to achieve optimal denoising results.
3. **Sobel Algorithm-based Segmentation Module**:
   * Develop a segmentation module using the Sobel algorithm for edge detection and region segmentation in skin disease images.
   * Extract relevant regions and features from the images, focusing on areas of interest for classification.
4. **CNN with Vision Transformer Model Integration Module**:
   * Design and implement a hybrid model that combines CNNs with vision transformer architectures for skin disease classification.
   * Utilize pre-trained CNN and vision transformer models (such as ViT or DeiT) and fine-tune them on the segmented and denoised images for feature extraction and classification.
5. **Training and Validation Module**:
   * Split the dataset into training, validation, and test sets for model training and evaluation.
   * Train the integrated model using the segmented, denoised images, and validate its performance using appropriate metrics such as accuracy, precision, recall, and F1 score.

|  |
| --- |
| **Algorithm 1: CNN with** **Vision Transformer** |
| **Input:**   * Skin disease images with dimensions (), where is height, is width, and is the number of channels.   **Steps:**   **Preprocess Images:**   * Normalize the pixel values of images to the range [0, 1]. * Resize images to a consistent input size suitable for the CNN and vision transformer models.    **Split Dataset:**   * Divide the dataset into training, validation, and test sets for model training and evaluation.    **Extract Features using CNN:**   * Initialize a pre-trained CNN model to extract features from preprocessed images. * Pass each image through the CNN model to obtain feature maps of shape , where and are spatial dimensions, and is the number of feature channels.    **Transform Features using Vision Transformer:**   * Reshape the feature maps into sequences of tokens by flattening and reshaping, resulting in input tokens of shape , where is the batch size, is the sequence length, and is the embedding dimension. * Initialize a vision transformer model () with trainable weights and hyperparameters. * Pass the input tokens through the vision transformer model to obtain transformed feature embeddings of shape.    **Combine CNN and Vision Transformer Outputs:**   * Flatten and concatenate the transformed feature embeddings along the sequence length dimension to create a combined feature vector of shape. * Optionally, apply a linear projection or fusion layer to further integrate and refine the features from CNN and vision transformer models, resulting in a final feature vector of shape , where is the fused feature dimension.    **Classification using Fully Connected Layers:**   * Initialize a fully connected neural network (FCN) classifier with trainable weights and bias parameters. * Pass the final feature vector through the FCN classifier to obtain logits of shape, where is the number of classes/categories. * Apply a softmax activation function to the logits to obtain class probabilities of shape .   **Output:**   * Trained model capable of accurately classifying skin disease images. |

****

**SYSTEM STUDY**

**FEASIBILTY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution. The analyst conducting the study gathers information using a variety of methods, the most popular of which are:

* Interviewing users, employees, managers, and customers.
* Developing and administering questionnaires to interested stakeholders, such as potential users of the information system.
* Observing or monitoring users of the current system to determine their needs as well as their satisfaction and dissatisfaction with the current system.
* Collecting, examining, and analyzing documents, reports, layouts, procedures, manuals, and any other documentation relating to the operations of the current system.
* Modeling, observing, and simulating the work activities of the current system.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components. These components are:

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY
* OPERATIONAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**OPERATIONAL FEASIBILITY**

The ability, desire, and willingness of the stakeholders to use, support, and operate the proposed computer information system. The stakeholders include management, employees, customers, and suppliers. The stakeholders are interested in systems that are easy to operate, make few, if any, errors, produce the desired information, and fall within the objectives of the organization.

**SYSTEM ENVIRONMENT**

**LANGUAGE FEATURES**

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

**Statements and control flow**

The assignment statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of variables) illuminates many other features of the language. Assignment in C, e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address. The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed binding the name to the object. Since the name's storage location doesn't contain the indicated value, it is improper to call it a variable. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating storage to (at most) three names and one numeric object, to which all three names are bound. Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it. However at a given time a name will be bound to some object, which will have a type; thus there is dynamic typing.

The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if). The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block. The while statement, which executes a block of code as long as its condition is true.

The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits. The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming. The def statement, which defines a function or method. The with statement (from Python 2.5), which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing Resource Acquisition Is Initialization (RAII)-like behavior. The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block. The assert statement, used during debugging to check for conditions that ought to apply. The yield statement, which returns a value from a generator function. From Python 2.5, yield is also an operator. This form is used to implement coroutines.

The import statement, which is used to import modules whose functions or variables can be used in the current program. There are four ways of using import: import <module name> or from <module name> import \* or import numpy as np or from numpy import pi as Pie. The print statement was changed to the print() function in Python 3. Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will. However, better support for coroutine-like functionality is provided in 2.5, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed unidirectionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

**Tkinter**

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit,[1] and is Python's de facto standard GUI.[2] Tkinter is included with the standard Microsoft Windows and Mac OS X install of Python.

The name Tkinter comes from Tk interface. Tkinter was written by Fredrik Lundh.

As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

Python 2.7 and Python 3.1 incorporate the "themed Tk" ("ttk") functionality of Tk 8.5. This allows Tk widgets to be easily themed to look like the native desktop environment in which the application is running, thereby addressing a long-standing criticism of Tk (and hence of Tkinter).

tkinter provides us with a variety of common GUI elements which we can use to build our interface – such as buttons, menus and various kinds of entry fields and display areas. We call these elements widgets. We are going to construct a tree of widgets for our GUI – each widget will have a parent widget, all the way up to the root window of our application. For example, a button or a text field needs to be inside some kind of containing window.

The widget classes provide us with a lot of default functionality. They have methods for configuring the GUI’s appearance – for example, arranging the elements according to some kind of layout – and for handling various kinds of user-driven events. Once we have constructed the backbone of our GUI, we will need to customise it by integrating it with our internal application class.

**Window**

This term has different meanings in different contexts, but in general it refers to a rectangular area somewhere on the user's display screen.

**Top Level Window**

A window that exists independently on the screen. It will be decorated with the standard frame and controls for the desktop manager. It can be moved around the desktop, and can usually be resized.

**Widget**

The generic term for any of the building blocks that make up an application in a graphical user interface. Examples of widgets: buttons, radiobuttons, text fields, frames, and text labels.

**Frame**

In Tkinter, the Frame widget is the basic unit of organization for complex layouts. A frame is a rectangular area that can contain other widgets.

**Child and parent**

When any widget is created, a parent-child relationship is created. For example, if you place a text label inside a frame, the frame is the parent of the label.

It have often been asked: "Is there no Tk for Python?" or "Is Tkinter the same as Tk?" Of course, there is Tk for Python. Without Tk Python would be less attractive to many users. Tk is called Tkinter in Python, or to be precise, Tkinter is the Python interface for Tk. Tkinter is an acronym for "Tk interface".

