AUTOMATED DETECTION OF WHITE BLOOD CELL CANCER DISEASES

Submitted in partial fulfillment for the award of the degree of

Master of Technology

in

Internet of Things and Sensor Systems

by

SURENDAR.D

(22MTS0002)

Under the guidance of

Prof. / Dr. PRAKASH R

SCHOOL OF ELECTRONICS ENGINEERING

VIT, Vellore.

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

May,2024

DECLARATION

I here by declare that the thesis entitled "AUTOMATED DETECTION OF WHITE BLOOD CELL CANCER DISEASES" submitted by me, for the award of the degree of Master of Technology in Internet of Things and Sensor Systems to VIT is a record of bonafide work carried out by me under the supervision of PRAKASH R

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date: /05/2024 **Signature of the Candidate**

CERTIFICATE

This is to certify that the thesis entitled “AUTOMATED DETECTION OF WHITE BLOOD CELL CANCER DISEASES” submitted by Surendar.D & 22MTS0002, School of Electronics Engineering, VIT, for the award of the degree of *Master of Technology in Internet of Things and Sensor Systems*, is a record of bonafide work carried out by him / her under my supervision during the period, 03. 01. 2024 to 05.05.2024, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date :

Signature of the Guide

Internal Examiner

External Examiner

Department of Sensor & Biomedical Technology

Internet of Things and Sensor Systems

ACKNOWLEDGEMENTS

It is my pleasure to express with deep sense of gratitude to Dr. PRAKASH R, Associate Professor Sr., School of Electronics Engineering, Vellore Institute of Technology, for his/her constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him / her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of image processing.

I would like to express my gratitude to Dr. G. Viswanathan, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr. G.V.Selvam, Dr. V. S. Kanchana Bhaaskaran, Dr. Partha Sharathi Mallick, and Dr. Sivanantham S, School of Electronics Engineering, for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Head of the Department Dr. VIDHYA S, Associate Professor Sr. all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Vellore **surendar.D**

Date: /05/2024 Name of the student

Executive Summary

In this proposed system we tend to introduce and produce an enhanced application that helps in diagnosing process of white blood cell disease which is an enhancement of the existing approach. The main objective is to find and classify white blood cell cancer diseases. Our planned increased system is meant better higher results than the prevailing system in such some way that to classify all sub forms of illness. Using MATLAB we tend to show the results that are higher than the prevailing system with the performance analysis parameters of detection and classification accuracy. The proposed methodology is valid and results are going to be compared with previously planned techniques, which shows that planned techniques are going to be reliably applied for prior disclosure of white blood cells cancer disorder. The proposed system presents the application that helps in diagnosing process white blood cell disease. These illnesses are separated into two groups: leukemia and myeloma. Similar symptoms and illnesses that can be difficult to diagnose are included in each area. By computing distinct features, this system is applied to one of the two disease groups. Lastly KNN, SVM, RF and CNN are applied the final decision. The proposed approach aims to early discovery of white blood cell cancer, reduce the misdiagnosis cases in addition to improving the system learning methodology.

CONTENTS Page no.

CHAPTER 4 SYSTEM

REQUIREMENTS

CHAPTER 5

SYSTEM DESIGN

SPECIFICATIONS

CHAPTER 6

SYSTEM DESIGN IMPLEMENTATIONS

CHAPTER 7

PROJECT

DEMONSTRATION

CHAPTER 8

CONCLUSION &

FUTURE WORK

CHAPTER 9

SOCIAL/ENVIRONMENTAL IMPACT

CHAPTER 10

SUMMARY

APPENDIX A

APPENDIX B

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ALL	ACUTE LYMPHOBLASTIC LEUKEMIA
AML	ACUTE MYELOID LEUKEMIA
CB	COEFFICIENT BLUE
CR	COEFFICIENT RED
CNN	CONVOLUTIONAL NEURAL NETWORK
KNN	K-NEAREST NEIGHBOUR
SVM	SUPPORT VECTOR MACHINE
YCBCR	LUMA & BLUE MINUS & RED MINUS
GLCM	GRAY-LEVEL CO-OCCURRENCE MATRIX

Chapter 1

Introduction

1.1OBJECTIVE

The principal aim of this undertaking is to acquire a comprehensive understanding and practical proficiency in the application of Random Forest, CNN, SVM, and KNN (K-Nearest Neighbors) are four neural network architectures in the identification and classification of diseases related to white blood cell cancer. The focus extends to the accurate categorization of various subforms within white blood cell cancers.

1.2OVERVIEW

The project overview centers around the classification of white blood cell cancer. We present an innovative four-step approach for the automatic detection of white blood cell cancer diseases, demonstrating superior accuracy when compared to existing methods.

1.3INTRODUCTION

Image processing involves the transformation of an image into a digital format, allowing for a variety of manipulations to improve the image or extract useful data. This technology uses well-established signal processing techniques to handle images as two-dimensional signals and get the necessary results. The field is rapidly advancing and finds applications across various business domains.

Image processing encompasses both analog and digital methods. Analog or visual techniques are applicable to hard copies such as printouts and photographs, with image analysts relying on interpretative fundamentals. The process is not limited to the specific area under study but extends to the knowledge and experience of the analyst. Association plays a crucial role in visual techniques, where analysts combine personal knowledge with collateral data for effective image processing.

Digital processing techniques, on the other hand, leverage computers to manipulate digital images. Raw data from imaging sensors, especially in satellite platforms, often contains deficiencies. To address these issues and ensure data accuracy, digital processing involves three key phases: pre-processing, enhancement and display, and information extraction. In the medical field, digital image information is crucial for enhancing the diagnostic capabilities of various diseases.

White blood cell cancer comprises two main types: Leukemia and Myeloma. Leukemia is characterized by the production of abnormal white blood cells in the bone marrow. Subtypes include Acute Myeloid Leukemia (AML) with further categorization into (M1, M2) and Acute Lymphoblastic Leukemia (ALL) with subtypes (L1, L2). Myeloma is another form of cancer originating from plasma cells in the bone marrow.

This work introduces a hybrid automated system designed to facilitate the diagnosis of different white blood cell cancer diseases, encompassing Leukemia subtypes (AML, ALL) and Myeloma. Two main approaches are employed to reduce confusion in discriminating between certain subtypes, ultimately enhancing the accuracy of diagnosis.

TYPES	SUB LABELS
AML	M1, M2
ALL	L1, L2

Table 1.3 – types of white blood cell cancer

Chapter 2

Literature Survey

2.1 RELATED WORK

[1] TITLE: Feature Extraction of White Blood Cells Using CMYK-Moment Localization and Deep Learning in Acute Myeloid Leukemia Blood Smear Microscopic Images.
AUTHOR : TUSNEEM AHMED M. ELHASSAN 1 , MOHD SHAFRY MOHD.

JOURNAL NAME AND YEAR: IEEE DIGITAL OBJECT IDENTIFIER
ACCESS , 2022.

DESCRIPTION

The authors highlight the inherent complexities and potential errors in the conventional AML diagnosis process, emphasizing the need for a more efficient and accurate solution. They propose a Computer-Aided Diagnosis (CAD) system, leveraging the capabilities of deep learning for White Blood Cells (WBC) detection – a crucial step in AML diagnosis. Recognizing the significance of accurate feature extraction in training pixel-wise classification models, the paper introduces a novel hybrid method that combines image processing techniques with deep learning approaches. This integration aims to enhance the quality of features used in the training process, ultimately improving the accuracy of WBC detection.Despite the advancements, the paper acknowledges a limitation in the speed of Convolutional Neural Networks (CNNs) due to operations like maxpooling. This insight provides valuable information for further research and optimization of deep learning models in the context of AML diagnosis.In summary, the study contributes to the field by presenting an innovative approach to feature extraction in the detection of white blood cells, shedding light on both its strengths and limitations for future research endeavors.

[2] TITLE: Detection of Blood Cancer-Leukemia using K-means Algorithm.

AUTHORS: SUDHARSHAN DUTH .P

JOURNAL NAME AND YEAR: IEEE TRANSACTIONS IN INTELLIGENT COMPUTING AND CONTROL SYSTEMS, 2021.

DESCRIPTION

a significant exploration of the challenges posed by blood cancer, specifically Leukemia, which remains a leading cause of human mortality. The effectiveness of treatment largely hinges on early detection and diagnosis, with Leukemia arising from the excessive production of abnormal white blood cells in the bone marrow. Hematologists undertake microscopic studies of human blood samples, employing methodologies like microscopic color imaging, image segmentation, clustering, and classification. These techniques prove crucial in the easy identification of individuals afflicted with this disease. The utilization of microscopic imaging allows for a diverse range of methods to detect blood cancer, particularly in visible and immature white blood cells. Recognizing Leukemia early on is paramount for healthcare practitioners, enabling them to administer prompt and appropriate treatments. The initial stage involves segmentation, where white blood cells are meticulously separated from other blood components. However, it is important to acknowledge certain limitations in the process, such as the dependence on initial values and the manual selection of the k value, both of which warrant consideration for future advancements in this domain.

[3] TITLE: Fuzzy based Blood Image Segmentation for Automated Leukemia Detection.
AUTHORS: SUBRAJEET MOHAPATRA, SUSHANTA.

JOURNAL NAME AND YEAR: IEEE TRANSACTIONS IN DIGITAL OBJECT IDENTIFIER, 2021 .

DESCRIPTION

the critical challenge of diagnosing Acute Lymphoblastic Leukemia (ALL), a predominant hematological neoplasia affecting children. Characterized by an excess of lymphoid blasts in the bloodstream, ALL accounts for approximately 80% of childhood leukemia cases, primarily occurring in the age range of 3-7 years. The ambiguity in the signs and symptoms of ALL often leads to misdiagnosis, compounded by the potential imitation of similar symptoms by other disorders. Accurate diagnosis traditionally relies on the meticulous microscopic examination of stained blood smears or bone marrow aspirates. While specific tests like fluorescence in situ hybridization (FISH), immunophenotyping, cytogenetic analysis, and cytochemistry aid in leukemia detection, their time-consuming and costly nature necessitates the quest for more efficient approaches. The authors advocate for the utilization of image analysis as a low-cost and effective solution for the automated detection of leukemia, providing a quantitative examination of stained blood microscopic images. However, it is crucial to note the acknowledged limitation, namely, the training complexity of Support Vector Machine (SVM), which is highly dependent on the size of the dataset.

[4] TITLE: Detection of White Blood Cell Cancer using Image Processing.
AUTHORS: ROHIT AGRAWAL

JOURNAL NAME AND YEAR: IEEE TRANSACTIONS IN INFORMATION SCIENCE, 2021.

DESCRIPTION

presents a pivotal contribution to the challenging task of identifying White Blood Cell (WBC) cancer diseases, including Acute Myeloid Leukemia (AML), Acute Lymphoblastic Leukemia (ALL), and Myeloma. Recognizing the complexity of these diseases due to their sudden onset, the paper introduces an innovative automated system designed to assist medical professionals in accurately diagnosing various types and subtypes of WBC cancer. The proposed methodology involves utilizing microscopic blood images as input, with a dataset comprising 100 images, of which 62 are allocated for training and 38 for testing. The images undergo conversion to the YCbCr format for segmentation, employing a combination of Gaussian Distribution and Otsu Adaptive Thresholding. Clustering is achieved through the K-Means method. Feature extraction, crucial for subsequent classification, is conducted using the Gray Level Co-occurrence Matrix (GLCM), and Convolutional Neural Network (CNN) serves as the classification tool. Despite these advancements, the methodology is acknowledged to lack spatial invariance to input data, suggesting areas for potential improvement.

[5] TITLE: White Blood Cell Classification and Counting Using Convolutional Neural Network.
AUTHORS: MERL JAMES MACAWILE, VONN VINCENT QUIÑONES.

JOURNAL NAME AND YEAR: IEEE TRANSACTIONS IN ROBOTICS ENGINEERING, 2021.

DESCRIPTION
This research underscores the pivotal importance of accurately counting and categorizing leukocytes, or White Blood Cells (WBCs), in the assessment and detection of various illnesses affecting the immune system. Conditions such as infections, anemia, leukemia, cancer, and AIDS (Acquired Immune Deficiency Syndrome) are among the health issues where an accurate enumeration of WBCs proves crucial. Traditionally, WBC counting involves the use of hematology analyzers and manual counting methods. However, in the era of modernization, the authors delve into the burgeoning field of image processing, incorporating diverse segmentation and classification techniques. Their aim is to provide alternative and more efficient approaches to WBC classification and counting.

[6] TITLE: Automated Detection of White Blood Cells Cancer Disease.
AUTHORS: TIBOR GRASSER, KARINA ROTT, HANS REISINGER

JOURNAL NAME AND YEAR: IEEE TRANSACTIONS IN DETECTION USING DEEP LEARNING, 2021.

DESCRIPTION

The authors categorize the diseases into two groups, each encompassing disorders with similar symptoms that pose challenges in diagnosis. Their innovative approach involves the implementation of two distinct methodologies, each tailored to one of the disease categories based on the physician's selection. These approaches compute different features, and a final decision is reached using a Random Forest classifier. The primary goal is the early discovery of white blood cell cancers, mitigating misdiagnosis cases, and refining the system's learning methodology. Notably, the proposed system leaves the final tuning of results in the hands of experts.

[7] TITLE: Automated Detection Of White Blood Cells Cancer Diseases

AUTHORS: HEND MOHAMED, ROWAN OMAR, NERMEEN SAEED, ALI ESSAM, NADA AYMAN, TARAGGY MOHIY AND ASHRAF ABDELRAOUF.

JOURNAL NAME AND YEAR: IEEE TRANSACTIONS, 2018.

DESCRIPTION

Automated diagnosis of white blood cell cancer diseases, notably Leukemia and Myeloma, represents a complex frontier in biomedical research. This study introduces a groundbreaking application, marking a paradigm shift in diagnostic methodologies. The diseases are stratified into two categories, each housing disorders with overlapping symptoms, posing diagnostic challenges. The system deploys two distinct approaches, chosen based on the physician's discretion, which compute diverse features and employ a Random Forest classifier for the final decision. The overarching objective is the timely detection of white blood cell cancers, reducing misdiagnosis rates, and refining the system's learning methodology. A noteworthy aspect is the provision for expert intervention, allowing professionals to fine-tune results. The proposed methodology showcases a remarkable accuracy of 93% in the first category and 95% in the second. This research contributes to the evolving landscape of automated diagnostic systems, particularly in the context of white blood cell cancers.

Chapter 3

System Analysis

3.1 EXISTING SYSTEM

The existing system incorporates a Convolutional Neural Network (CNN) meticulously designed and trained for robust feature extraction. Specifically, the architecture leverages ResNet50, a convolutional neural network renowned for its proficiency in analyzing visual images. This sophisticated network is instrumental in capturing intricate features from the dataset. To optimize the matching process within the dataset, an Ant Colony Optimization method is deployed, strategically navigating through the data to identify the most suitable class.

Furthermore, the system employs Contrast Limited Adaptive Histogram Equalization (CLAHE) to enhance image quality. Unlike conventional Adaptive Histogram Equalization (AHE), CLAHE addresses concerns related to contrast over-amplification, ensuring a nuanced and balanced improvement in image contrast. This refined approach to feature extraction and optimization techniques underscores the system's commitment to achieving heightened accuracy and performance in image analysis and classification.

CLASSIFICATION METHODS	PRE PROCESSING METHODS
CNN	Ant Colony Optimization
RCNN	Acute Lymphoblastic image database
Deep Learning Framework	Ant Colony Optimization

Table 3.1 – Existing Systems

3.1.1 DISADVANTAGE

The present system falls short of automation, necessitating the provision of two input parameters for its operation, introducing a manual aspect that may hinder seamless functionality.

An extensive training dataset is a prerequisite for the system's optimal performance. Additionally, the system encounters challenges in accurately decoding positional information, underscoring the need for enhancements in spatial understanding.

The convergence speed of the system is notably sluggish, adversely affecting its responsiveness and efficiency in arriving at conclusive results. This underscores a potential area for improvement in terms of algorithmic optimization.

The system exhibits a suboptimal level of accuracy in its classifications, signaling a scope for refinement to enhance its precision and reliability in determining outcomes.

The processing time for input images is protracted, suggesting a potential inefficiency in computational workflows. Addressing this prolonged processing time could contribute to a more streamlined and expeditious system operation.

3.2 PROPOSED SYSTEM

In the envisioned system, a meticulously designed framework is proposed for the creation, implementation, and assessment of an automated tool aimed at precisely identifying diseases related to white blood cells. Functioning as a recognition system, the approach systematically processes acquired blood microscopic images through distinct stages. Initially, during preprocessing, the system enhances image quality by employing YCBCR color conversion, where luminance (Y), blue minus luminance (CB), and red minus luminance (CR) components are manipulated. This step aims to improve the overall representation of the images.

Moving forward, in the segmentation stage, the system meticulously isolates pertinent components through cell segmentation, a crucial step for subsequent analysis. Feature extraction follows, encompassing the retrieval of diverse features such as texture, morphological attributes, statistical metrics, and size ratios. This comprehensive feature set plays a pivotal role in characterizing the visual information embedded in the images.

The classification stages introduce a variety of classifiers, including K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Random Forest (RF), and Convolutional Neural Network (CNN). This amalgamation of classifiers contributes to a robust and multifaceted approach, enabling accurate categorization based on the extracted features. The proposed system integrates advanced image processing, feature extraction methodologies, and a diverse set of classifiers to elevate the precision and reliability of the detection process for diseases related to white blood cells.

3.2.1 ADVANTAGES

Distinguishing itself from current systems, our innovative approach possesses the capacity to learn from past results, thereby augmenting the future accuracy and efficacy of the system.

Chapter 4

System Requirements

4.1 HARDWARE CONFIGURATION

Processor	:	i3 core
Hard Disk	:	10 GB
Speed	:	2.90 GHz
RAM	:	4 GB

4.2 SOFTWARE REQUIREMENTS

Operating system	:	WINDOWS 8/8.1/10/11, MAC.
Coding Language	:	MATLAB
Platform	:	MATLAB

4.3 TECHNOLOGIES USED

4.3.1 OVERVIEW OF MATLAB:

MATLAB stands out as a high-performance language tailored for technical computing, offering an integrated platform that seamlessly combines computation, visualization, and programming within an accessible environment. Here, problems and solutions are articulated in conventional mathematical notation, providing users with a user-friendly interface.

The versatile applications of MATLAB encompass a spectrum of functions, including:

Math and Computation: MATLAB excels in mathematical operations and computations, making it a powerful tool for tackling a myriad of technical challenges.
Algorithm Development: The platform supports the creation and refinement of algorithms, facilitating robust development processes.
Data Acquisition: MATLAB is adept at handling data acquisition, streamlining the collection and organization of diverse datasets.
Modeling, Simulation, and Prototyping: It serves as an invaluable resource for creating models, conducting simulations, and prototyping, crucial in diverse technical domains.
Data Analysis, Exploration, and Visualization: MATLAB's capabilities extend to in-depth data analysis, exploration, and vivid visualization, enhancing insights into complex datasets.
MATLAB operates as an interactive system, where arrays serve as the fundamental data element, eliminating the need for explicit dimensioning. This feature expedites problem-solving, particularly for challenges formulated with matrix and vector expressions. Contrasting with scalar non-interactive languages like C or Fortran, MATLAB's interactive nature significantly reduces programming time.

The nomenclature "MATLAB" originates from "matrix laboratory," reflecting its initial purpose of providing convenient access to matrix software developed by the LINPACK and EISPACK projects. Presently, MATLAB incorporates cutting-edge LAPACK and

BLAS libraries, establishing a contemporary standard in matrix computation.

This platform has evolved collaboratively, benefiting from input from a diverse user base. Widely adopted in academic settings, MATLAB serves as a fundamental instructional tool in mathematics, engineering, and science courses. In industrial contexts, it emerges as the preferred tool for efficient research, development, and analysis, epitomizing high productivity and versatility.

4.3.2 IMAGE PROCESSING TOOLBOX

The Image Processing Toolbox stands as a comprehensive resource, offering an extensive array of reference-standard algorithms and graphical tools tailored for image processing, analysis, visualization, and algorithm development. This versatile toolbox empowers users to engage in various image-related tasks, including but not limited to image enhancement, deblurring, feature detection, noise reduction, segmentation, spatial transformations, and image registration.

Noteworthy features of the Image Processing Toolbox include its support for multithreading, capitalizing on the capabilities of multicore and multiprocessor computers to enhance computational efficiency. The toolbox accommodates a diverse range of image types, encompassing high dynamic range, gigapixel resolution, ICC-compliant color, and tomographic images. This flexibility ensures adaptability to various imaging scenarios, making it a valuable asset for researchers and practitioners working with diverse datasets.

The graphical tools integrated into the toolbox provide a dynamic interface for users to explore images, scrutinize pixel regions, fine-tune contrast, generate contours or histograms, and manipulate regions of interest (ROIs). These interactive features contribute to a more intuitive and user-friendly experience, facilitating a deeper understanding and analysis of images.

The algorithms within the toolbox enable users to address a multitude of image-related challenges, such as restoring degraded images, detecting and measuring features, analyzing shapes and textures, and adjusting the color balance. This toolbox serves as an indispensable resource for professionals across different domains, offering a rich set of tools and functionalities for comprehensive image processing tasks.

4.3.3 APPLICATION OF MATLAB

The MATLAB toolbox for Statistics and Machine Learning proves invaluable for programmers, offering seamless implementation of statistical methods, both descriptive and inferential. This extends to the realm of machine learning, where diverse models can be effortlessly employed to address contemporary challenges. The toolbox's adaptability to big data applications enhances its utility, making it a versatile asset for data-driven solutions.

In the domain of Mapping, applications abound, with the MapReduce tool, particularly significant in Big Data scenarios. Its real-world applications span theft analysis, financial fraud detection, regression models, contingency analysis, predictive techniques in social media, and comprehensive data monitoring. The utility of data mapping extends across diverse industries, showcasing its relevance in modern analytics.

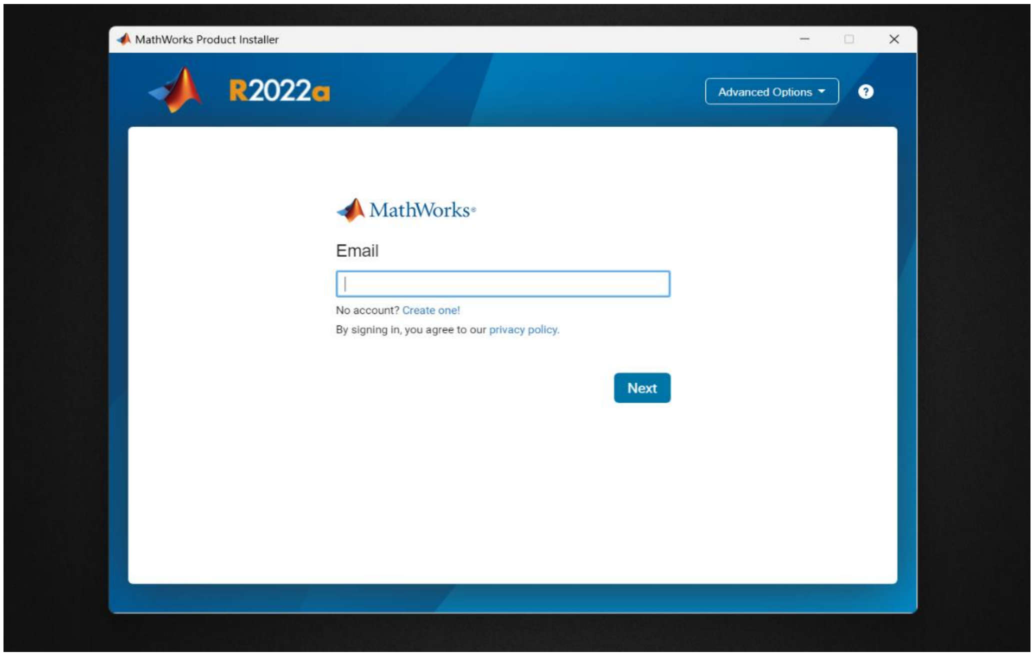
Deep Learning, a subset of machine learning, emerges as a powerful tool with applications in speech recognition, financial fraud detection, and medical image analysis. Leveraging tools such as time-series analysis, Artificial Neural Networks (ANN), Fuzzy logic, or combinations thereof further enhances its capabilities, allowing for intricate problem-solving across various domains.

The toolbox for Image Processing in MATLAB finds ubiquitous applications in our daily lives, ranging from commonplace barcode scanners to sophisticated selfie enhancements (such as face beauty and background blurring) and face detection algorithms. Additionally, it plays a pivotal role in crucial areas like satellite data transmission, employing algorithms to receive, decode, and enhance transmitted images. This underscores the toolbox's significance in diverse applications, making it an essential resource for image-related processing tasks.

4.3 HOW TO INSTALL MATLAB:

Acquiring MATLAB Software:
Visit the official MathWorks website (MathWorks Website) to obtain the MATLAB software.

Downloading MATLAB:
After logging in to your MathWorks account, navigate to the "Downloads" section and select the appropriate MATLAB version compatible with your operating system (Windows, macOS, or Linux).
Running the Installer:
Locate the downloaded MATLAB installer file and execute it. Follow the on-screen instructions to initiate the installation process.
Account Sign-In:
If prompted, sign in to your MathWorks account during the installation process for licensing purposes.



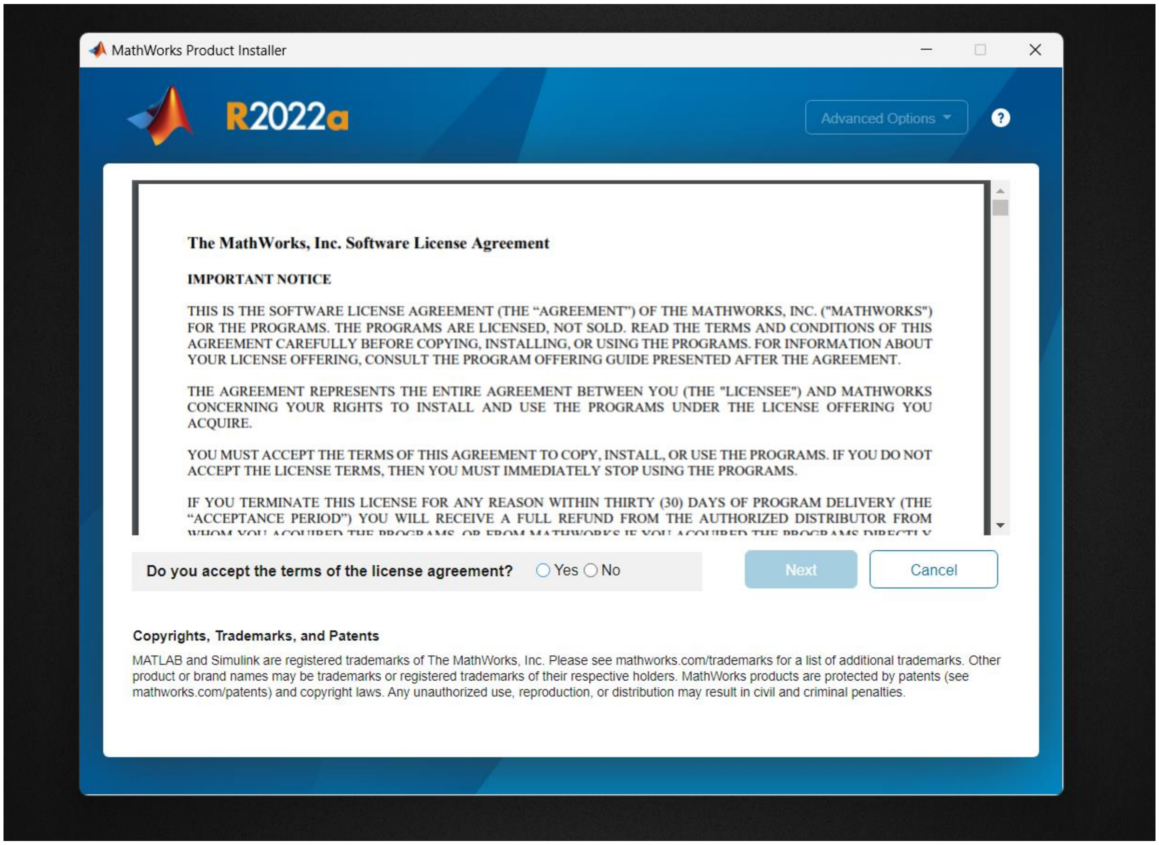
No account? Create one!
By signing in, you agree to our privacy policy.

Choosing Installation Type:

Opt for the "Install" option and select the installation type that aligns with your license (Typical, Custom, or Network).

Accepting License Agreement:

Carefully read and accept the terms of the license agreement presented during the installation.



The MathWorks, Inc. Software License Agreement

IMPORTANT NOTICE

THIS IS THE SOFTWARE LICENSE AGREEMENT (THE "AGREEMENT") OF THE MATHWORKS, INC. ("MATHWORKS") FOR THE PROGRAMS. THE PROGRAMS ARE LICENSED, NOT SOLD. READ THE TERMS AND CONDITIONS OF THIS AGREEMENT CAREFULLY BEFORE COPYING, INSTALLING, OR USING THE PROGRAMS. FOR INFORMATION ABOUT YOUR LICENSE OFFERING, CONSULT THE PROGRAM OFFERING GUIDE PRESENTED AFTER THE AGREEMENT.

THE AGREEMENT REPRESENTS THE ENTIRE AGREEMENT BETWEEN YOU (THE "LICENSEE") AND MATHWORKS CONCERNING YOUR RIGHTS TO INSTALL AND USE THE PROGRAMS UNDER THE LICENSE OFFERING YOU ACQUIRE.

YOU MUST ACCEPT THE TERMS OF THIS AGREEMENT TO COPY, INSTALL, OR USE THE PROGRAMS. IF YOU DO NOT ACCEPT THE LICENSE TERMS, THEN YOU MUST IMMEDIATELY STOP USING THE PROGRAMS.

IF YOU TERMINATE THIS LICENSE FOR ANY REASON WITHIN THIRTY (30) DAYS OF PROGRAM DELIVERY (THE "ACCEPTANCE PERIOD") YOU WILL RECEIVE A FULL REFUND FROM THE AUTHORIZED DISTRIBUTOR FROM WHOM YOU ACQUIRED THE PROGRAMS, OR FROM MATHWORKS IF YOU ACQUIRED THE PROGRAMS DIRECTLY

Do you accept the terms of the license agreement? ○ Yes ○ No

Copyrights, Trademarks, and Patents

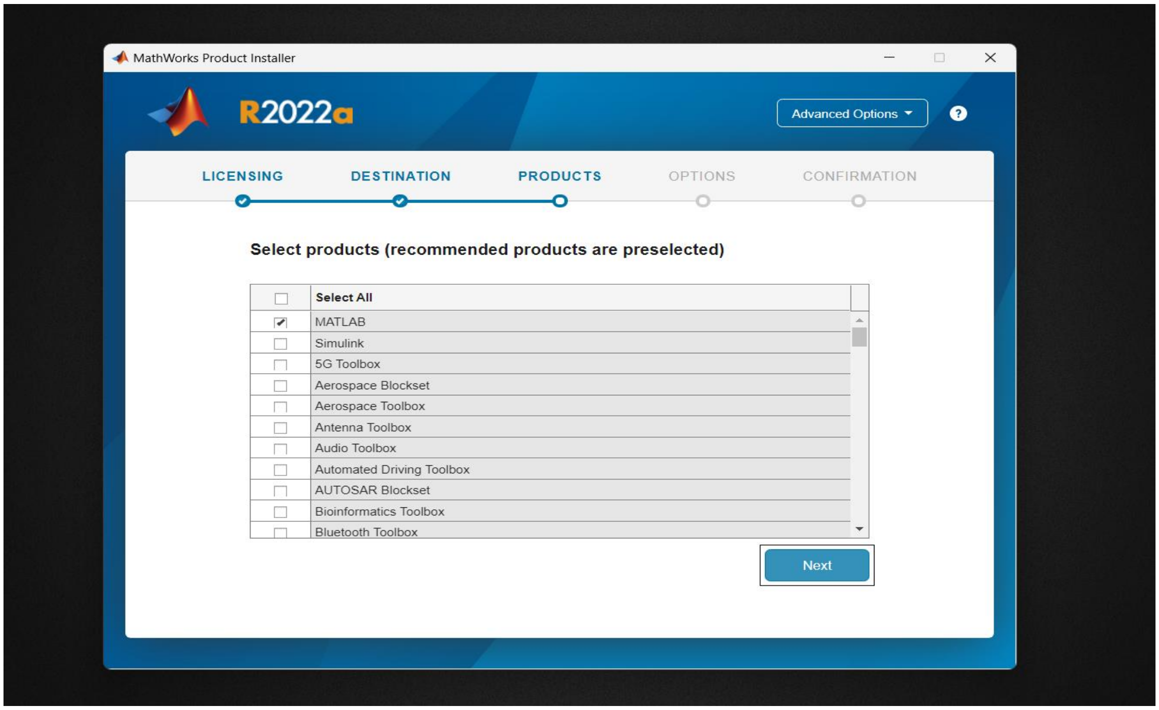
MATLAB and Simulink are registered trademarks of The MathWorks, Inc. Please see mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders. MathWorks products are protected by patents (see mathworks.com/patents) and copyright laws. Any unauthorized use, reproduction, or distribution may result in civil and criminal penalties.

Specifying Destination Folder:

Designate the installation folder for MATLAB according to your preferences.

Selecting Products:

For Custom installations, choose the specific MATLAB products and toolboxes required for your project.