Tk was developed as a GUI extension for the Tcl scripting language by John Ousterhout. The first release was in 1991. Tk proved as extremely successful in the 1990's, because it is easier to learn and to use than other toolkits. So it is no wonder that many programmers wanted to use Tk independently of Tcl. That's why bindings for lots of other programming languages have been developed, including Perl, Ada (called TASH), Python (called Tkinter), Ruby, and Common Lisp.   
  
Tk provides the following widgets:

* button
* canvas
* checkbutton
* combobox
* entry
* frame
* label
* labelframe
* listbox
* menu
* menubutton
* message
* notebook
* tk\_optionMenu
* panedwindow
* progressbar
* radiobutton
* scale
* scrollbar
* separator
* sizegrip
* spinbox
* text
* treeview

It provides the following top-level windows:

* tk\_chooseColor - pops up a dialog box for the user to select a color.
* tk\_chooseDirectory - pops up a dialog box for the user to select a directory.
* tk\_dialog - creates a modal dialog and waits for a response.
* tk\_getOpenFile - pops up a dialog box for the user to select a file to open.
* tk\_getSaveFile - pops up a dialog box for the user to select a file to save.
* tk\_messageBox - pops up a message window and waits for a user response.
* tk\_popup - posts a popup menu.
* toplevel - creates and manipulates toplevel widgets.

Tk also provides three geometry managers:

* place - which positions widgets at absolute locations
* grid - which arranges widgets in a grid
* pack - which packs widgets into a cavity

### Distutils Basics and Design Flaws

Distutils contains commands, each of which is a class with a run method that can be called with some options. Distutils also provides a Distribution class that contains global values every command can look at.

To use Distutils, a developer adds a single Python module to a project, conventionally called setup.py. This module contains a call to Distutils' main entry point: the setup function. This function can take many options, which are held by a Distribution instance and used by commands. Here's an example that defines a few standard options like the name and version of the project, and a list of modules it contains:

from distutils.core import setup

setup(name='MyProject', version='1.0', py\_modules=['mycode.py'])

This module can then be used to run Distutils commands like sdist, which creates a source distribution in an archive and places it in a dist directory:

$ python setup.py sdist

Using the same script, you can install the project using the install command:

$ python setup.py install

Distutils provides other commands such as:

* upload to upload a distribution into an online repository.
* register to register the metadata of a project in an online repository without necessary uploading a distribution,
* bdist to creates a binary distribution, and
* bdist\_msi to create a .msi file for Windows.

It will also let you get information about the project via other command line options.

So installing a project or getting information about it is always done by invoking Distutils through this file. For example, to find out the name of the project:

$ python setup.py --name

MyProject

setup.py is therefore how everyone interacts with the project, whether to build, package, publish, or install it. The developer describes the content of his project through options passed to a function, and uses that file for all his packaging tasks. The file is also used by installers to install the project on a target system.

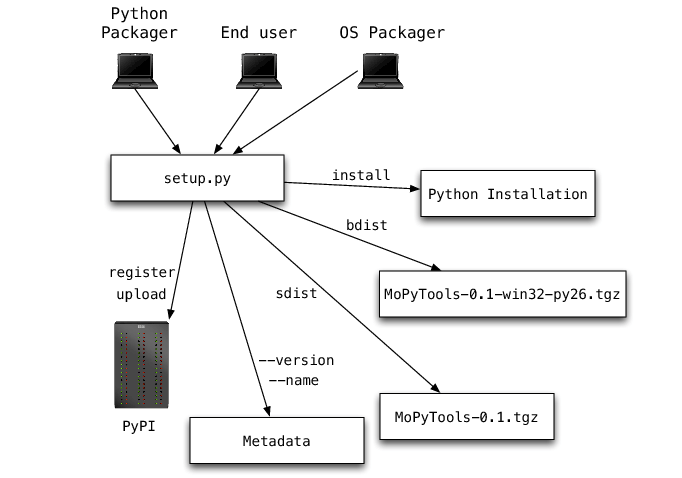


Figure 14.1: Setup

## *What is Python?*

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

**Introduction**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to www browsers to games.

## History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

## Python Features

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.
* Apart from the above-mentioned features, Python has a big list of good features, few are listed below −
* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

## Local Environment Setup

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

* Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
* Win 9x/NT/2000
* Macintosh (Intel, PPC, 68K)
* OS/2
* DOS (multiple versions)
* PalmOS
* Nokia mobile phones
* Windows CE
* Acorn/RISC OS
* BeOS
* Amiga
* VMS/OpenVMS
* QNX
* VxWorks
* Psion
* Python has also been ported to the Java and .NET virtual machines.

### Integrated Development Environment

You can run Python from a Graphical User Interface (GUI) environment as well, if you have a GUI application on your system that supports Python.

* **Unix** − IDLE is the very first Unix IDE for Python.
* **Windows** − PythonWin is the first Windows interface for Python and is an IDE with a GUI.
* **Macintosh** − The Macintosh version of Python along with the IDLE IDE is available from the main website, downloadable as either MacBinary or BinHex'd files.

If you are not able to set up the environment properly, then you can take help from your system admin. Make sure the Python environment is properly set up and working perfectly fine.

# My-SQL

# MySQL is a simple, yet powerful Open Source Software relational database management system that uses SQL. MySQL is a true multi-user, multithreaded SQL database server. MySQL is fast and flexible enough to store logs and pictures in it. Its main goals are speed, robustness, and ease of use. Most likely MySQL and PHP combination is encountered today and probably for the years to come.

This basic MySQL explains some of the basic SQL statements. If this is the first time you have used a relational database management system, this tutorial gives you everything you need to know to work with MySQL such as querying data, updating data, managing databases, and creating tables. If you are already familiar with other relational database management systems such as PostgreSQL, Oracle, and Microsoft SQL Server. you can use this tutorial to refresh your knowledge and understand how SQL dialect of MySQL is different from other systems.

**Strengths of MySQL**

* MySQL has the biggest market share of any open source database. Almost any web-hosting company can provide.
* After setting database and access to it, managing the database is straightforward. Initial access needs to be configured by a database administrator. Tools such as MySQL Administrator or phpMyAdmin help to manage database.
* SQL has always been relatively fast, much due to its simplicity. In the last few years, MySQL has gained foothold in the enterprise market due to new “enterprise class” features and general maturity without compromising performance for simple usage.

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. This tutorial will give you a quick start to MySQL and make you comfortable with MySQL programming.