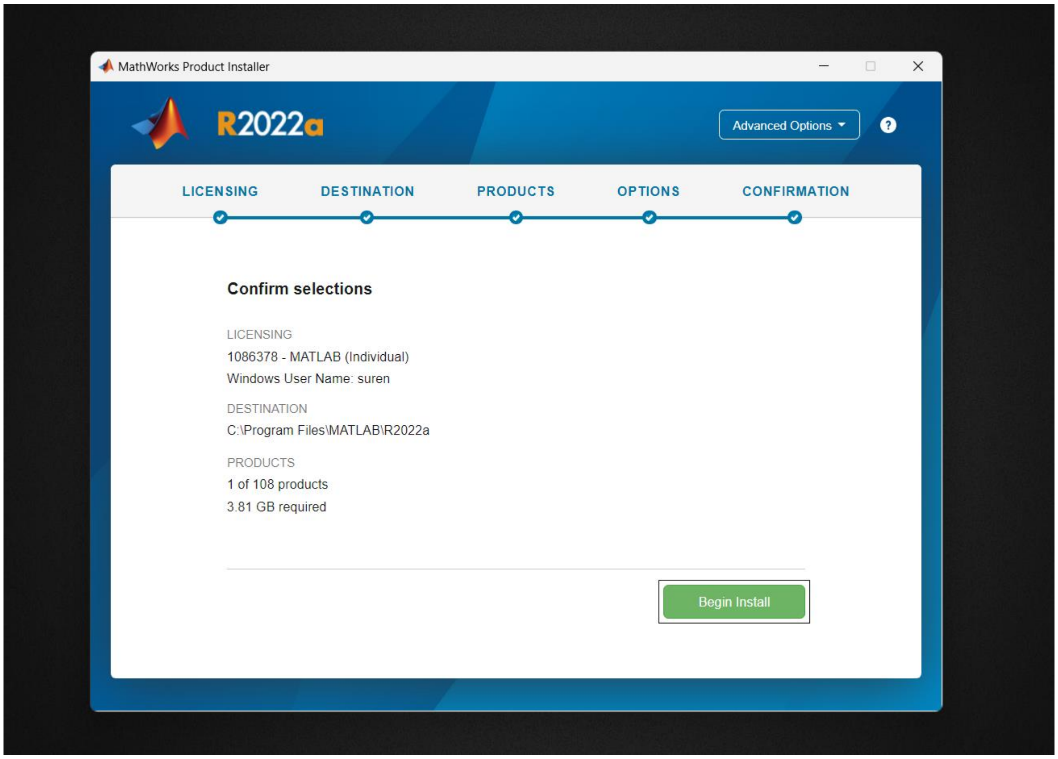


Select products (recommended products are preselected)

□ **Select All**
☑ MATLAB
□ Simulink
□ 5G Toolbox
□ Aerospace Blockset
□ Aerospace Toolbox
□ Antenna Toolbox
□ Audio Toolbox
□ Automated Driving Toolbox
□ AUTOSAR Blockset
□ Bioinformatics Toolbox
□ Bluetooth Toolbox

Commencing Installation:

Initiate the installation process and follow the prompts provided.



Confirm selections

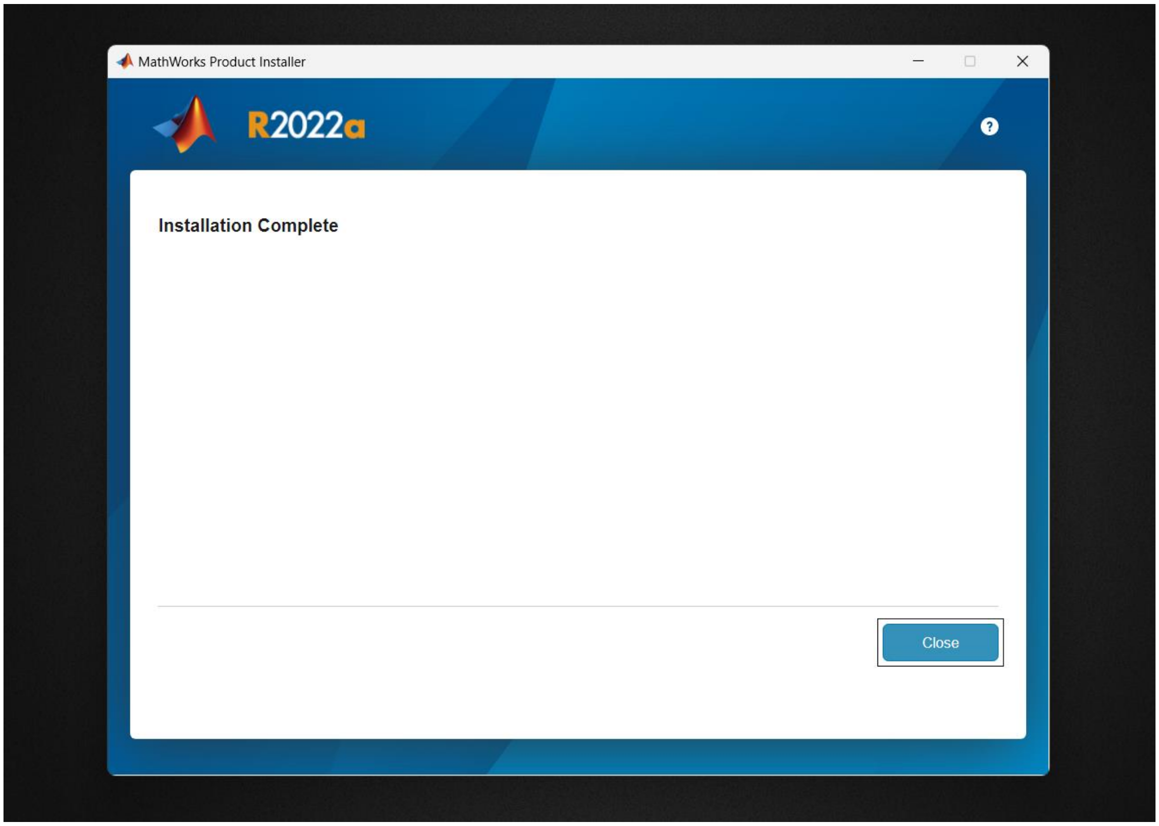
LICENSING
1086378 - MATLAB (Individual)
Windows User Name: suren

DESTINATION
C:\Program Files\MATLAB\R2022a

PRODUCTS
1 of 108 products
3.81 GB required

Completing Installation:

Finish the installation process and confirm that MATLAB launches successfully.



MathWorks Product Installer

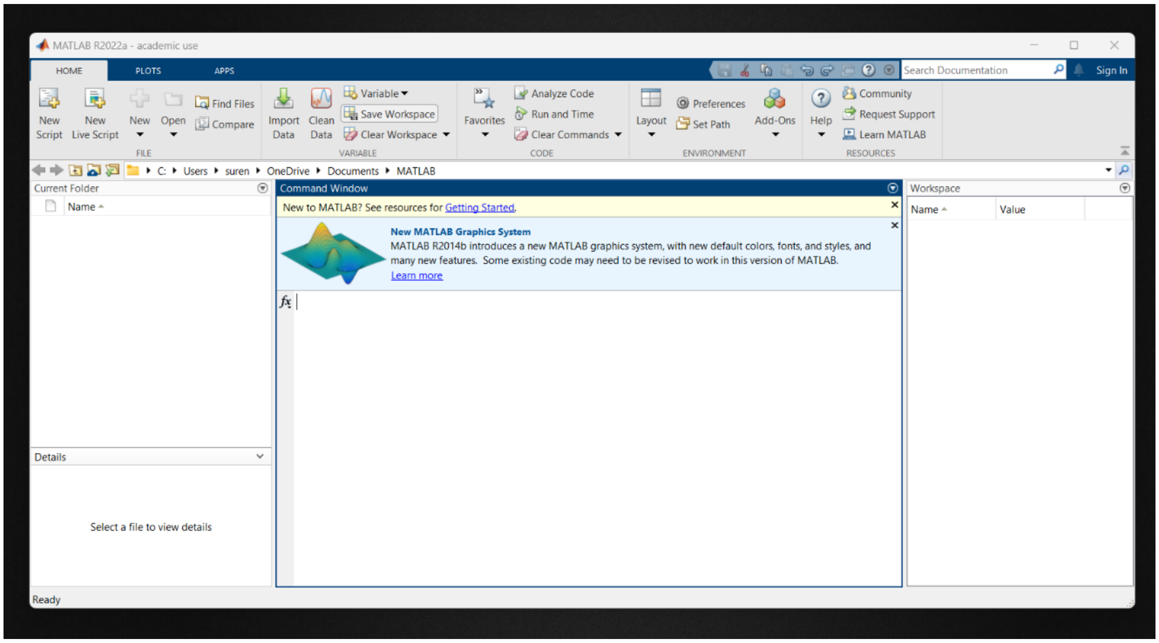
Installation Complete

Setting Up MATLAB Path:

If required, add MATLAB to your system's PATH variable to enable easy access.

Testing MATLAB:

Open MATLAB and execute a simple test script to ensure proper functionality.



New to MATLAB? See resources for Getting Started.

New MATLAB Graphics System

MATLAB R2014b introduces a new MATLAB graphics system, with new default colors, fonts, and styles, and many new features. Some existing code may need to be revised to work in this version of MATLAB.

Learn more

Select a file to view details

Ready

Chapter 5

System Design Specifications

5.1 ARCHITECTURE DIAGRAM

The input image undergoes a series of intricate processing steps, starting with the Preprocessing section, where it is transformed using the YCBCR converter to enhance color information. Subsequently, Noise Removal techniques are applied to refine the image quality. The image is then subjected to Normalization through the utilization of an Adaptive Threshold Algorithm, ensuring uniformity and enhancing contrast.

Moving to the Segmentation section, the image undergoes Initial Segmentation, where regions of interest are delineated. Further refinement occurs through Nucleus Extraction, isolating critical components. Additionally, Color Segmentation is performed using Gaussian distribution, enhancing the visual discrimination of different structures.

Following segmentation, the image enters the Feature Extraction section. Here, Gradient features are extracted using the Gray-Level Co-occurrence Matrix (GLCM), providing insight into texture patterns. Statistical and morphological features are then computed, contributing to a comprehensive representation of the image's characteristics.

Transitioning into the classifier section, various algorithms such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Random Forests (RF), and Convolutional Neural Networks (CNN) are employed. These classifiers play a pivotal role in accurately identifying classes and subtypes within the image data. The goal is to achieve superior results by leveraging the strengths of each classifier.

To gauge the effectiveness of the classification process, a thorough Performance Analysis is conducted within the Classifier section. This involves evaluating the accuracy, precision, recall, and F1 score of the classification results. By employing a multi-step approach and leveraging diverse classifiers, the system aims to achieve robust and accurate identification of classes and subtypes, ensuring a comprehensive analysis of the input image.

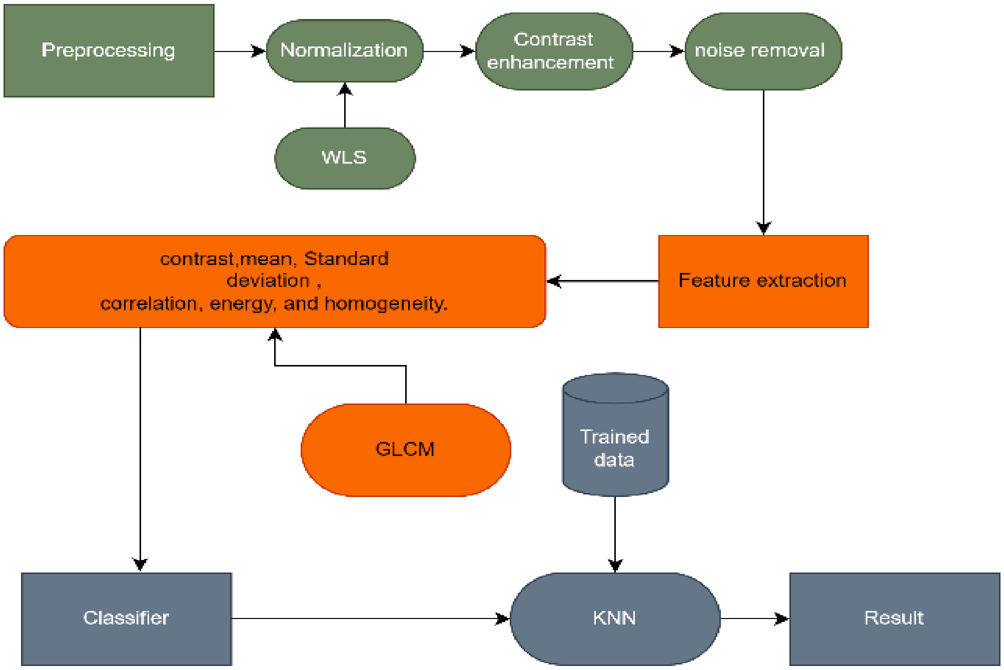


Fig 5.1 – Overall System Diagram of KNN

In the SVM algorithm, the features extracted during the preprocessing stage serve as the basis for training the model. Utilizing a set of training images, the SVM algorithm learns to distinguish patterns and relationships within the feature space. These features, derived from the preprocessed images, play a crucial role in training the SVM model to recognize relevant patterns associated with different classes or categories.

During the training phase, the SVM algorithm optimizes its parameters to create an effective decision boundary that separates different classes in the feature space. This boundary is designed to maximize the margin between classes, enhancing the model's generalization capabilities.

Once the SVM model is trained, it can be applied to classify new, unseen images. The features extracted from a test image are fed into the trained SVM model, which then makes a decision regarding the image's class or category. The SVM algorithm utilizes its learned decision boundary to classify the input features and assign the test image to the appropriate class.

This process ensures that the SVM model can generalize its learning from the training set to accurately classify new images based on the features extracted during the preprocessing stage. The success of the classification is contingent on the effectiveness of the SVM algorithm in creating a discriminative decision boundary during the training process.

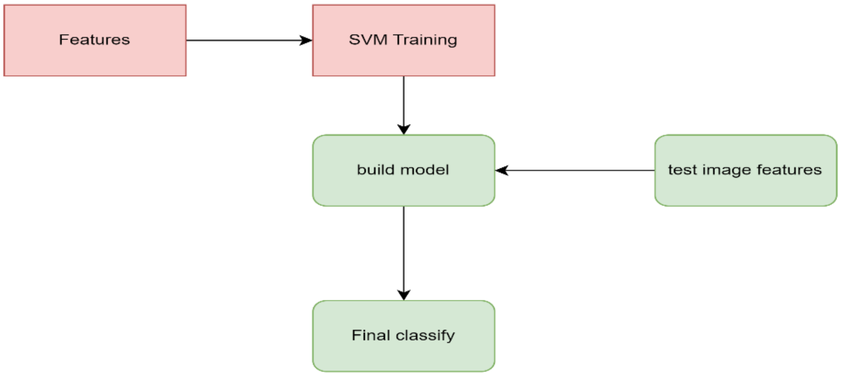


Fig 5.2 – Overall System Diagram of SVM

In the CNN algorithm, the features extracted during the preprocessing stage serve as input for building a connection mapping layer. These features, derived from the preprocessed images, are crucial for the CNN model to establish meaningful relationships and patterns within the image data.

The CNN architecture involves constructing multiple layers, including convolutional layers, pooling layers, and fully connected layers. The connection mapping layer, formed through these architectural elements, captures hierarchical representations of features from the input images.

During the training phase, the CNN model learns to recognize intricate patterns and spatial dependencies present in the features extracted from the training images. The convolutional layers perform operations such as feature detection and extraction, enabling the model to understand complex relationships within the image data.

Once the CNN model is trained, it can be applied to classify new, unseen images. The features extracted from a test image are input into the connection mapping layer, where the model processes these features through its learned network of connections. This process allows the CNN to capture intricate details and hierarchical representations, enabling it to make informed decisions regarding the class or category of the test image.

The success of the CNN algorithm lies in its ability to automatically learn and hierarchically represent features, allowing for robust image classification. By leveraging the interconnected layers within the model, CNNs excel at capturing spatial hierarchies, making them effective for tasks such as image recognition and classification.

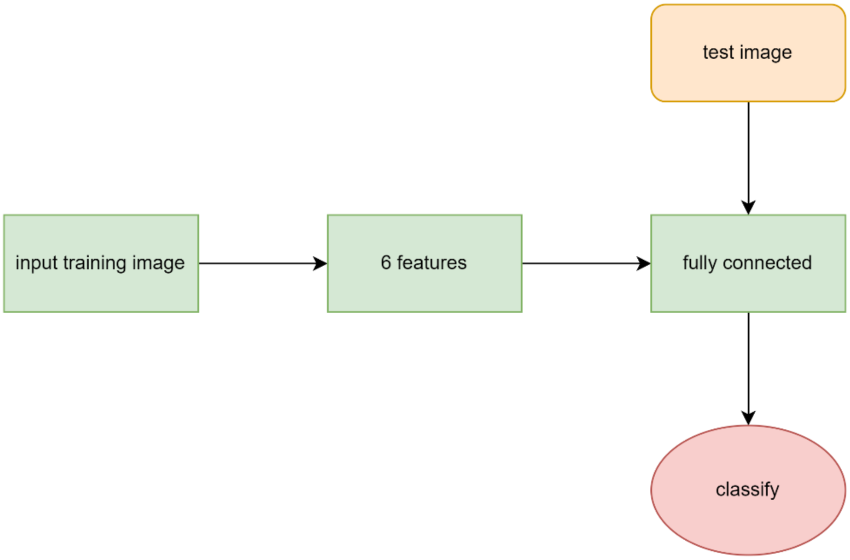


Fig 5.3 – Overall System Diagram of CNN

Within the Random Forest (RF) algorithm, the features extracted during the preprocessing stage serve as critical inputs for constructing decision trees. These features, derived from the preprocessed images, play a pivotal role in training the Random Forest model to discern intricate patterns within the image data.

The RF algorithm operates by creating an ensemble of decision trees during the training phase. Each decision tree within the forest is built using a subset of features, contributing to diversity and robustness in the overall model. The decision trees collectively form a powerful and adaptable framework for classification tasks.

During the training process, the RF algorithm optimizes each decision tree by evaluating different subsets of features at various nodes. This allows the model to capture diverse relationships and dependencies within the feature space. The final decision of the RF model is an aggregation of the decisions made by individual trees, often through voting or averaging.

When presented with a new, unseen image, the features extracted from the test image serve as input for the decision trees in the Random Forest. Each decision tree processes the input features through its nodes, ultimately classifying the test image based on the learned patterns and relationships.

The strength of the Random Forest algorithm lies in its ability to handle complex datasets, mitigate overfitting, and provide robust predictions. By leveraging the collective intelligence of multiple decision trees, Random Forests excel in capturing diverse feature interactions, making them effective for image classification tasks.

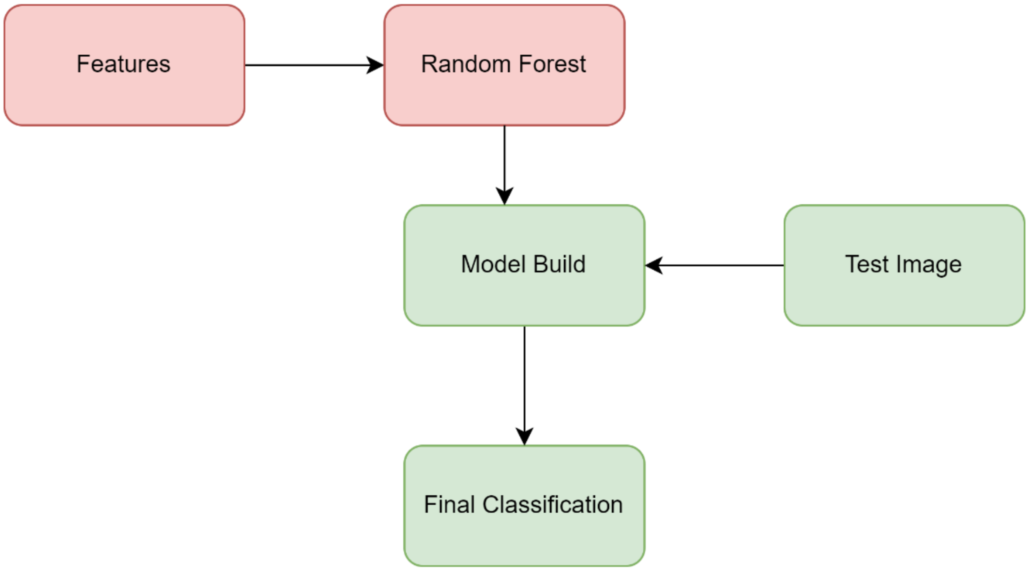


Fig 5.4 – Overall System Diagram of RF

5.2 DATA FLOW DIAGRAM

The data flow diagram visually illustrates the progression of data, commencing from the input image and undergoing a sequence of processing steps.

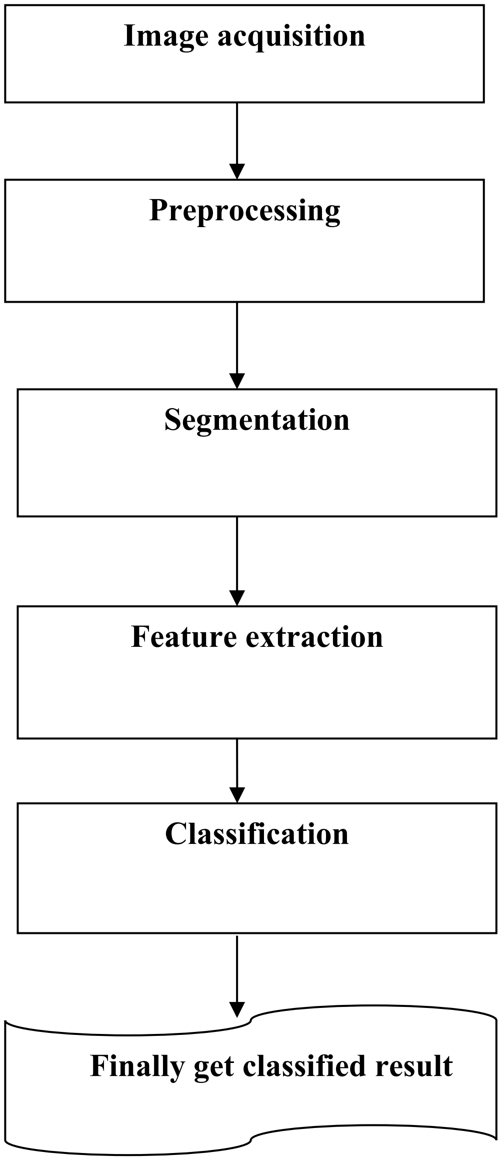


Fig 5.2.1– Data Flow Diagram

This multi-step process culminates in the final classification of the image with the assistance of four distinct algorithms: Random Forest (RF), Convolutional Neural Network (CNN), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). All these processes operate on the preprocessed image data to achieve accurate and comprehensive classification results.

5.3 USE CASE DIAGRAM

A use case diagram serves as a visual representation capturing the dynamic interactions within a system. This diagram encapsulates the functionality of the system by incorporating key elements such as use cases, actors, and their interrelationships.

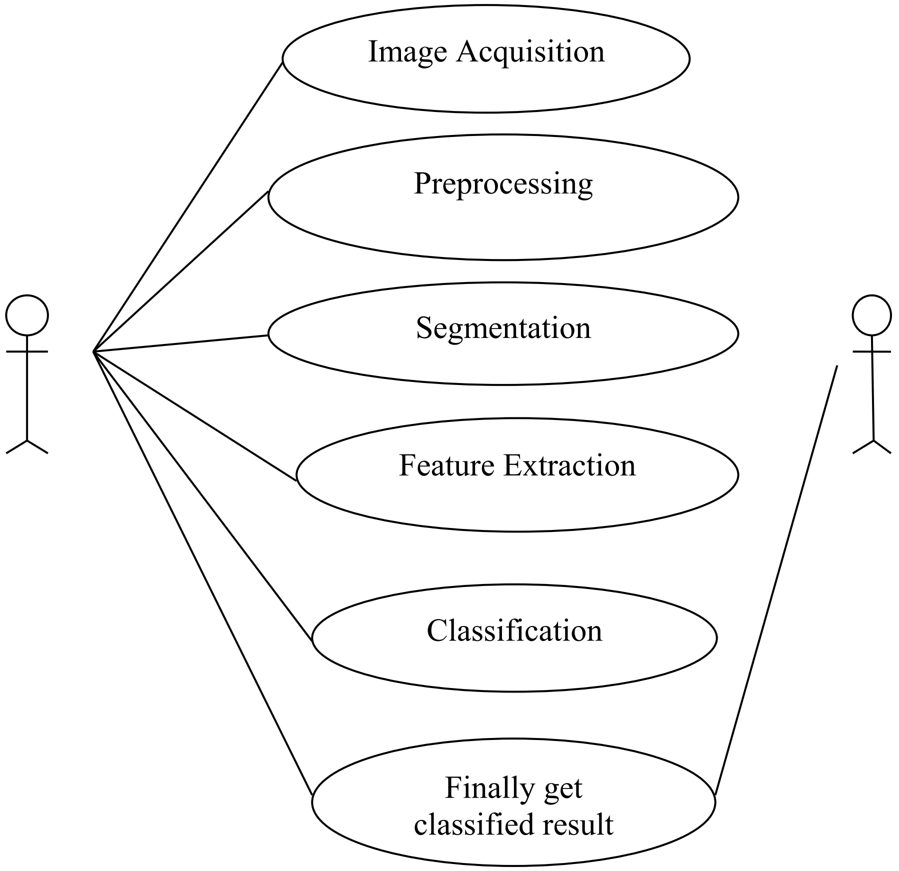


Fig 5.2.2 – Use case diagram

By modeling tasks, services, and functions, it provides a comprehensive overview of the system or subsystem within an application. This high-level depiction showcases how users interact with the system and illustrates the essential tasks and services that the system must perform.

The primary purpose of a use case diagram is to convey the dynamic aspects of a system. It effectively consolidates the system's requirements, encompassing both internal processes and external influences that shape its behavior. Overall, a use case diagram is a valuable tool for understanding and visualizing the functional aspects and user interactions within a complex system.

Chapter 6

System Design Implementation

6.1 IMPLEMENTATION

The implementation initiates with the input phase, where an image is provided as input. Subsequently, the image undergoes preprocessing before being categorized by Random Forest (RF), Convolutional Neural Network (CNN), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) to identify white blood cells. Finally, the input is classified into its corresponding white blood cell cancer type based on the outcomes of the categorization process.

6.2 MODULE DESCRIPTION

The modules in this project are

1. Preprocessing.
2. Segmentation.
3. Feature Extraction.
4. Classification Using KNN, RF, CNN & SVM.

Preprocessing:

This module involves initial processing steps to enhance and prepare the input data before further analysis.

Segmentation:

The segmentation module focuses on dividing the processed data into meaningful and distinct segments for more targeted analysis.

Feature Extraction:

In this module, essential features are identified and extracted from the segmented data, providing valuable information for subsequent analysis.

Classification Using KNN, CNN, RF, SVM:

The classification module employs various algorithms, including K-Nearest Neighbors (KNN), Convolutional Neural Network (CNN), Random Forest (RF), and Support Vector Machine (SVM), to categorize and identify patterns within the extracted features. This step facilitates the system's ability to make informed decisions and predictions based on the input data.

6.2.1 PRE PROCESSING

In this module, the blood sample image undergoes preparation for segmentation by transitioning from the RGB color space to the YCBCR color space. This color space, where Y represents brightness (luma), CB represents blue minus luma (B-Y), and CR represents red minus luma (R-Y), is chosen due to the distinctive reddish and bluish colors present in our blood samples. Post-conversion to YCBCR space, the CB and CR coefficients are extracted and utilized in the subsequent cell segmentation process.

The preprocessing phase is further divided into three key sections:

Normalization:

This section standardizes the image data to a consistent scale, ensuring uniformity and aiding in subsequent processing steps.

Contrast Enhancement:

The contrast enhancement step aims to improve the visual quality of the images, emphasizing variations in intensity and aiding in the identification of crucial features.

Noise Removal:

Within this section, efforts are made to eliminate any unwanted noise or disturbances present in the images, contributing to a cleaner and more accurate segmentation process.

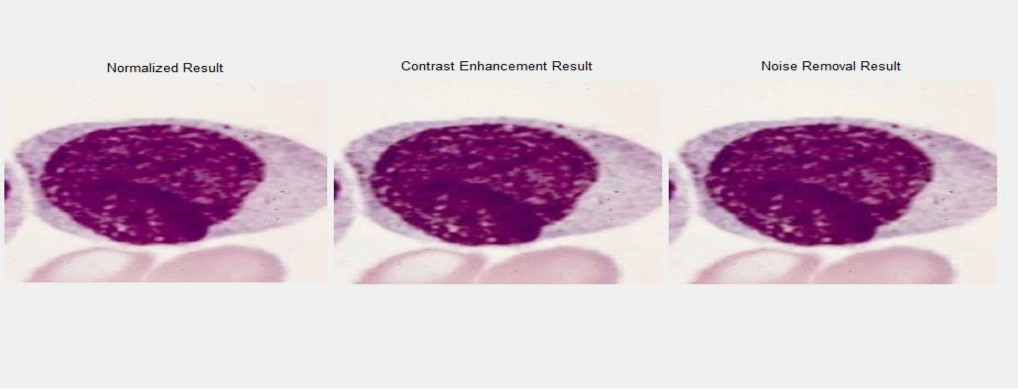


Fig: 6.2.1 – Pre processing

6.2.2. SEGMENTATION

This stage is designed to delineate the entire cell from its surrounding background. Utilizing the Cb and Cr coefficients extracted from the training images during the preprocessing stage in YCbCr color space, we isolate valuable pixels that are highly likely to be part of our regions of interest (ROI). This segmentation process aims to accurately identify and separate the cell components from the relative background, facilitating focused analysis on the areas of interest within the images.

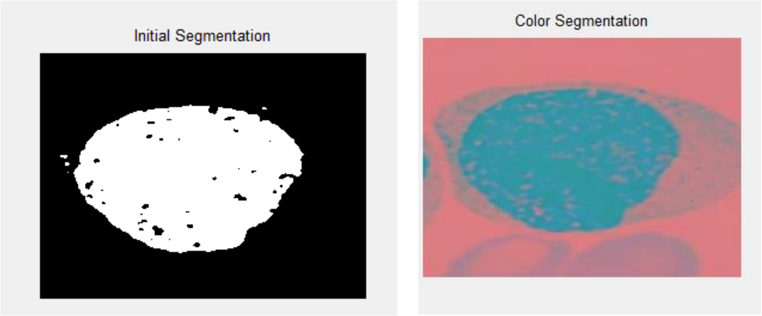


Fig 6.2.2- Segmentation

6.2.3 FEATURE EXTRACTION

Initially, the segmentation dataset is loaded during this stage, and the subsequent focus is on feature extraction. Our proposed approach involves computing a diverse set of features, including morphological, statistical, size ratio, and texture features. Within this framework, we specifically calculate contrast, correlation, energy, homogeneity, mean, and standard deviation as key metrics.

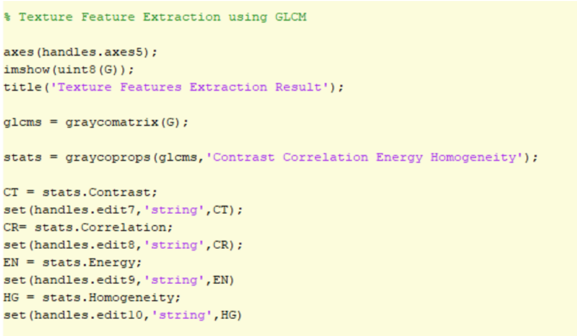
Mean

This represents the average value of pixel intensity within the Region of Interest (ROI).

Standard Deviation:

Standard deviation provides insights into how pixel intensity deviates from the mean intensity within the ROI, reflecting the overall variability.

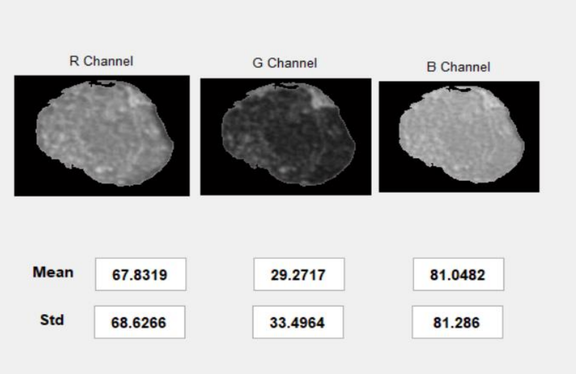
Utilizing the Gray-Level Co-occurrence Matrix (GLCM), we derive crucial features such as contrast, correlation, energy, and homogeneity. This matrix enables the quantification of spatial relationships between pixel intensities, offering valuable insights into the texture characteristics of the segmented regions.



```
% Texture Feature Extraction using GLCM

axes(handles.axes5);
imshow(uint8(G));
title('Texture Features Extraction Result');

glcms = graycomatrix(G);

stats = graycoprops(glcms,'Contrast Correlation Energy Homogeneity');

CT = stats.Contrast;
set(handles.edit7,'string',CT);
CR= stats.Correlation;
set(handles.edit8,'string',CR);
EN = stats.Energy;
set(handles.edit9,'string',EN)
HG = stats.Homogeneity;
set(handles.edit10,'string',HG)
```

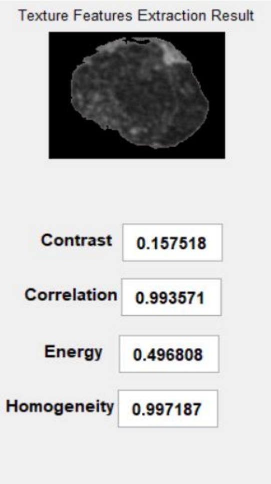
Fig 6.2.3.1- Feature extraction

After calculating the feature then the data will be stored in the FFV.mat dataset.



	R Channel	G Channel	B Channel
Mean	67.8319	29.2717	81.0482
Std	68.6266	33.4964	81.286

Fig 6.2.3.2 -Statistical color Feature Extraction



Texture Features Extraction Result

Contrast 0.157518

Correlation 0.993571

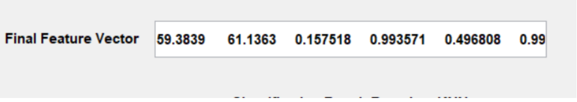
Energy 0.496808

Homogeneity 0.997187

Fig 6.2.3.3- Texture Feature Extraction

6.2.4 CLASSIFICATION USING KNN & SVM CLASSIFIER

In the KNN algorithm, a non-parametric approach is employed for both classification and regression tasks. The process involves determining the k closest neighbors from the trained dataset. The specific application of KNN depends on whether it is used for classification or regression in machine learning. It is categorized as an unsupervised methodology. For the classifier, the SR and FFV datasets are loaded, and feature extraction calculates the necessary features stored as vector values.



Final Feature Vector 59.3839 61.1363 0.157518 0.993571 0.496808 0.99

Fig 6.2.4- sample feature vector.