* A Relational DataBase Management System (RDBMS) is a software that −
* Enables you to implement a database with tables, columns and indexes.
* Guarantees the Referential Integrity between rows of various tables.
* Updates the indexes automatically.
* Interprets an SQL query and combines information from various tables.
* RDBMS Terminology
* Before we proceed to explain the MySQL database system, let us revise a few definitions related to the database.
* Database − A database is a collection of tables, with related data.
* Table − A table is a matrix with data. A table in a database looks like a simple spreadsheet.
* Column − One column (data element) contains data of one and the same kind, for example the column postcode.
* Row − A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
* Redundancy − Storing data twice, redundantly to make the system faster.
* Primary Key − A primary key is unique. A key value can not occur twice in one table. With a key, you can only find one row.
* Foreign Key − A foreign key is the linking pin between two tables.
* Compound Key − A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
* Index − An index in a database resembles an index at the back of a book.

Referential Integrity − Referential Integrity makes sure that a foreign key value always points to an existing row.

What is a Database?

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those type of systems.

Nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as Foreign Keys.

* MySQL Database
* MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons −
* MySQL is released under an open-source license. So you have nothing to pay to use it.
* MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language.
* MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
* MySQL works very quickly and works well even with large data sets.
* MySQL is very friendly to PHP, the most appreciated language for web development.
* MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

## Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, **Manpower** and **manpower** are two different identifiers in Python.

Here are naming conventions for Python identifiers −

* Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
* Starting an identifier with a single leading underscore indicates that the identifier is private.
* Starting an identifier with two leading underscores indicates a strongly private identifier.
* If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

## *Standard Data Types*

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

# Flask

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

What is Web Framework?

Web Application Framework or simply Web Framework represents a collection of libraries and modules that enables a web application developer to write applications without having to bother about low-level details such as protocols, thread management etc.

What is Flask?

Flask is a web application framework written in Python. It is developed by Armin Ronacher, who leads an international group of Python enthusiasts named Pocco. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

A **Flask** application is started by calling the **run()** method. However, while the application is under development, it should be restarted manually for each change in the code. To avoid this inconvenience, enable **debug support**. The server will then reload itself if the code changes. It will also provide a useful debugger to track the errors if any, in the application.

The **Debug** mode is enabled by setting the **debug** property of the **application** object to **True** before running or passing the debug parameter to the **run()** method.

app.debug = True

app.run()

app.run(debug = True)

**Development**

Top-down approaches emphasize planning and a complete understanding of the system. It is inherent that no coding can begin until a sufficient level of detail has been reached in the design of at least some part of system. Programming, actually writing software code is just one part of the process, which is why people prefer to be called developers rather than merely programmers.

Top-down programming is a programming style, the mainstay of traditional procedural languages, in which design begins by specifying complex task and then dividing them into separate modules. Eventually, the components are specific enough to be coded and the program is written.

The waterfall model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Initiation, Analysis, Design, Code, Testing, Production/Implementation and Maintenance.

*Advantages of waterfall model:*

* It is the simplest software process model in terms of complexity and ease of implementation.
* This model is extremely easy to understand and therefore, is implemented at various project management levels
* It employs a systematic method of project development and delivery. So Waterfall model is chosen to develop this system.

**SYSTEM MAINTENANCE**

**Maintenance**

The term “Software Maintenance” is used to describe software engineering activities. Maintenance activities involve making enhancements to software products, adapting to new environments and correcting problems. Software product enhancements may involve providing new functional capabilities, improving user displays and nodes of interaction, upgrading external documents and internal documentation or upgrading the performance characteristics of a system. Adaptation of software to a new environment may involve moving the software to a different machine, or for instance, modifying the software to accommodate a new telecommunication protocol or an additional disk drives. Problem correction involves modification and revalidation of software to correct errors.

Many activities performed during software development enhance the maintainability of a software product. They are:-

**Analysis activities:**

The analysis phase of software development is concerned with determining customer requirements and constraints and establishing feasibility of the product.

* Develop standards and guidelines
* Set milestones for supporting documents
* Specify quality assurance procedures
* Identify likely product enhancements
* Determine resources required for maintenance
* Estimate maintenance costs

**Architectural Design Activities:**

* Emphasize clarity and modularity as design criteria
* Design to ease likely enhancement
* Use standardized notations to document, data flows, functions, structure and interconnections
* Observe the principles of information hiding, data abstraction and top-down hierarchical decomposition

**Detailed Design Activities**

* Use standardized notations to specify algorithms, data structures and procedure interface specifications
* Specify side effects and exception handling for each routine

**Implementation activities**

* Use single entry, single exit constructs
* Use standard indentation of constructs
* Use simple, clear coding style
* Use symbolic constants to parameterize routines
* Provide margins on resources
* Provide standard documentation
* Follow standard internal commenting guidelines

**Other activities:**

* Develop a maintenance guide
* Develop a test suite
* Provide test suite documentation

**SYSTEM Implementation**

Implementation is the process that actually yields the lowest-level system elements in the system hierarchy (system breakdown structure). The system elements are made, bought, or reused. Production involves the hardware fabrication processes of forming, removing, joining, and finishing; or the software realization processes of coding and testing; or the operational procedures development processes for operators' roles. If implementation involves a production process, a manufacturing system which uses the established technical and management processes may be required.

The purpose of the implementation process is to design and create (or fabricate) a system element conforming to that element’s design properties and/or requirements. The element is constructed employing appropriate technologies and industry practices. This process bridges the system definition processes and the integration process.

System Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage is achieving a successful system and in giving confidence on the new system for the user that it will work efficiently and effectively. The existing system was long time process.

The proposed system was developed using python. The existing system caused long time transmission process but the system developed now has a very good user-friendly tool, which has a menu-based interface, graphical interface for the end user. After coding and testing, the project is to be installed on the necessary system. The executable file is to be created and loaded in the system. Again the code is tested in the installed system. Installing the developed code in system in the form of executable file is implementation.

**QUALITY ASSURANCE**

Quality assurance comprises all those planned and systematic actions necessary to provide confidence that a structure, system or component will perform satisfactorily is service.

Quality assurance includes formal view of care, problem definition, corrective actions to remedy any deficiencies and evaluation of actions that to be taken.

The function of software quality that assures that the standards, processes, and procedures are appropriate for the project and are correctly implemented. This is an “umbrella activity” that is applied throughout the engineering process. Quality software is reasonably bug-free, delivered on time and within budget, meets requirements and/or expectations, and is maintainable.

The system is developed such that it ensures all the level of quality. It checks whether a user friendly environment is provided to the users and that there is a reliable, accurate and efficient flow of data within the system. The system also checks that due it contains the level of security required for the user. Hence as long as there is no hardware complaint, there is no problem with the software.

SYSTEM DESIGN

INPUT DESIGN

Input design is the process of converting user-originated inputs to a computer-based format. Input design is one of the most expensive phases of the operation of computerized system and is often the major problem of a system. Input design is a part of overall design, which requires careful attribute. Inaccurate input data are the most common cause of errors in data processing. The goal of designing input data is to make data entry as easy, logical and free from errors. In the system design phase input data are collected and organized in to groups of similar data.