The vector value of the test image is then compared with the existing dataset using KNN, which identifies the k closest matches. Subsequently, KNN generates the shortest match vector value, forming the basis for the algorithm to produce the final output.

In the SVM algorithm, the same features extracted during the preprocessing stage are utilized. The model is trained using the features from the training images, and when presented with the feature set of a test image, the model classifies the new image.

For the CNN algorithm, the features extracted during preprocessing are employed to construct a connection mapping layer. The test image's features are input into this layer, which processes them to classify the new image based on the established connections within the model.

In the Random Forest (RF) algorithm, decision trees are built based on the trained set. The test image enters the decision tree, and classification is carried out based on the tree's structure, allowing the algorithm to classify the test image effectively.

Methods	Training	Testing	Correct	Wrong
KNN	450	50	48	2
SVM	450	50	43	7
RF	450	50	41	9
CNN	450	50	45	5

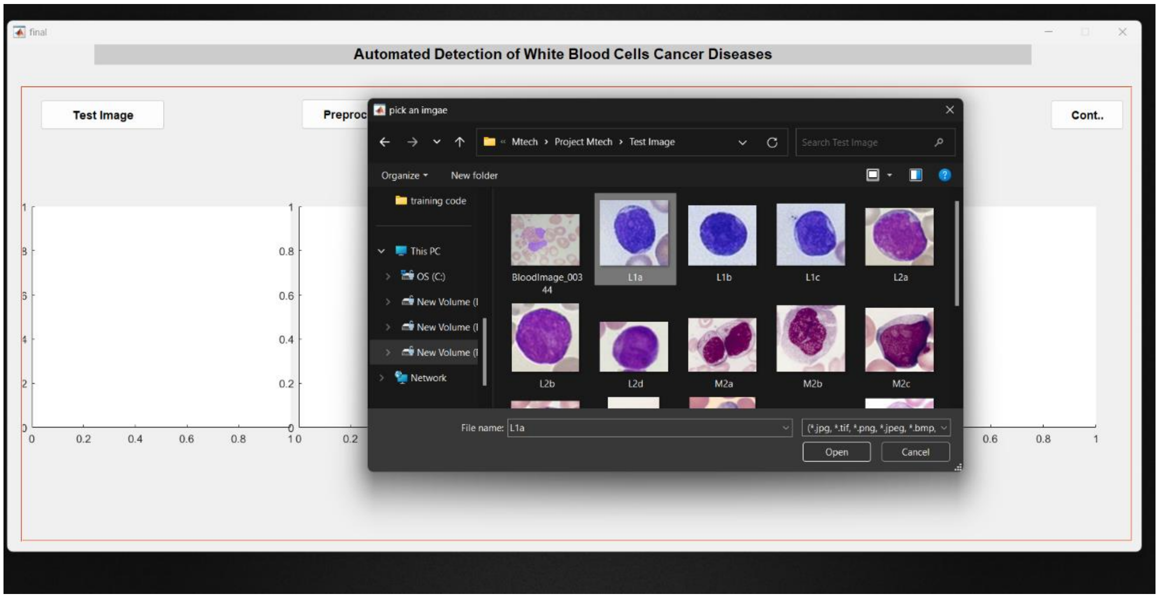
Table 6.2.4 – Classifications

Chapter – 7

PROJECT DEMONSTRATION

7.1 IMPLEMENTATION PROCESS:

Introducing the training image into the model



File name: L1a

Fig 7.1.1- load Image.

Following the loading process, preprocessing steps will be initiated.

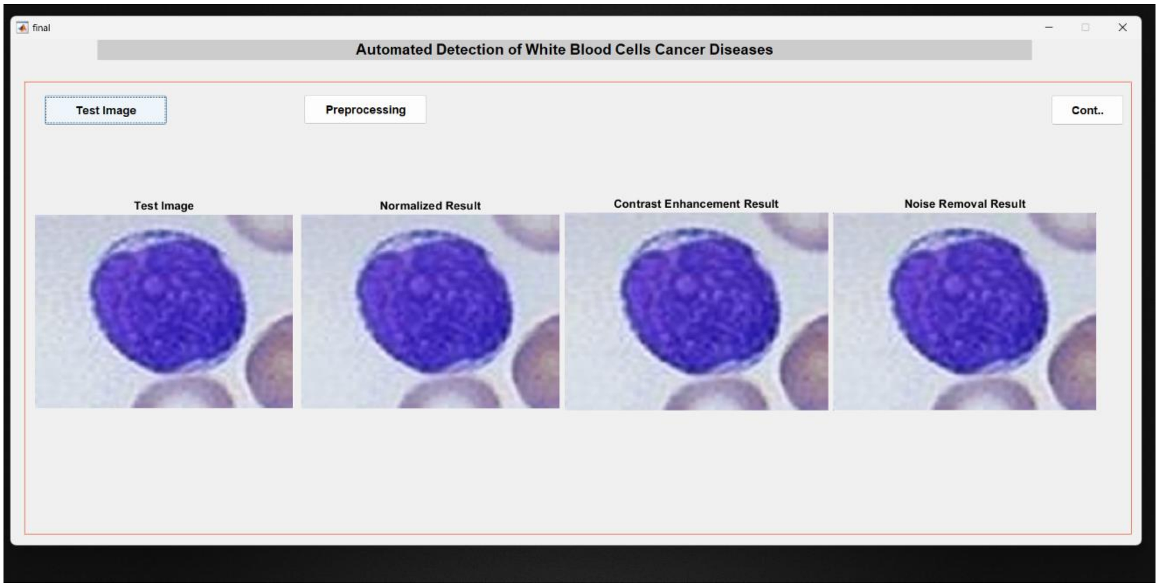

final
Automated Detection of White Blood Cells Cancer Diseases
Test Image
Preprocessing
Cont..
Test Image
Normalized Result
Contrast Enhancement Result
Noise Removal Result

Fig 7.1.2 – Preprocessing

Subsequent to the preprocessing stage, the segmentation process unfolds.

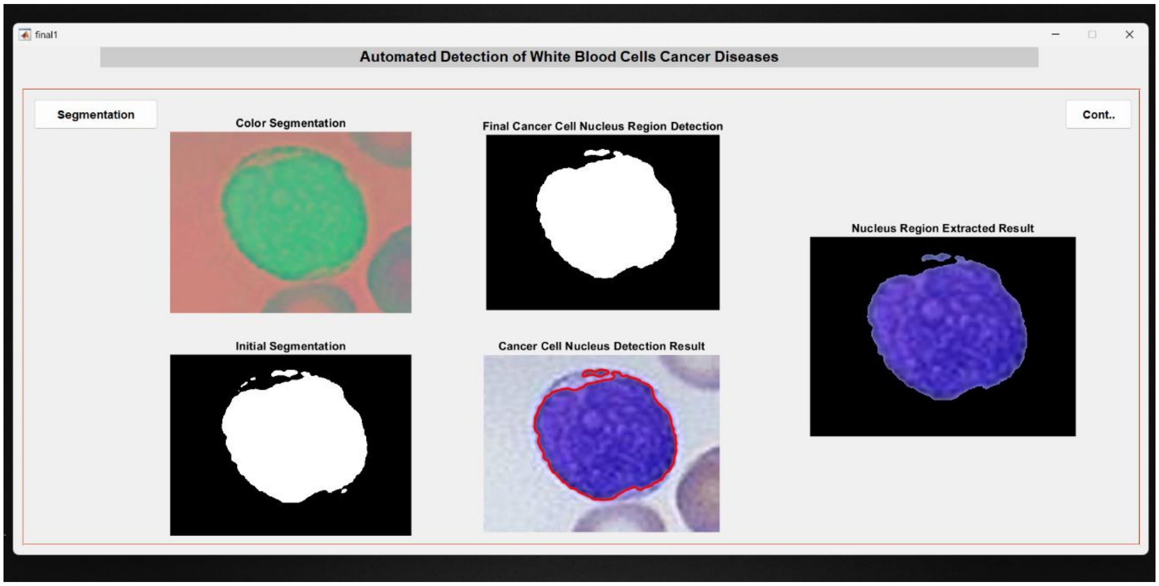
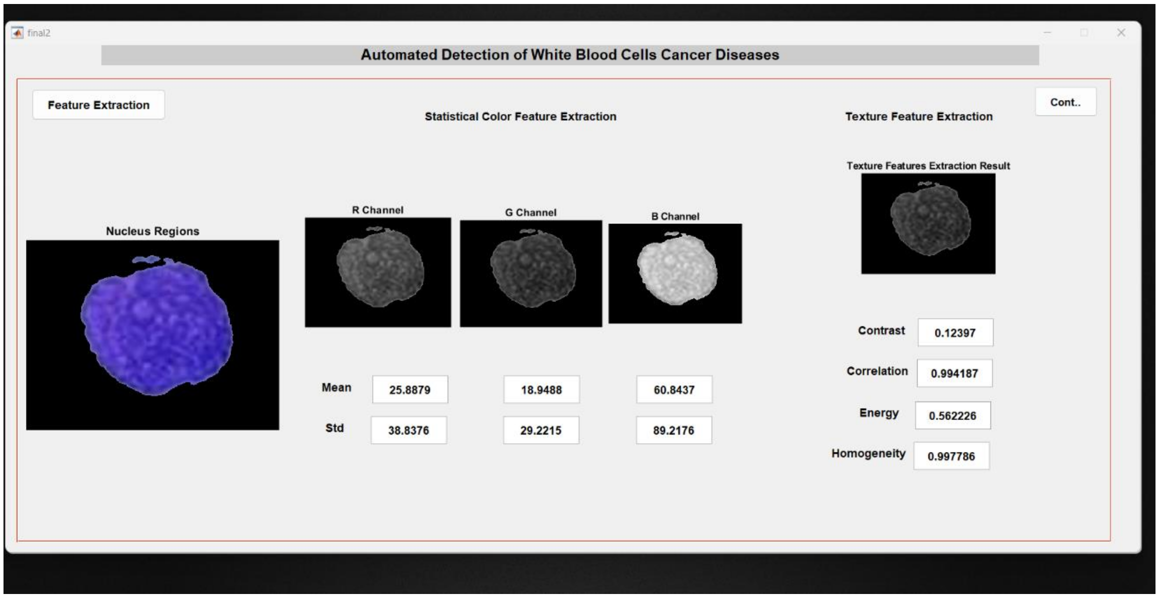

final1
Automated Detection of White Blood Cells Cancer Diseases
Segmentation
Color Segmentation
Final Cancer Cell Nucleus Region Detection
Cont..
Nucleus Region Extracted Result
Initial Segmentation
Cancer Cell Nucleus Detection Result

Fig 7.1.3 – Segmentation.

Following the segmentation process, the next step involves feature extraction.

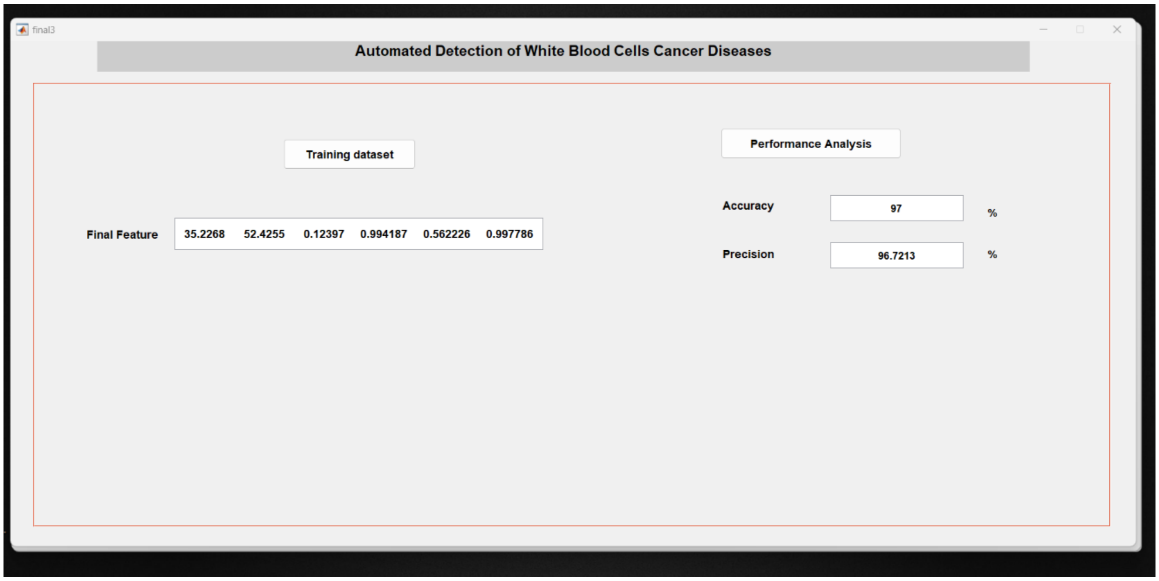

final2
Automated Detection of White Blood Cells Cancer Diseases
Feature Extraction
Statistical Color Feature Extraction
Texture Feature Extraction
Cont..
Texture Features Extraction Result
Nucleus Regions
R Channel
G Channel
B Channel

	R Channel	G Channel	B Channel
Mean	25.8879	18.9488	60.8437
Std	38.8376	29.2215	89.2176

Contrast	0.12397
Correlation	0.994187
Energy	0.562226
Homogeneity	0.997786

Fig 7.1.4 – Feature Extraction.

Following the feature extraction process, the extracted features are incorporated into the dataset. This step involves integrating the newly derived features into the existing dataset, enhancing the dataset's richness and comprehensiveness with the additional information obtained through the feature extraction phase.



Automated Detection of White Blood Cells Cancer Diseases

Final Feature: 35.2268 52.4255 0.12397 0.994187 0.562226 0.997786

Accuracy: 97 %

Precision: 96.7213 %

Fig 7.1.5 – Training of KNN

Initiating the KNN model by inputting the test image.

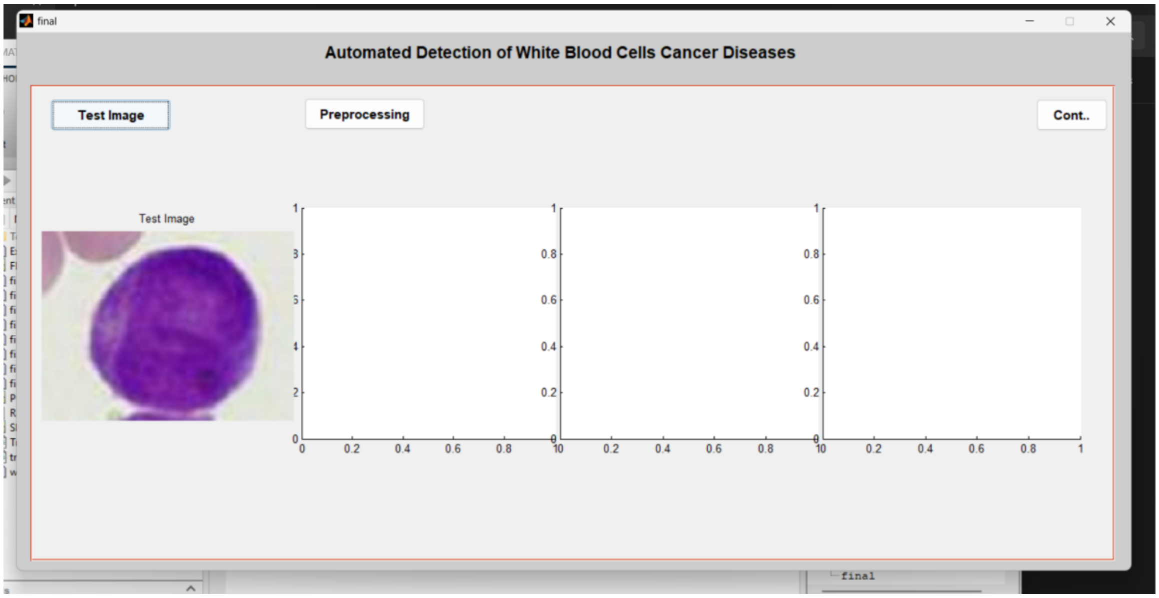


Fig 7.1.6 – Load Test Image.

Following the loading phase, preprocessing steps will be executed.

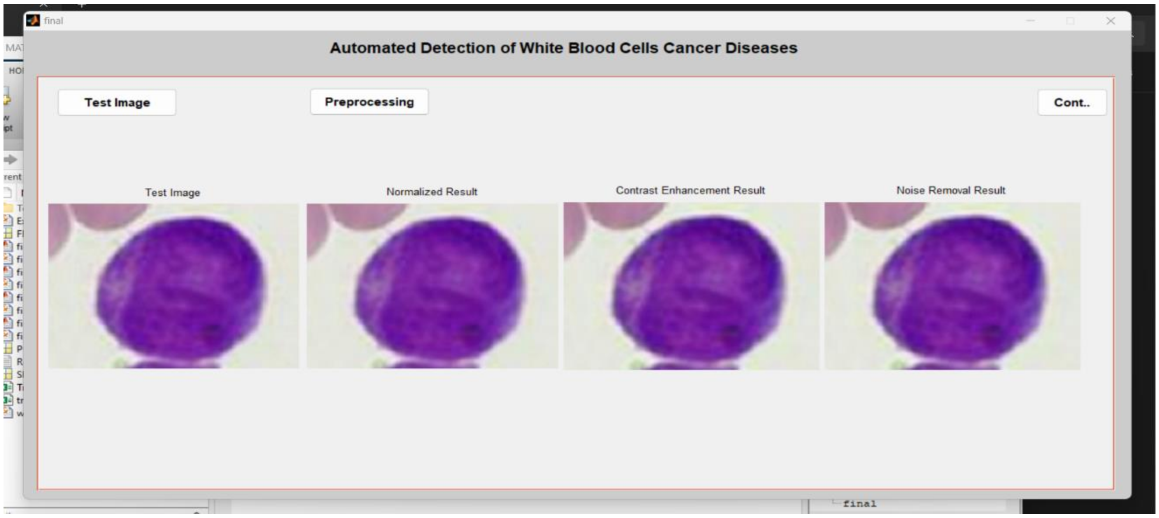

final
Automated Detection of White Blood Cells Cancer Diseases
Test Image
Preprocessing
Cont..
Test Image
Normalized Result
Contrast Enhancement Result
Noise Removal Result

Fig 7.1.7 – Preprocessing.