**OUTPUT DESIGN**

Output design generally refers to the results and information that are generated by the system for many end-users; output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application. Computer output is the most important and direct source of information to the user. Output design is very important phase because the output will be in an interactive manner.

DATABASE DESIGN

The database design is a must for any application developed especially more for the data store projects. Since the chatting method involves storing the message in the table and produced to the sender and receiver, proper handling of the table is a must. In the project, login table is designed to be unique in accepting the username and the length of the username and password should be greater than zero. The different users view the data in different format according to the privileges given.

**FILE DESIGN**

At the highest level a ﬁle system is a way to organize, store, retrieve, and manage information on a permanent storage medium such as a disk. File systems manage permanent storage and form an integral part of all operating systems. There are many different approaches to the task of managing permanent storage. At one end of the spectrum are simple ﬁle systems that impose enough restrictions to inconvenience users and make using the ﬁle system difficult. At the other end of the spectrum are persistent object stores and object-oriented databases that abstract the whole notion of permanent storage so that the user and programmer never even need to be aware of it. The problem of storing, retrieving, and manipulating information on a computer is of a general-enough nature that there are many solutions to the problem.

The primary functionality that all ﬁle systems must provide is a way to store a named piece of data and to later retrieve that data using the name given to it. We often refer to a named piece of data as a ﬁle. A ﬁle provides only the most basic level of functionality in a ﬁle system. A ﬁle is where a program stores data permanently. In its simplest form a ﬁle stores a single piece of information. A piece of information can be a bit of text (e.g., a letter, program source code, etc.), a graphic image, a database, or any collection of bytes a user wishes to store permanently. The size of data stored may range from only a few bytes to the entire capacity of a volume. A ﬁle system should be able to hold a large number of ﬁles, where “large” ranges from tens of thousands to millions.

The name of a ﬁle is metadata because it is a piece of information about the ﬁle that is not in the stream of bytes that make up the ﬁle. There are several other pieces of metadata about a ﬁle as well—for example, the owner, security access controls, date of last modiﬁcation, creation time, and size. The ﬁle system needs a place to store this metadata in addition to storing the ﬁle contents.

**CODE DESIGN:**

The code design should be such that with less amount of coding we can achieve more results. The speed of the system will be more if the coding is less. Whether the data in the system is usable and readable by the system is depending on the coding.

In the project, the coding is being done such that proper validations are made to get the perfect input. No error inputs are accepted. In addition care is taken such that the data integrity and referential integrity is not violated in the database. In addition, coding is designed such that concurrency avoidance of accessing the database, limited user access to the table is made perfect.

Testing

**Introduction**

The most important phase in system development life cycle is system testing. The number and nature of errors in a newly designed system depends on the system specifications and the time frame given for the design.

A newly designed system should have all the subsystems working together, but in reality each subsystems work independently. During this phase, all the subsystems are gathered into one pool and tested to determine whether it meets the user requirements.

Testing is done at two level -Testing of individual modules and testing the entire system. During the system testing, the system is used experimentally to ensure that the software will run according to the specifications and in the way the user expects. Each test case is designed with the intent of finding errors in the way the system will process it.

Testing plays a very critical role in determining the reliability and efficiency of software and hence is a very important stage in software development. Software testing is done at different levels. They are the unit testing and system testing which comprises of integration testing and acceptance testing.

**TYPES OF TESTING**

**Unit Testing**

This is the first level of testing. The different modules are tested against the specifications produced during the integration. This is done to test the internal logic of each module. Those resulting from the interaction between modules are initially avoided. The input received and output generated is also tested to see whether it falls in the expected range of values. Unit testing is performed from the bottom up, starting with the smallest and lowest modules and proceeding one at a time.

The units in a system are the modules and routines that are assembled and integrated to perform a specific function. The programs are tested for correctness of logic applied and detection of errors in coding. Each of the modules was tested and errors are rectified. They were then found to function properly.

**Integration Testing**

In integration testing, the tested modules are combined into sub-systems, which are then tested. The goal of integration testing to check whether the modules can be integrated properly emphasizing on the interfaces between modules. The different modules were linked together and integration testing done on them.

**Validation Testing**

The objective of the validation test is to tell the user about the validity and reliability of the system. It verifies whether the system operates as specified and the integrity of important data is maintained. User motivation is very important for the successful performance of the system.

All the modules were tested individually using both test data and live data. After each module was ascertained that it was working correctly and it had been "integrated" with the system. Again the system was tested as a whole. We hold the system tested with different types of users. The System Design, Data Flow Diagrams, procedures etc. were well documented so that the system can be easily maintained and upgraded by any computer professional at a later

**System Testing**

The integration of each module in the system is checked during this level of testing. The objective of system testing is to check if the software meets its requirements. System testing is done to uncover errors that were not found in earlier tests. This includes forced system failures and validation of total system as the user in the operational environment implements it. Under this testing, low volumes of transactions are generally based on live data. This volume is increased until the maximum level for each transactions type is reached. The total system is also tested for recovery after various major failures to ensure that no data are lost during the breakdown.

**BIBILIOGRPHY**

* Python Essential Reference: Developer's Library Paperback – 9 Jul 2009 by David Beazley (Author)
* Python Programming: An Introduction to Computer Science 2nd Edition by John Zelle (Author)
* Automate the Boring Stuff with Python: Practical Programming for Total Beginners 1st Edition by Al Sweigart (Author)
* Python Programming for the Absolute Beginner, 3rd Edition 3rd Edition by Michael Dawson (Author)
* Python Programming for Beginners: An Introduction to the Python Computer Language and Computer Programming 1st Edition by Jason Cannon (Author)
* Python: Python Programming For Beginners - The Comprehensive Guide To Python Programming: Computer Programming, Computer Language, Computer Science Paperback – August 9, 2016 by Adam Stark (Author)

Websites

<https://www.w3schools.com/python/>

<https://en.wikipedia.org/wiki/Python_(programming_language)>

<https://www.codecademy.com/learn/learn-python>

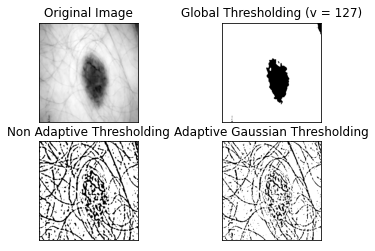
<https://www.learnpython.org/>

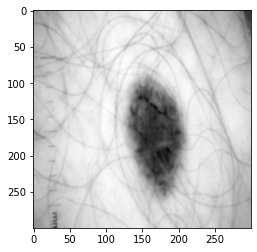
**Conclusion**

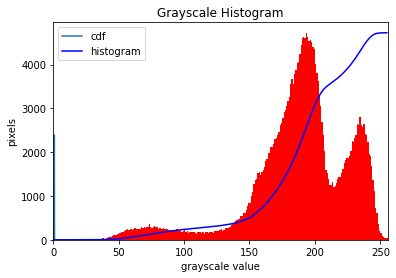
The results of our study underscore the efficacy of our proposed comprehensive approach in advancing skin disease diagnosis and classification systems. By integrating convolutional neural networks (CNNs) with vision transformer models, adaptive Gaussian thresholding for denoising, and the Sobel algorithm for segmentation, we have demonstrated significant improvements in accuracy, precision, recall, and F1 score compared to traditional methods. Our model's ability to learn hierarchical representations, capture spatial patterns, model long-range dependencies, and focus on relevant regions within images has led to robust and reliable classification results. These findings highlight the potential of deep learning and image processing fusion techniques in revolutionizing dermatological diagnostic systems, paving the way for more efficient and accurate healthcare interventions for skin diseases. Future research directions may include exploring additional advanced deep learning architectures, incorporating multi-modal data sources, and conducting clinical validation studies to further enhance the clinical utility of automated dermatological diagnosis systems.