Subsequent to the preprocessing stage, the segmentation process unfolds.

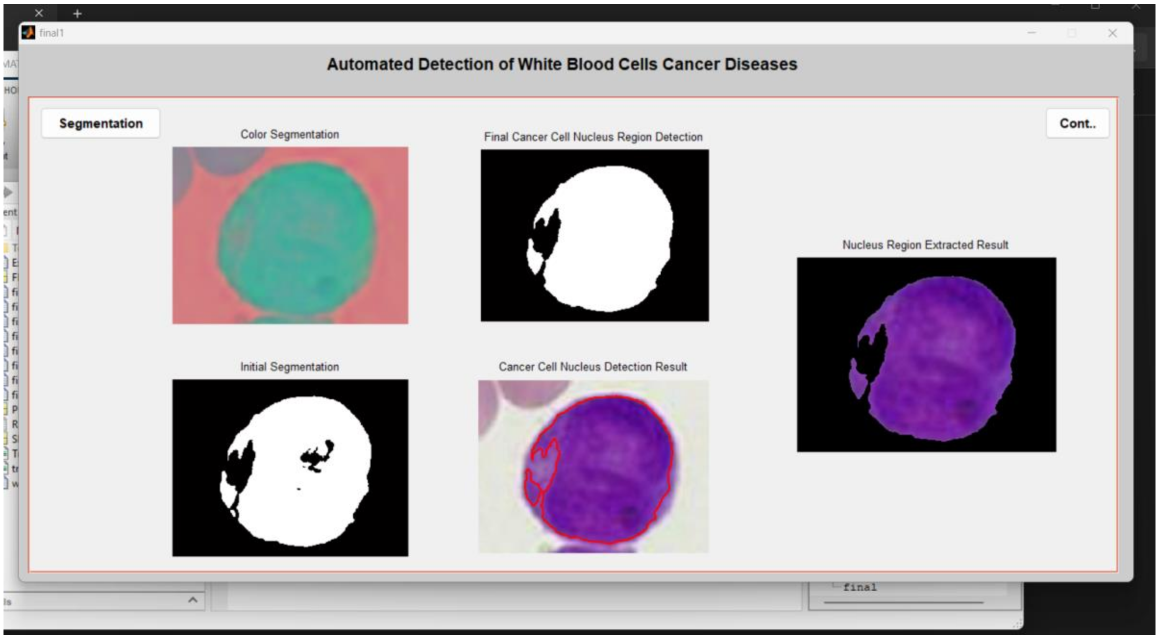
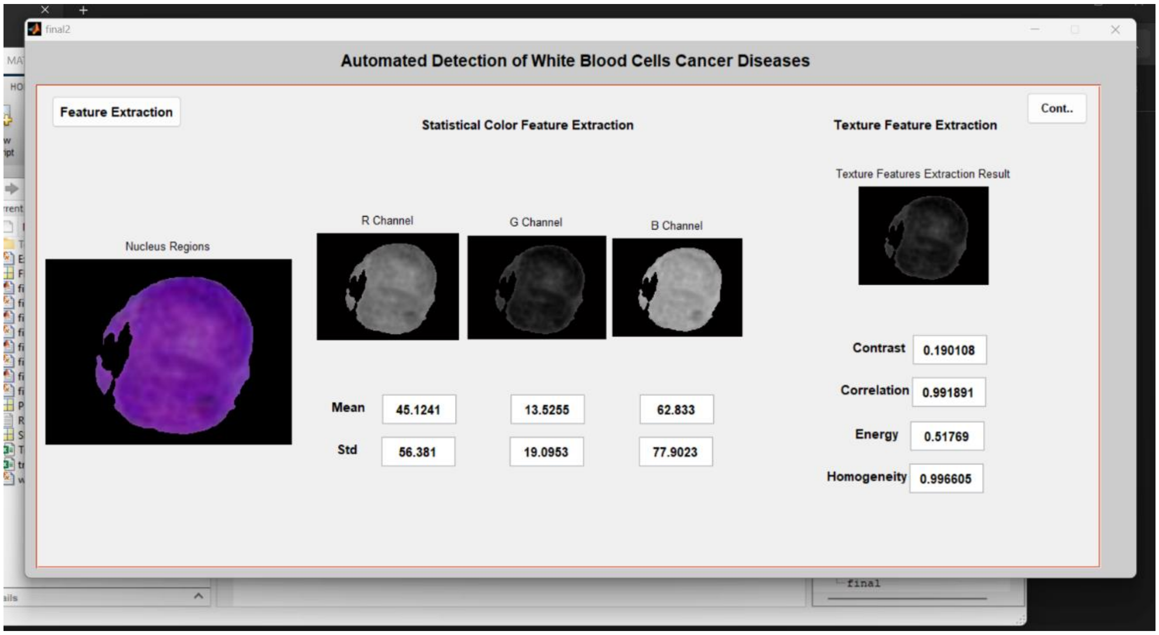

final1
Automated Detection of White Blood Cells Cancer Diseases
Segmentation
Cont..
Color Segmentation
Final Cancer Cell Nucleus Region Detection
Nucleus Region Extracted Result
Initial Segmentation
Cancer Cell Nucleus Detection Result
final

Fig 7.1.8 – Segmentation.

Following the segmentation process, the subsequent step involves feature extraction.


final2
Automated Detection of White Blood Cells Cancer Diseases
Feature Extraction
Cont..
Statistical Color Feature Extraction
Texture Feature Extraction
Texture Features Extraction Result
Nucleus Regions
R Channel
G Channel
B Channel

	R Channel	G Channel	B Channel
Mean	45.1241	13.5255	62.833
Std	56.381	19.0953	77.9023

Contrast 0.190108

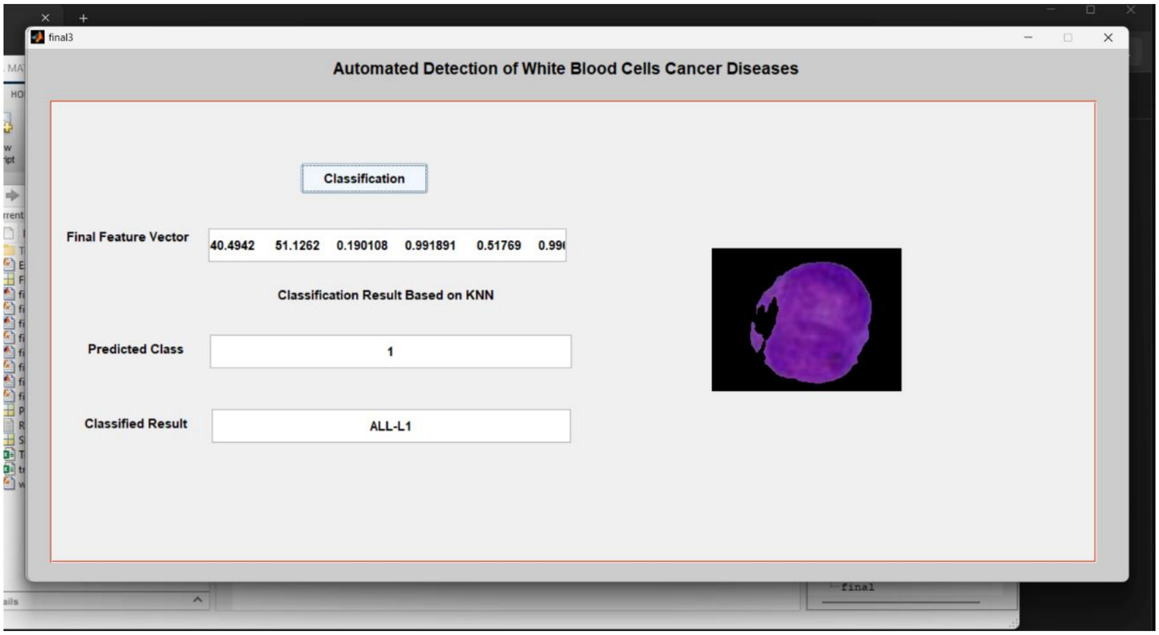
Correlation 0.991891

Energy 0.51769

Homogeneity 0.996605

Fig 7.1.9 – Feature Extraction.

Following the feature extraction phase, the features extracted from the image are compared with the dataset. The closely matching feature class is then identified and printed as the final classification for the image. This step involves evaluating the similarity between the extracted features and those stored in the dataset, allowing the system to assign the most appropriate class to the input image based on the degree of resemblance between their features.



Automated Detection of White Blood Cells Cancer Diseases

Final Feature Vector: 40.4942 51.1262 0.190108 0.991891 0.51769 0.99

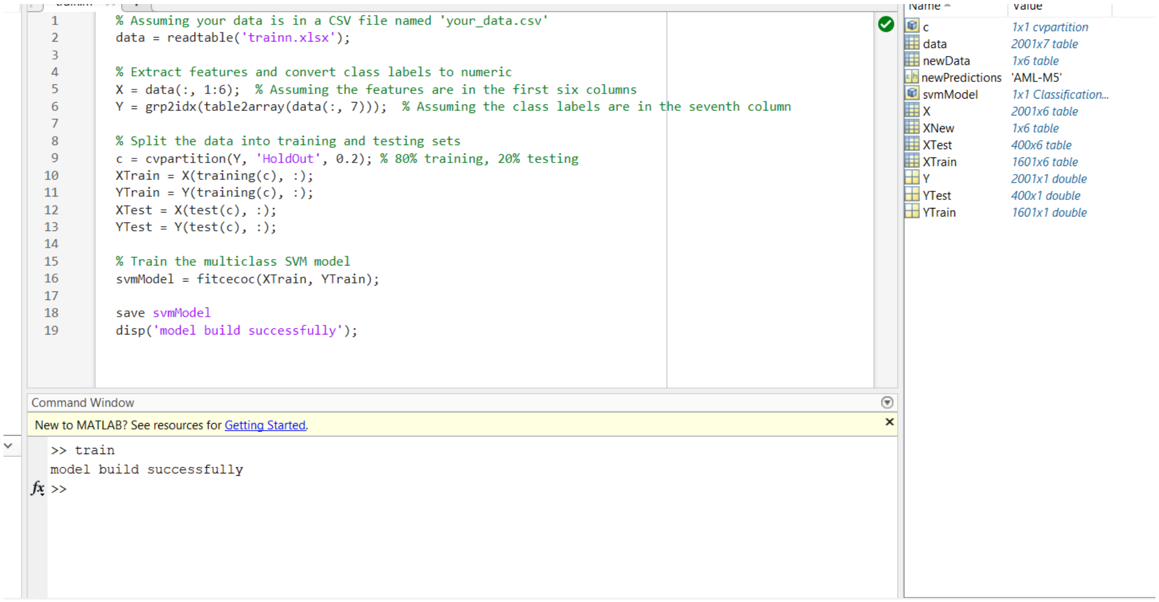
Classification Result Based on KNN

Predicted Class: 1

Classified Result: ALL-L1

Fig 7.1.10 – Classify.

The training process for Support Vector Machine (SVM) involves training the model to determine the optimal hyperplane that effectively separates distinct classes, aiming to maximize the margin between them. Further refinement is achieved through hyperparameter tuning and evaluation metrics, such as accuracy and precision. This iterative process persists until the desired performance is attained. Ultimately, the trained SVM model is deployed for making predictions on new, unseen data, showcasing its effectiveness in classification tasks.



```matlab
% Assuming your data is in a CSV file named 'your_data.csv'
data = readtable('trainn.xlsx');

% Extract features and convert class labels to numeric
X = data(:, 1:6);  % Assuming the features are in the first six columns
Y = grp2idx(table2array(data(:, 7)));  % Assuming the class labels are in the seventh column

% Split the data into training and testing sets
c = cvpartition(Y, 'HoldOut', 0.2); % 80% training, 20% testing
XTrain = X(training(c), :);
YTrain = Y(training(c), :);
XTest = X(test(c), :);
YTest = Y(test(c), :);

% Train the multiclass SVM model
svmModel = fitcecoc(XTrain, YTrain);

save svmModel
disp('model build successfully');
```

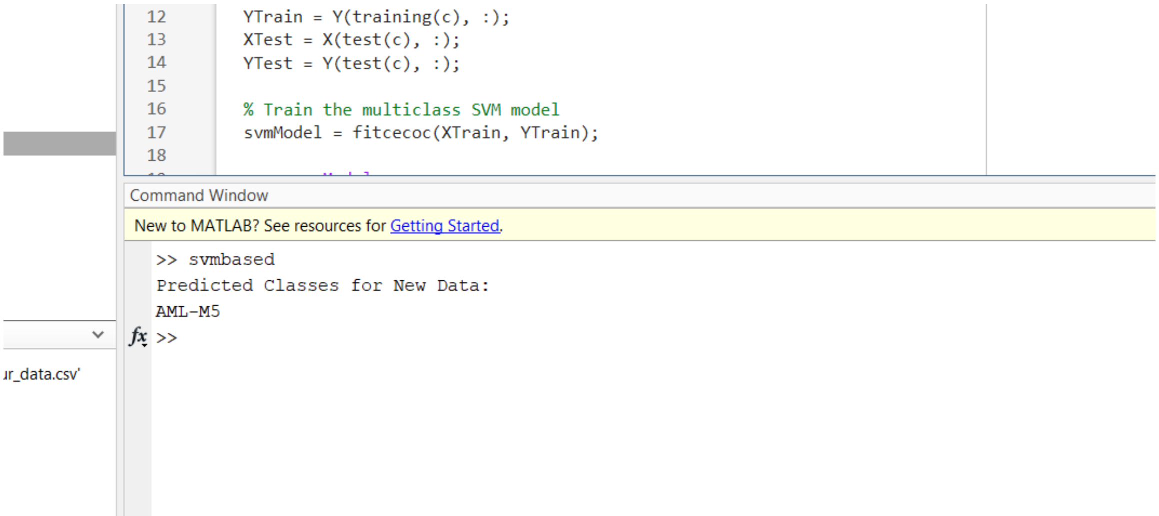
Command Window

New to MATLAB? See resources for Getting Started.

```
>> train
model build successfully
fx >>
```

Name	Value
c	1x1 cvpartition
data	2001x7 table
newData	1x6 table
newPredictions	'AML-M5'
svmModel	1x1 Classification...
X	2001x6 table
XNew	1x6 table
XTest	400x6 table
XTrain	1601x6 table
Y	2001x1 double
YTest	400x1 double
YTrain	1601x1 double

Fig 7.1.11 – SVM Training.



```matlab
YTrain = Y(training(c), :);
XTest = X(test(c), :);
YTest = Y(test(c), :);

% Train the multiclass SVM model
svmModel = fitcecoc(XTrain, YTrain);
```

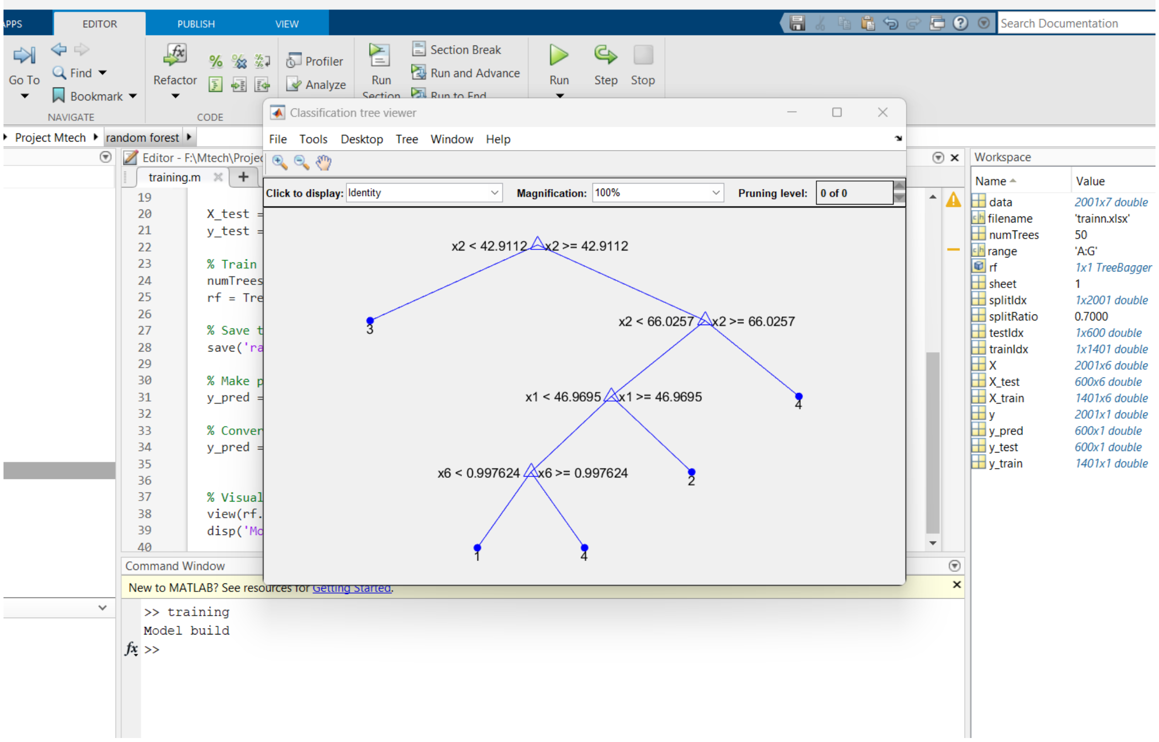
Command Window

New to MATLAB? See resources for Getting Started.

```
>> svmbased
Predicted Classes for New Data:
AML-M5
fx >>
```

Fig 7.1.12 – SVM Classification

The Random Forest training process entails several key steps to create a robust ensemble model. Initially, a labeled dataset is compiled, comprising input features and their corresponding class labels. Subsequently, multiple decision trees are constructed, each trained on a random subset of the dataset using bootstrapping. In the training of each tree, a random subset of features is considered at each split point, ensuring diversity among the trees. These individual trees collectively form a forest, and their outputs are aggregated through a voting mechanism for classification tasks.



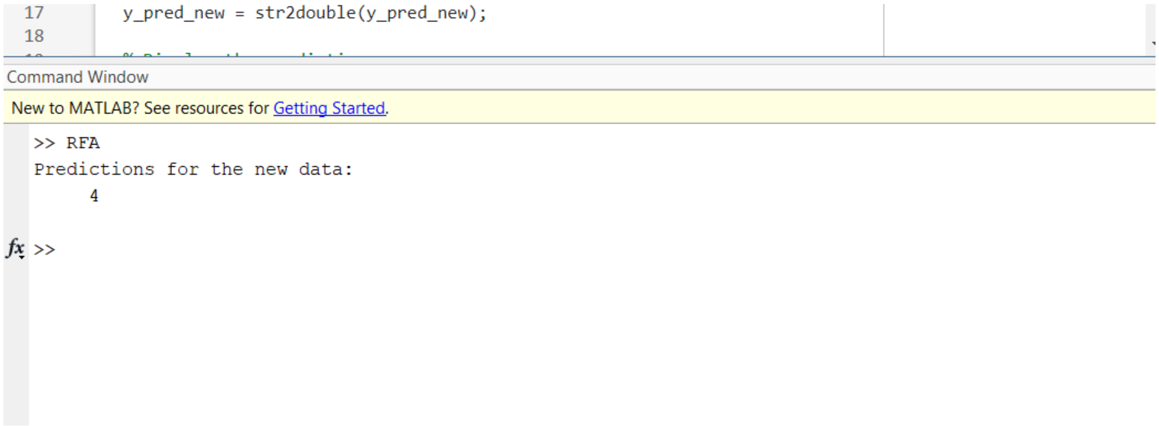
```
X_test =
y_test =

% Train
numTrees
rf = Tre

% Save t
save('ra

% Make p
y_pred =

% Conver
y_pred =


% Visual
view(rf.
disp('Mo
```

Name	Value
data	2001x7 double
filename	'trainn.xlsx'
numTrees	50
range	'A:G'
rf	1x1 TreeBagger
sheet	1
splitIdx	1x2001 double
splitRatio	0.7000
testIdx	1x600 double
trainIdx	1x1401 double
X	2001x6 double
X_test	600x6 double
X_train	1401x6 double
y	2001x1 double
y_pred	600x1 double
y_test	600x1 double
y_train	1401x1 double

New to MATLAB? See resources for Getting Started.

```
>> training
Model build
>>
```

Fig 7.1.12 – Random Forest Training.



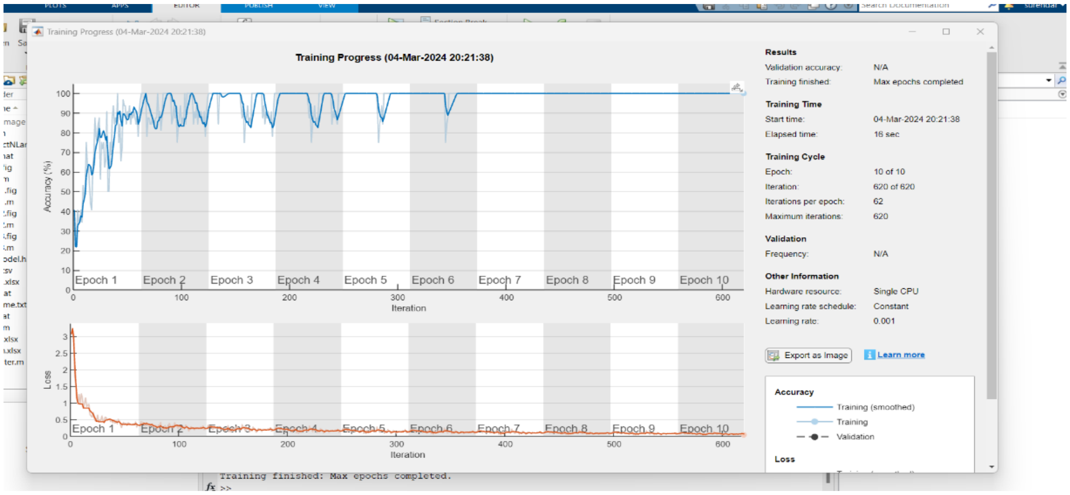
```
y_pred_new = str2double(y_pred_new);
```

New to MATLAB? See resources for Getting Started.

```
>> RFA
Predictions for the new data:
     4

>>
```

FIG 7.1.13 – RF Classification

For the CNN Training , the features extracted during preprocessing are employed to construct a connection mapping layer. The test image's features are input into this layer, which processes them to classify the new image based on the established connections within the model.



Training Progress (04-Mar-2024 20:21:38)

Results

Validation accuracy: N/A

Training finished: Max epochs completed

Training Time

Start time: 04-Mar-2024 20:21:38

Elapsed time: 16 sec

Training Cycle

Epoch: 10 of 10

Iteration: 620 of 620

Iterations per epoch: 62

Maximum iterations: 620

Validation

Frequency: N/A

Other Information

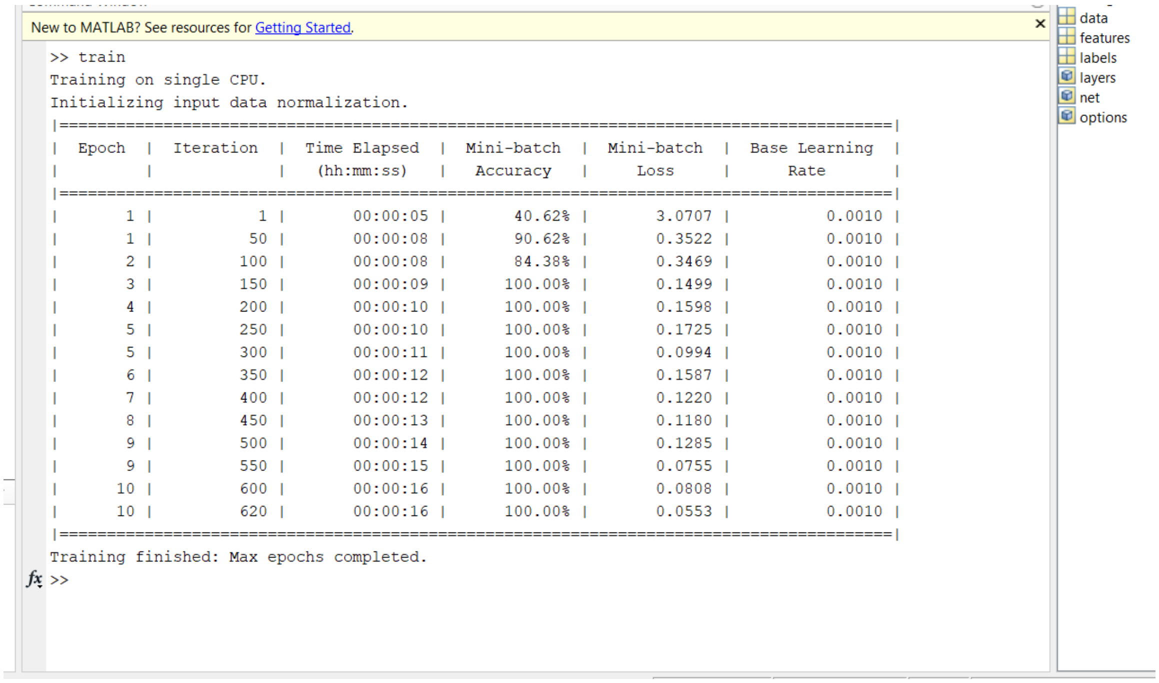
Hardware resource: Single CPU

Learning rate schedule: Constant

Learning rate: 0.001

```
Training finished: Max epochs completed.
fx >>
```

Fig 7.1.13 – CNN Training.

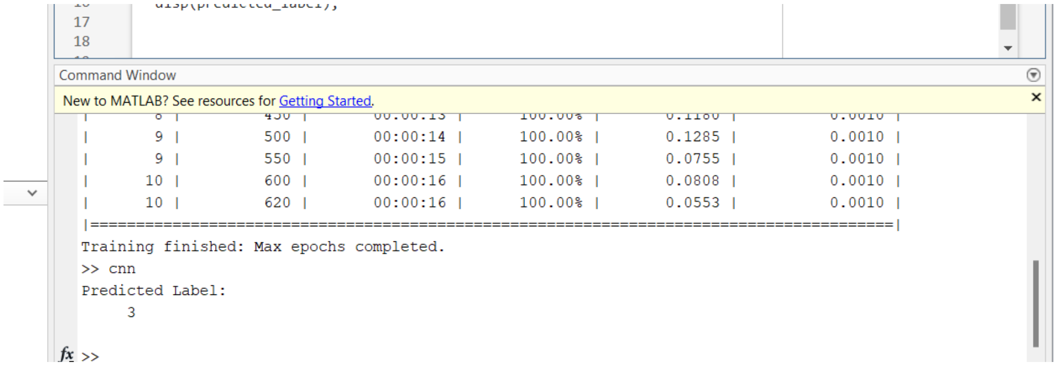


New to MATLAB? See resources for Getting Started.

```
>> train
Training on single CPU.
Initializing input data normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:00:05 |       40.62% |       3.0707 |          0.0010 |
|       1 |          50 |       00:00:08 |       90.62% |       0.3522 |          0.0010 |
|       2 |         100 |       00:00:08 |       84.38% |       0.3469 |          0.0010 |
|       3 |         150 |       00:00:09 |      100.00% |       0.1499 |          0.0010 |
|       4 |         200 |       00:00:10 |      100.00% |       0.1598 |          0.0010 |
|       5 |         250 |       00:00:10 |      100.00% |       0.1725 |          0.0010 |
|       5 |         300 |       00:00:11 |      100.00% |       0.0994 |          0.0010 |
|       6 |         350 |       00:00:12 |      100.00% |       0.1587 |          0.0010 |
|       7 |         400 |       00:00:12 |      100.00% |       0.1220 |          0.0010 |
|       8 |         450 |       00:00:13 |      100.00% |       0.1180 |          0.0010 |
|       9 |         500 |       00:00:14 |      100.00% |       0.1285 |          0.0010 |
|       9 |         550 |       00:00:15 |      100.00% |       0.0755 |          0.0010 |
|      10 |         600 |       00:00:16 |      100.00% |       0.0808 |          0.0010 |
|      10 |         620 |       00:00:16 |      100.00% |       0.0553 |          0.0010 |
|========================================================================================|
Training finished: Max epochs completed.
fx >>
```

data
features
labels
layers
net
options

Fig 7.1.14 – CNN Epoch process.



```
|        9 |         500 |     00:00:14 |      100.00% |       0.1285 |           0.0010 |
|        9 |         550 |     00:00:15 |      100.00% |       0.0755 |           0.0010 |
|       10 |         600 |     00:00:16 |      100.00% |       0.0808 |           0.0010 |
|       10 |         620 |     00:00:16 |      100.00% |       0.0553 |           0.0010 |
|======================================================================================|
Training finished: Max epochs completed.
>> cnn
Predicted Label:
     3

>>
```

Fig 7.1.14 – CNN Classification

CHAPTER 8

Conclusion & Future Work

8.1 CONCLUSION

We present a novel approach for the automatic detection of white blood cell cancer disease, encompassing four key stages: preprocessing, segmentation, feature extraction, and classification. Each stage plays a vital role in achieving accurate disease detection. The preprocessing stage involves YCbCr color conversion, enhancing the system's ability to discern critical features. Segmentation focuses on cell segmentation, isolating relevant components for further analysis. Feature extraction encompasses the extraction of texture, morphological, statistical, and size ratio features, providing comprehensive insights into the cell's characteristics.

In the classification stage, four distinct models—Convolutional Neural Network (CNN), Random Forest (RF), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM)—are employed to classify the cells and identify potential cancerous conditions. Notably, our system is designed to function as an expert system, enhancing learning efficiency by allowing the system to adapt and learn from input tests, thereby classifying diseases with new labels.

Methods	Accuracy
KNN	96%
SVM	86%
RF	85%
CNN	91%

Table 8.1 - accuracy

Through rigorous testing and evaluation, our proposed method has demonstrated superior accuracy compared to existing systems. A detailed analysis reveals that the KNN model outperforms other classification models in terms of accuracy during the testing stage, emphasizing its effectiveness in white blood cell cancer disease detection.

8.2 FUTURE WORK

The future evolution of this concept holds the potential for significant advancement by incorporating additional features, specifically the ability to detect a broader range of white blood cell cancer diseases. This expansion aims to culminate in the creation of a comprehensive system designed to address a spectrum of white blood cell diseases. The overarching goal is to enhance the system's capabilities, providing a more inclusive and robust solution for the detection and classification of various types of white blood cell cancers.

Chapter - 9
SOCIAL/ENVIRONMENTAL IMPACT

9.1 IMPACT

Healthcare System: By preventing advanced-stage cases that call for more involved and costly treatments, early detection of leukemia allows for prompt intervention and treatment, potentially lessening the strain on healthcare systems. Quality of Life: By providing early access to suitable medical care and support services, timely detection can enhance the quality of life for people with leukemia. This can lessen the financial, emotional, and physical toll that advanced-stage cancer has to take. Public Awareness: The discovery of leukemia brings attention to the significance of routine health examinations and early cancer detection in all forms. It encourages dialogue regarding symptoms, risk factors, and resources accessible to individuals impacted by the illness.

Chapter - 10
SUMMARY

10.1 SUMMARY:

As an improvement on the current method, we tend to develop and produce an upgraded application in our proposed system that aids in the diagnosis of white blood cell disease. The primary goal is to identify and categorize cancers of the white blood cells. Our enlarged system is intended to outperform the current one in some manner, allowing us to classify all subtypes of sickness. We typically display results using MATLAB that surpass the performance analysis parameters of detection and classification accuracy of the current system. The results will be compared with previously planned procedures, demonstrating the validity of the suggested methodology and the reliability of using planned strategies for prior disclosure of white blood cell cancer disease. The application that aids in the diagnosis of process white blood cell disorder is presented by the proposed system. Leukemia and myeloma are the two categories into which these diseases fall. Each region includes illnesses with similar symptoms that might be challenging to diagnose. This system is applied to one of the two illness groups by computing unique properties. Finally, the final judgment is applied using KNN, SVM, RF, and CNN. The suggested method seeks to improve system learning, decrease the number of cases of misdiagnosis, and enable early detection of white blood cell cancer

REFERENCES

1. Anwar, S., & Alam, A. (2020). A convolutional neural network–based learning approach to acute lymphoblastic leukaemia detection with automated feature extraction. Medical & Biological Engineering & Computing, 58(12), 3113–3121.
2. Chand, S., & Vishwakarma, V. P. (2022). A novel Deep Learning Framework (DLF) for classification of Acute Lymphoblastic Leukemia. Multimedia Tools and Applications, 81(26), 37243–37262.
3. Khouani, A., Daho, M. E. H., Mahmoudi, S. A., Chikh, M. A., & Benzineb, B. (2020). Automated recognition of white blood cells using deep learning. Biomedical Engineering Letters, 10(3), 359–367.
4. Shahzad, A., Raza, M., Shah, J. H., Sharif, M., & Nayak, R. S. (2021). Categorizing white blood cells by utilizing deep features of proposed 4B-AdditionNet-based CNN network with ant colony optimization. Complex & Intelligent Systems, 8(4), 3143–3159.
5. Agustin, R. I., Arif, A., & Sukorini, U. (2021). Classification of immature white blood cells in acute lymphoblastic leukemia L1 using neural networks particle swarm optimization. Neural Computing and Applications, 33(17), 10869–10880.
6. Acharya, V., & Kumar, P. (2019). Detection of acute lymphoblastic leukemia using image segmentation and data mining algorithms. Medical & Biological Engineering & Computing, 57(8), 1783–1811.
7. Das, B. K., & Dutta, H. S. (2020). GFNB: Gini index–based Fuzzy Naive Bayes and blast cell segmentation for leukemia detection using multi-cell blood smear images. Medical & Biological Engineering & Computing, 58(11), 2789–2803.
8. Zolfaghari, M. R., & Sajedi, H. (2022). A survey on automated detection and classification of acute leukemia and WBCs in microscopic blood cells. Multimedia Tools and Applications, 81(5), 6723–6753.
9. Shamama Anwar , Afrin Alam (2020). A convolutional neural network–based learning approach to acute lymphoblastic leukaemia detection with automated feature extraction. Medical & Biological Engineering & Computing 58:3113–3121.