**Reference**

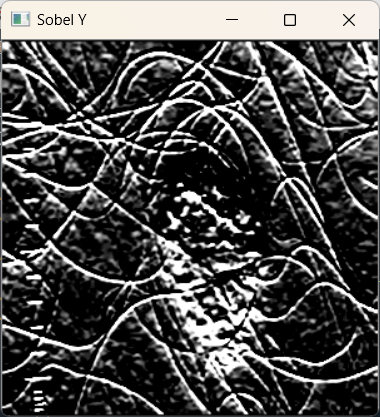
1. Adegun, A., & Viriri, S. (2020). Deep learning techniques for skin lesion analysis and melanoma cancer detection: a survey of state-of-the-art. Artificial Intelligence Review. doi:10.1007/s10462-020-09865-y
2. Ali, M. S., Miah, M. S., Haque, J., Rahman, M. M., & Islam, M. K. (2021). An enhanced technique of skin cancer classification using deep convolutional neural network with transfer learning models. Machine Learning with Applications, 5, 100036. doi:10.1016/j.mlwa.2021.100036
3. Dascalu, A., & David, E. O. (2019). Skin cancer detection by deep learning and sound analysis algorithms: A prospective clinical study of an elementary dermoscope. EBioMedicine. doi:10.1016/j.ebiom.2019.04.055
4. Han, S. S., Park, I., Eun Chang, S., Lim, W., Kim, M. S., Park, G. H., … Na, J.-I. (2020). Augmented Intelligence Dermatology: Deep Neural Networks Empower Medical Professionals in Diagnosing Skin Cancer and Predicting Treatment Options for 134 Skin Disorders. Journal of Investigative Dermatology. doi:10.1016/j.jid.2020.01.019
5. Hasan, M., Barman, S. D., Islam, S., & Reza, A. W. (2019). Skin Cancer Detection Using Convolutional Neural Network. Proceedings of the 2019 5th International Conference on Computing and Artificial Intelligence - ICCAI ’19. doi:10.1145/3330482.3330525
6. Hekler, A., Utikal, J. S., Enk, A. H., Hauschild, A., Weichenthal, M., Maron, R. C., … Schadendorf, D. (2019). Superior skin cancer classification by the combination of human and artificial intelligence. European Journal of Cancer, 120, 114–121. doi:10.1016/j.ejca.2019.07.019
7. Khanna, A., Gupta, D., Pólkowski, Z., Bhattacharyya, S., & Castillo, O. (Eds.). (2021). Data Analytics and Management. Lecture Notes on Data Engineering and Communications Technologies. doi:10.1007/978-981-15-8335-3
8. Mijwil, M. M. (2021). Skin cancer disease images classification using deep learning solutions. Multimedia Tools and Applications. doi:10.1007/s11042-021-10952-7
9. Monika, M. K., Arun Vignesh, N., Usha Kumari, C., Kumar, M. N. V. S. S., & Lydia, E. L. (2020). Skin cancer detection and classification using machine learning. Materials Today: Proceedings. doi:10.1016/j.matpr.2020.07.366
10. Murugan, A., Nair, S. A. H., Preethi, A. A. P., & Kumar, K. P. S. (2021). Diagnosis of skin cancer using machine learning techniques. Microprocessors and Microsystems, 81, 103727. doi:10.1016/j.micpro.2020.103727
11. Saba, T. (2020). Recent advancement in cancer detection using machine learning: Systematic survey of decades, comparisons and challenges. Journal of Infection and Public Health. doi:10.1016/j.jiph.2020.06.033
12. Tan, T. Y., Zhang, L., & Lim, C. P. (2019). Intelligent skin cancer diagnosis using improved particle swarm optimization and deep learning models. Applied Soft Computing, 105725. doi:10.1016/j.asoc.2019.105725
13. Thurnhofer-Hemsi, K., & Domínguez, E. (2020). A Convolutional Neural Network Framework for Accurate Skin Cancer Detection. Neural Processing Letters. doi:10.1007/s11063-020-10364-y

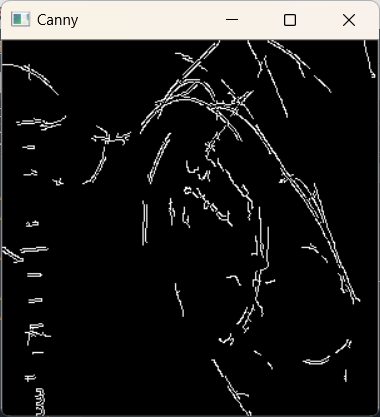


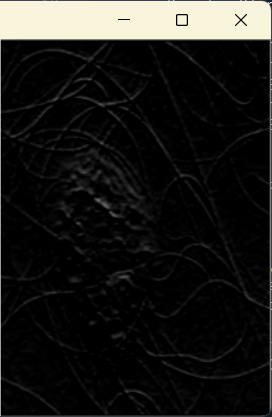








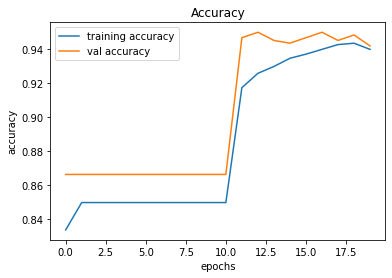
\



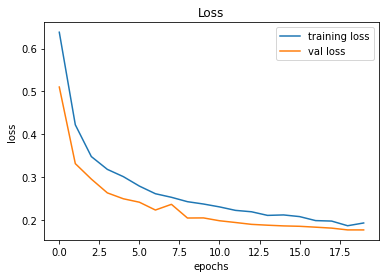
Root Mean Squared Error (RMSE): 8.730965964122564

Peak Signal-to-Noise Ratio (PSNR): 26.981650176017617

Structural Similarity Index (SSIM): 0.9923103506465271



\



Epoch 1/20

78/78 [==============================] - 4s 46ms/step - loss: 0.6381 - accuracy: 0.8334 - val\_loss: 0.5101 - val\_accuracy: 0.8661

Epoch 2/20

78/78 [==============================] - 3s 43ms/step - loss: 0.4220 - accuracy: 0.8495 - val\_loss: 0.3312 - val\_accuracy: 0.8661

Epoch 3/20

78/78 [==============================] - 3s 42ms/step - loss: 0.3477 - accuracy: 0.8495 - val\_loss: 0.2952 - val\_accuracy: 0.8661

Epoch 4/20

78/78 [==============================] - 4s 47ms/step - loss: 0.3178 - accuracy: 0.8495 - val\_loss: 0.2630 - val\_accuracy: 0.8661

Epoch 5/20

78/78 [==============================] - 4s 46ms/step - loss: 0.3008 - accuracy: 0.8495 - val\_loss: 0.2492 - val\_accuracy: 0.8661

Epoch 6/20

78/78 [==============================] - 3s 45ms/step - loss: 0.2788 - accuracy: 0.8495 - val\_loss: 0.2411 - val\_accuracy: 0.8661

Epoch 7/20

78/78 [==============================] - 4s 52ms/step - loss: 0.2610 - accuracy: 0.8495 - val\_loss: 0.2229 - val\_accuracy: 0.8661

Epoch 8/20

78/78 [==============================] - 4s 46ms/step - loss: 0.2526 - accuracy: 0.8495 - val\_loss: 0.2362 - val\_accuracy: 0.8661

Epoch 9/20

78/78 [==============================] - 4s 57ms/step - loss: 0.2424 - accuracy: 0.8495 - val\_loss: 0.2042 - val\_accuracy: 0.8661

Epoch 10/20

78/78 [==============================] - 4s 54ms/step - loss: 0.2369 - accuracy: 0.8495 - val\_loss: 0.2045 - val\_accuracy: 0.8661

Epoch 11/20

78/78 [==============================] - 3s 43ms/step - loss: 0.2302 - accuracy: 0.8495 - val\_loss: 0.1979 - val\_accuracy: 0.8661

Epoch 12/20

78/78 [==============================] - 3s 44ms/step - loss: 0.2221 - accuracy: 0.9173 - val\_loss: 0.1938 - val\_accuracy: 0.9468

Epoch 13/20

78/78 [==============================] - 3s 42ms/step - loss: 0.2188 - accuracy: 0.9258 - val\_loss: 0.1894 - val\_accuracy: 0.9500

Epoch 14/20

78/78 [==============================] - 3s 42ms/step - loss: 0.2104 - accuracy: 0.9298 - val\_loss: 0.1875 - val\_accuracy: 0.9452

Epoch 15/20

78/78 [==============================] - 3s 39ms/step - loss: 0.2114 - accuracy: 0.9347 - val\_loss: 0.1858 - val\_accuracy: 0.9435

Epoch 16/20

78/78 [==============================] - 3s 40ms/step - loss: 0.2075 - accuracy: 0.9371 - val\_loss: 0.1849 - val\_accuracy: 0.9468

Epoch 17/20

78/78 [==============================] - 3s 42ms/step - loss: 0.1982 - accuracy: 0.9399 - val\_loss: 0.1828 - val\_accuracy: 0.9500

Epoch 18/20

78/78 [==============================] - 3s 40ms/step - loss: 0.1971 - accuracy: 0.9427 - val\_loss: 0.1806 - val\_accuracy: 0.9452

Epoch 19/20

78/78 [==============================] - 3s 40ms/step - loss: 0.1862 - accuracy: 0.9435 - val\_loss: 0.1765 - val\_accuracy: 0.9484

Epoch 20/20

78/78 [==============================] - 3s 41ms/step - loss: 0.1927 - accuracy: 0.9399 - val\_loss: 0.1764 - val\_accuracy: 0.9419

Test loss: 0.1764451265335083

Test accuracy: 0.9419354796409607

**Code**

import cv2

import csv

import collections

import numpy as np

from tracker import \*

fr="dataset/test/benign/melanoma\_9608.jpg"

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread(fr,0)

img = cv2.medianBlur(img,5)

ret,th1 = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)

th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_MEAN\_C,\

cv2.THRESH\_BINARY,11,2)

th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,\

cv2.THRESH\_BINARY,11,2)

titles = ['Original Image', 'Global Thresholding (v = 127)',

'Non Adaptive Thresholding', 'Adaptive Gaussian Thresholding']

images = [img, th1, th2, th3]

for i in range(4):

plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')

plt.title(titles[i])

plt.xticks([]),plt.yticks([])

plt.show()

# Initialize Tracker

tracker = EuclideanDistTracker()

import numpy as np

import cv2

from matplotlib import pyplot as plt

import numpy as np

import glob

import matplotlib.pyplot as plt

import skimage.io

import skimage.color

import skimage.filters

img = cv2.imread(fr)

dst = cv2.fastNlMeansDenoisingColored(img,None,10,10,7,21)

plt.subplot(121),plt.imshow(img)

plt.subplot(122),plt.imshow(dst)

plt.show()

gray\_image = skimage.color.rgb2gray(img)

# blur the image to denoise

blurred\_image = skimage.filters.gaussian(gray\_image, sigma=1.0)

fig, ax = plt.subplots()

plt.imshow(blurred\_image, cmap="gray")

histogram, bin\_edges = np.histogram(blurred\_image, bins=256, range=(0.0, 1.0))

fig, ax = plt.subplots()

plt.plot(bin\_edges[0:-1], histogram)

plt.title("Grayscale Histogram")

plt.xlabel("grayscale value")

plt.ylabel("pixels")

plt.xlim(0, 1.0)

import numpy as np

import cv2 as cv

import os

from matplotlib import pyplot as plt

img = cv.imread(fr)

hist,bins = np.histogram(img.flatten(),256,[0,256])

cdf = hist.cumsum()

cdf\_normalized = cdf \* float(hist.max()) / cdf.max()

plt.plot(cdf\_normalized, color = 'b')

plt.hist(img.flatten(),256,[0,256], color = 'r')

plt.xlim([0,256])

plt.legend(('cdf','histogram'), loc = 'upper left')

plt.show()

from PIL import Image

img = cv.imread(fr,0)

# create a CLAHE object (Arguments are optional).

clahe = cv.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))

cl1 = clahe.apply(img)

cv.imwrite('clahe\_2.jpg',cl1)

im = Image.open("clahe\_2.jpg")

im.show()

import cv2

import numpy as np

img = cv2.imread(fr)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

img\_gaussian = cv2.GaussianBlur(gray,(3,3),0)

#canny

img\_canny = cv2.Canny(img,100,200)

#sobel

img\_sobelx = cv2.Sobel(img\_gaussian,cv2.CV\_8U,1,0,ksize=5)

img\_sobely = cv2.Sobel(img\_gaussian,cv2.CV\_8U,0,1,ksize=5)

img\_sobel = img\_sobelx + img\_sobely

#prewitt

kernelx = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])

kernely = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])

img\_prewittx = cv2.filter2D(img\_gaussian, -1, kernelx)

img\_prewitty = cv2.filter2D(img\_gaussian, -1, kernely)

cv2.imshow("Original Image", img)

cv2.imshow("Canny", img\_canny)

cv2.imshow("Sobel X", img\_sobelx)

cv2.imshow("Sobel Y", img\_sobely)

cv2.imshow("Sobel", img\_sobel)

cv2.imshow("Prewitt X", img\_prewittx)

cv2.imshow("Prewitt Y", img\_prewitty)

cv2.imshow("Prewitt", img\_prewittx + img\_prewitty)

cv2.waitKey(0)

cv2.destroyAllWindows()

from skimage.exposure import histogram

hist, hist\_centers = histogram(img)

#Plotting the Image and the Histogram of gray values

fig, axes = plt.subplots(1, 2, figsize=(8, 3))

axes[0].imshow(img, cmap=plt.cm.gray)

axes[0].axis('off')

axes[1].plot(hist\_centers, hist, lw=2)

axes[1].set\_title('histogram of gray values')

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load the image

image\_path = fr # Replace with the actual path to your image

image = cv2.imread(image\_path)

image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

\_, thresholded\_image = cv2.threshold(gray\_image, 120, 255, cv2.THRESH\_BINARY)

contours, \_ = cv2.findContours(thresholded\_image, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

mask = np.zeros\_like(image\_rgb)

cv2.drawContours(mask, contours, -1, (255, 255, 255), thickness=cv2.FILLED)

segmented\_image = cv2.bitwise\_and(image\_rgb, mask)

plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)

plt.imshow(image\_rgb)

plt.title('Original Image')

plt.subplot(1, 3, 2)

plt.imshow(thresholded\_image, cmap='gray')

plt.title('Afected Area')

plt.subplot(1, 3, 3)

plt.imshow(segmented\_image)

plt.title('Segmented Image')

plt.show()

import cv2

import numpy as np

import matplotlib.pyplot as plt

image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

\_, thresholded\_image = cv2.threshold(gray\_image, 120, 255, cv2.THRESH\_BINARY)

contours, \_ = cv2.findContours(thresholded\_image, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

mask = np.zeros\_like(image\_rgb)

cv2.drawContours(mask, contours, -1, (255, 255, 255), thickness=cv2.FILLED)

segmented\_image = cv2.bitwise\_and(image\_rgb, mask)

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load image

image = cv2.imread(fr)

# Convert image to RGB (matplotlib uses RGB)

image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

# Create a mask initialized with zeros for GrabCut

mask = np.zeros(image.shape[:2], np.uint8)

# Define the background and foreground models for GrabCut

bgdModel = np.zeros((1, 65), np.float64)

fgdModel = np.zeros((1, 65), np.float64)

# Define the rectangle enclosing the region of interest (the skin lesion)

rect = (50, 50, image.shape[1] - 100, image.shape[0] - 100)

# Apply GrabCut algorithm to segment the skin lesion

cv2.grabCut(image, mask, rect, bgdModel, fgdModel, 5, cv2.GC\_INIT\_WITH\_RECT)

# Create a mask where probable background and definite background are 0, others are 1

mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')

# Apply the mask to the original image to extract the segmented region

segmented = image \* mask2[:, :, np.newaxis]

# Plot images

fig, axs = plt.subplots(1, 2, figsize=(10, 5))

axs[0].imshow(image\_rgb)

axs[0].set\_title('Original Image')

axs[0].axis('off')

axs[1].imshow(cv2.cvtColor(segmented, cv2.COLOR\_BGR2RGB))

axs[1].set\_title('Segmented Image')

axs[1].axis('off')

plt.show()

import cv2

from skimage.metrics import structural\_similarity as ssim

import numpy as np

# Load ground truth image

ground\_truth = cv2.imread(fr)

segmented\_image = segmented\_image

# Convert images to grayscale (if needed)

ground\_truth\_gray = cv2.cvtColor(ground\_truth, cv2.COLOR\_BGR2GRAY)

segmented\_gray = cv2.cvtColor(segmented\_image, cv2.COLOR\_BGR2GRAY)

# Compute Root Mean Squared Error (RMSE)

rmse = np.sqrt(((ground\_truth\_gray - segmented\_gray) \*\* 2).mean())

# Compute Peak Signal-to-Noise Ratio (PSNR)

psnr = cv2.PSNR(ground\_truth\_gray, segmented\_gray)

# Compute Structural Similarity Index (SSIM)

ssim\_index, \_ = ssim(ground\_truth\_gray, segmented\_gray, full=True)

print("Root Mean Squared Error (RMSE):", rmse)

print("Peak Signal-to-Noise Ratio (PSNR):", psnr)

print("Structural Similarity Index (SSIM):", ssim\_index)

import tensorflow as tf

from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Dropout, concatenate, UpSampling2D

import numpy as np

import matplotlib.pyplot as plt

from torchvision import transforms, datasets

from torch.utils.data import DataLoader

import torch.nn as nn

import random

random.seed(0)

np.random.seed(0)

tf.random.set\_seed(0)

# Define a double convolution block

class BasicBlock(nn.Module):

def \_\_init\_\_(self, in\_channels, out\_channels, stride=1):

super(BasicBlock, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(in\_channels, out\_channels, kernel\_size=3, stride=stride, padding=1, bias=False)

self.bn1 = nn.BatchNorm2d(out\_channels)

self.relu = nn.ReLU(inplace=True)

self.conv2 = nn.Conv2d(out\_channels, out\_channels, kernel\_size=3, stride=1, padding=1, bias=False)

self.bn2 = nn.BatchNorm2d(out\_channels)

self.stride = stride

def forward(self, x):

residual = x

out = self.conv1(x)

out = self.bn1(out)

out = self.relu(out)

out = self.conv2(out)

out = self.bn2(out)

if self.stride != 1 or x.size(1) != out.size(1):

residual = nn.Conv2d(x.size(1), out.size(1), kernel\_size=1, stride=self.stride, bias=False)(x)

out += residual

out = self.relu(out)

return out

def double\_conv(in\_channels, out\_channels):

return nn.Sequential(

nn.Conv2d(in\_channels, out\_channels, kernel\_size=3, padding=1),

nn.BatchNorm2d(out\_channels),

nn.ReLU(inplace=True),

nn.Conv2d(out\_channels, out\_channels, kernel\_size=3, padding=1),

nn.BatchNorm2d(out\_channels),

nn.ReLU(inplace=True)

)

# Define the loss function

def dice\_loss(pred, target, smooth=1.):

intersection = (pred \* target).sum()

dice = (2. \* intersection + smooth)/(pred.sum() + target.sum() + smooth)

return 1. - dice

# Define the performance metrics

def iou\_score(pred, target, smooth=1.):

intersection = (pred & target).sum().float()

union = (pred | target).sum().float()

iou = (intersection + smooth) / (union + smooth)

return iou

def dice\_coef(pred, target, smooth=1.):

intersection = (pred \* target).sum()

dice = (2. \* intersection + smooth)/(pred.sum() + target.sum() + smooth)

return dice

from tensorflow import keras

from tensorflow.keras import layers

IMG\_HEIGHT = 256

IMG\_WIDTH = 256

import torch

import torch.nn as nn

import torchvision

from einops import rearrange

# Vision Transformer Model

class VisionTransformer(nn.Module):

def \_\_init\_\_(self, image\_size, patch\_size, num\_classes, dim, depth, heads, mlp\_dim, channels=3):

super(VisionTransformer, self).\_\_init\_\_()

assert image\_size % patch\_size == 0, "Image dimensions must be divisible by the patch size."

num\_patches = (image\_size // patch\_size) \*\* 2

patch\_dim = channels \* patch\_size \*\* 2

# Patch embedding layer

self.patch\_embedding = nn.Conv2d(channels, dim, kernel\_size=patch\_size, stride=patch\_size)

self.cls\_token = nn.Parameter(torch.randn(1, 1, dim))

# Positional embedding

self.positional\_embedding = nn.Parameter(torch.randn(num\_patches + 1, dim))

# Transformer Encoder

self.transformer\_encoder = nn.TransformerEncoder(nn.TransformerEncoderLayer(

d\_model=dim, nhead=heads, dim\_feedforward=mlp\_dim), num\_layers=depth)

# Classifier head

self.fc = nn.Linear(dim, num\_classes)

def forward(self, x):

# Patch embedding

x = self.patch\_embedding(x)

x = rearrange(x, 'b c h w -> b (h w) c') # Flatten the spatial dimensions into patches

x = torch.cat((self.cls\_token.expand(x.size(0), -1, -1), x), dim=1) # Add classification token

# Add positional embeddings

x += self.positional\_embedding

x = x.permute(1, 0, 2) # Transpose for transformer input

# Transformer encoder

x = self.transformer\_encoder(x)

# Classification head

x = x.mean(dim=0) # Take mean over tokens

x = self.fc(x)

return x

# Example usage

image\_size = 224

patch\_size = 16

num\_classes = 1000

dim = 512

depth = 6

heads = 8

mlp\_dim = 2048

# Instantiate Vision Transformer model

vit\_model = VisionTransformer(image\_size, patch\_size, num\_classes, dim, depth, heads, mlp\_dim)

# Example input

input\_tensor = torch.randn(1, 3, image\_size, image\_size) # Batch size 1, 3 channels (RGB), image size 224x224

# Forward pass

output = vit\_model(input\_tensor)

print("Output shape:", output.shape)

import numpy as np

import matplotlib.pyplot as plt

import cv2

import os

import tensorflow as tf

from PIL import Image

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.utils import normalize, to\_categorical

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Activation, Dropout, Flatten, Dense

# ## Load Data

image\_directory='dataset/test/'

no\_tumor\_images=os.listdir(image\_directory+ 'benign/')

yes\_tumor\_images=os.listdir(image\_directory+ 'malignant/')

print('Train: ', len(no\_tumor\_images))

print('Test: ',len(yes\_tumor\_images))

dataset=[]

label=[]

INPUT\_SIZE=64

# ## Create labels

for i , image\_name in enumerate(no\_tumor\_images):

if(image\_name.split('.')[1]=='jpg'):

image=cv2.imread(image\_directory+'benign/'+image\_name)

image=Image.fromarray(image,'RGB')

image=image.resize((INPUT\_SIZE,INPUT\_SIZE))

dataset.append(np.array(image))

label.append(0)

for i , image\_name in enumerate(yes\_tumor\_images):

if(image\_name.split('.')[1]=='jpg'):

image=cv2.imread(image\_directory+'malignant/'+image\_name)

image=Image.fromarray(image, 'RGB')

image=image.resize((INPUT\_SIZE,INPUT\_SIZE))

dataset.append(np.array(image))

label.append(1)

dataset=np.array(dataset)

label=np.array(label)

print('Dataset: ',len(dataset))

print('Label: ',len(label))

# ## Train-Test Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(dataset, label, test\_size=0.20, random\_state=2523)

# ## Normalize the Data

X\_train = normalize(X\_train, axis=1)

X\_test = normalize(X\_test, axis=1)

vit\_model = VisionTransformer(image\_size, patch\_size, num\_classes, dim, depth, heads, mlp\_dim)

model=Sequential()

model.add(Conv2D(32, (3,3),activation='relu', input\_shape=(INPUT\_SIZE, INPUT\_SIZE, 3)))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Flatten())

model.add(Dense(64,activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(1,activation='sigmoid'))

model.add(Dense(64,activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(1,activation='sigmoid'))

model.compile(optimizer=tf.keras.optimizers.Adam(lr=1e-4), loss=dice\_loss, metrics=[dice\_coef])

model.summary()

model.compile(loss='binary\_crossentropy',optimizer='adam', metrics=['accuracy'])

from tensorflow.keras.utils import plot\_model

plot\_model(model, to\_file='model\_architecture.png', show\_shapes=True, show\_layer\_names=True)

history=model.fit(X\_train, y\_train,

batch\_size=32,

verbose=1, epochs=20,

validation\_data=(X\_test, y\_test),

shuffle=False)

plt.figure(0)

plt.plot(history.history['accuracy'], label='training accuracy')

plt.plot(history.history['val\_accuracy'], label='val accuracy')

plt.title('Accuracy')

plt.xlabel('epochs')

plt.ylabel('accuracy')

plt.legend()

plt.show()

plt.figure(1)

plt.plot(history.history['loss'], label='training loss')

plt.plot(history.history['val\_loss'], label='val loss')

plt.title('Loss')

plt.xlabel('epochs')

plt.ylabel('loss')

plt.legend()

plt.show()

# ## Save the Model

model.save('model.h5')

# ## Load Model

model = load\_model('model.h5')

model\_json = model.to\_json()

with open('model\_architecture.json', 'w') as f:

f.write(model\_json)

# ## Make Prediction on New Data

#show\_result('1.png')

model.load\_weights('model.h5')

y\_pred = model.predict(X\_test)

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report

y\_pred = model.predict(X\_test)

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

acc = accuracy\_score(y\_test, y\_pred.round())

scores = model.evaluate(X\_test, y\_test, verbose=0)

print('Test loss:', scores[0])

print('Test accuracy:', scores[1])