10. Amin Khouani , Mostafa El Habib Daho , Sidi Ahmed Mahmoud (2020). Automated recognition of white blood cells using deep learning. Biomedical Engineering Letters 10:359–367

11. Sunita Chand , Virendra P. Vishwakarma (2022). A novel Deep Learning Framework (DLF) for classification of Acute Lymphoblastic Leukemia. Multimedia Tools and Applications 81:37243–37262

12. Nadeem Akram , Sharjeel Adnan , Muhammad Asi , Syed Muhammad Ali Imran (2022). Exploiting the Multiscale Information Fusion Capabilities for Aiding the Leukemia Diagnosis Through White Blood Cells Segmentation. Digital Object Identifier 10.1109/ACCESS.2022.3171916

13. X. Zheng, Y. Wang, G. Wang, and J. Liu.Fast (2018). Robust segmentation of white blood cell images by self-supervised learning. Micron, vol. 107, pp. 55–71.

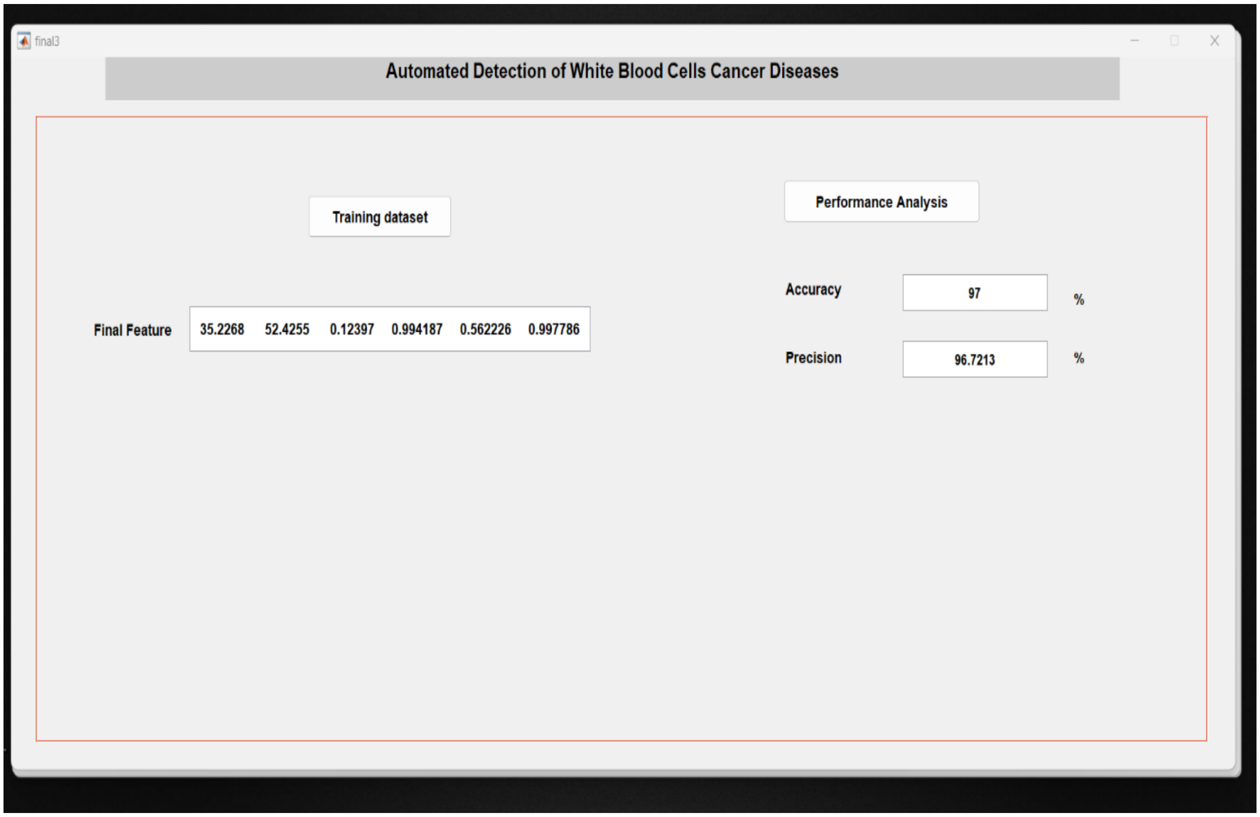
14. M. LaFleur-Brooks (2018). A Student-Directed Approach, 10th ed., D. L. Brooks, Ed. St. Louis, MO, USA: Elsevier, pp. 757–758.

15. Loey M, Naman MR, Zayed HH (2020). A survey on blood image diseases detection using deep learning, vol. 11.

APPENDIX A

OUTPUT SCREENSHOTS

TRAINING OF KNN :



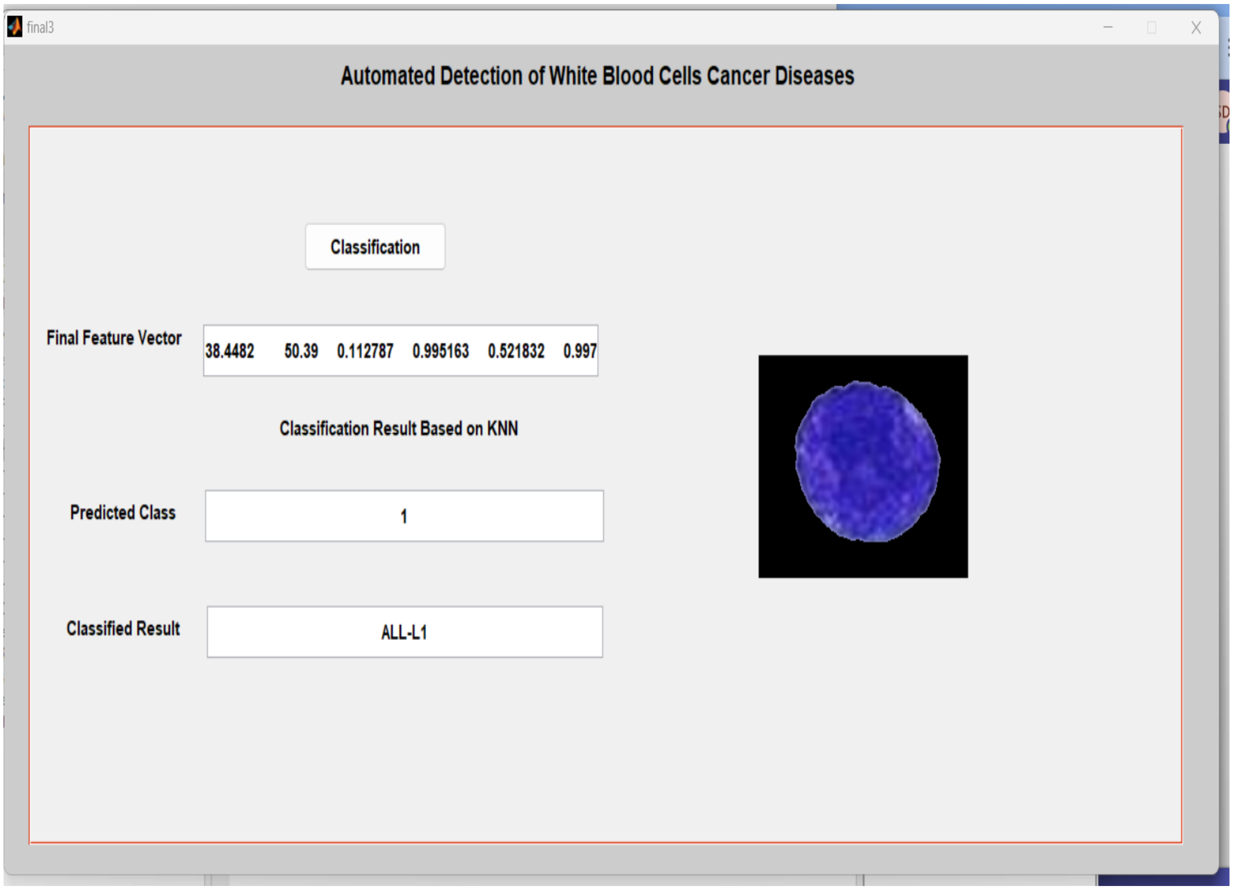
Automated Detection of White Blood Cells Cancer Diseases

Final Feature 35.2268 52.4255 0.12397 0.994187 0.562226 0.997786

Accuracy 97 %

Precision 96.7213 %

CLASSFICATION OF KNN:



Automated Detection of White Blood Cells Cancer Diseases

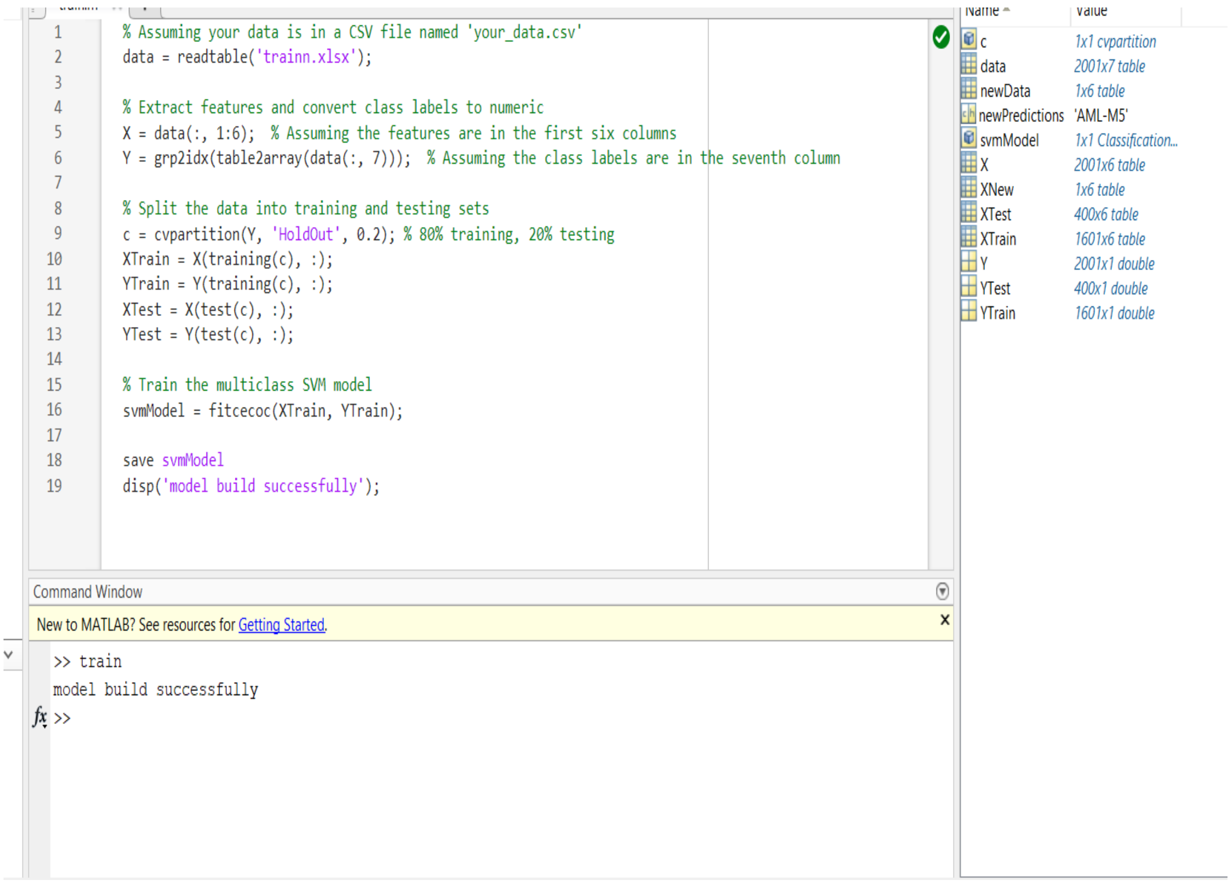
Final Feature Vector: 38.4482 50.39 0.112787 0.995163 0.521832 0.997

Classification Result Based on KNN

Predicted Class: 1

Classified Result: ALL-L1

TRAINING OF SVM :



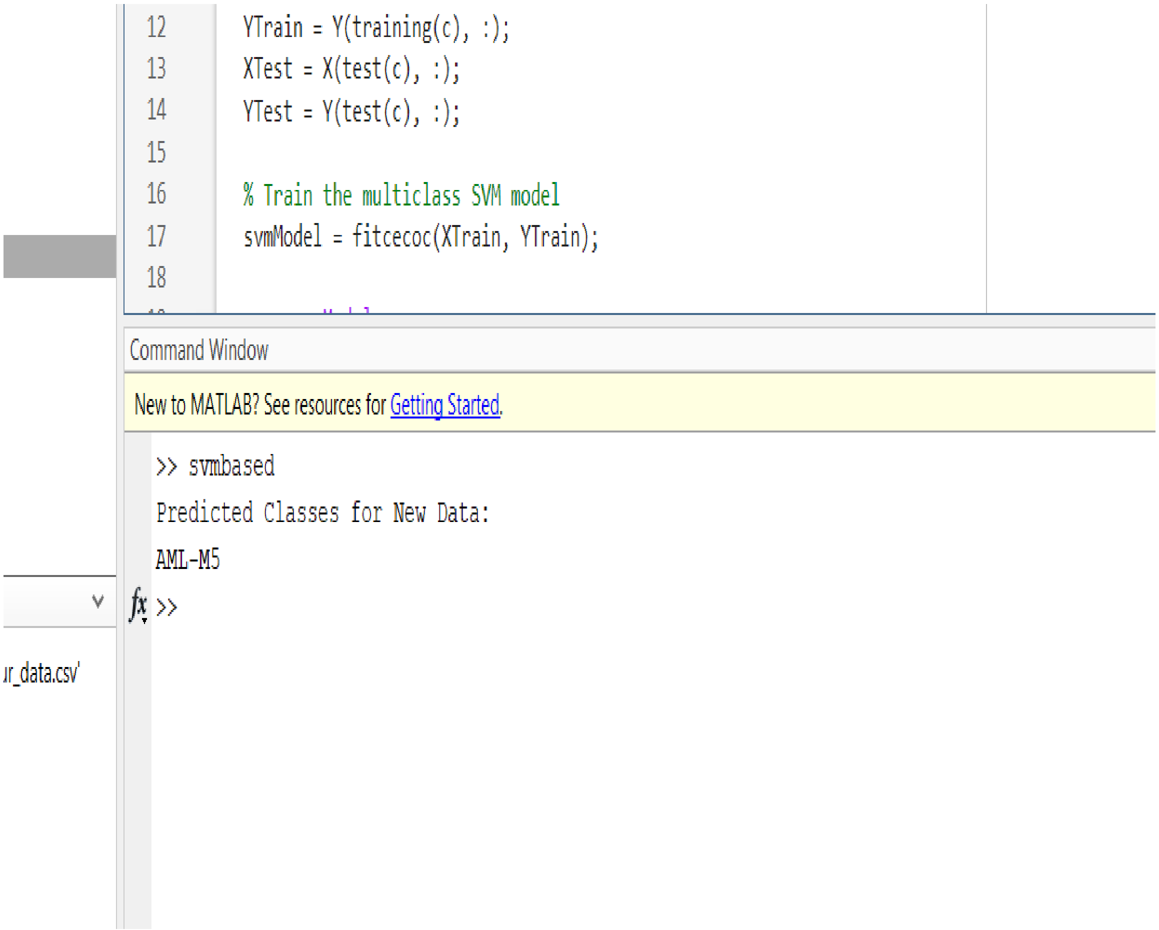
```
% Assuming your data is in a CSV file named 'your_data.csv'
data = readtable('trainn.xlsx');

% Extract features and convert class labels to numeric
X = data(:, 1:6);  % Assuming the features are in the first six columns
Y = grp2idx(table2array(data(:, 7)));  % Assuming the class labels are in the seventh column

% Split the data into training and testing sets
c = cvpartition(Y, 'HoldOut', 0.2); % 80% training, 20% testing
XTrain = X(training(c), :);
YTrain = Y(training(c), :);
XTest = X(test(c), :);
YTest = Y(test(c), :);

% Train the multiclass SVM model
svmModel = fitcecoc(XTrain, YTrain);

save svmModel
disp('model build successfully');
```

New to MATLAB? See resources for Getting Started.

```
>> train
model build successfully
>>
```

Name	Value
c	1x1 cvpartition
data	2001x7 table
newData	1x6 table
newPredictions	'AML-M5'
svmModel	1x1 Classification...
X	2001x6 table
XNew	1x6 table
XTest	400x6 table
XTrain	1601x6 table
Y	2001x1 double
YTest	400x1 double
YTrain	1601x1 double

CLASSFICATION OF SVM:

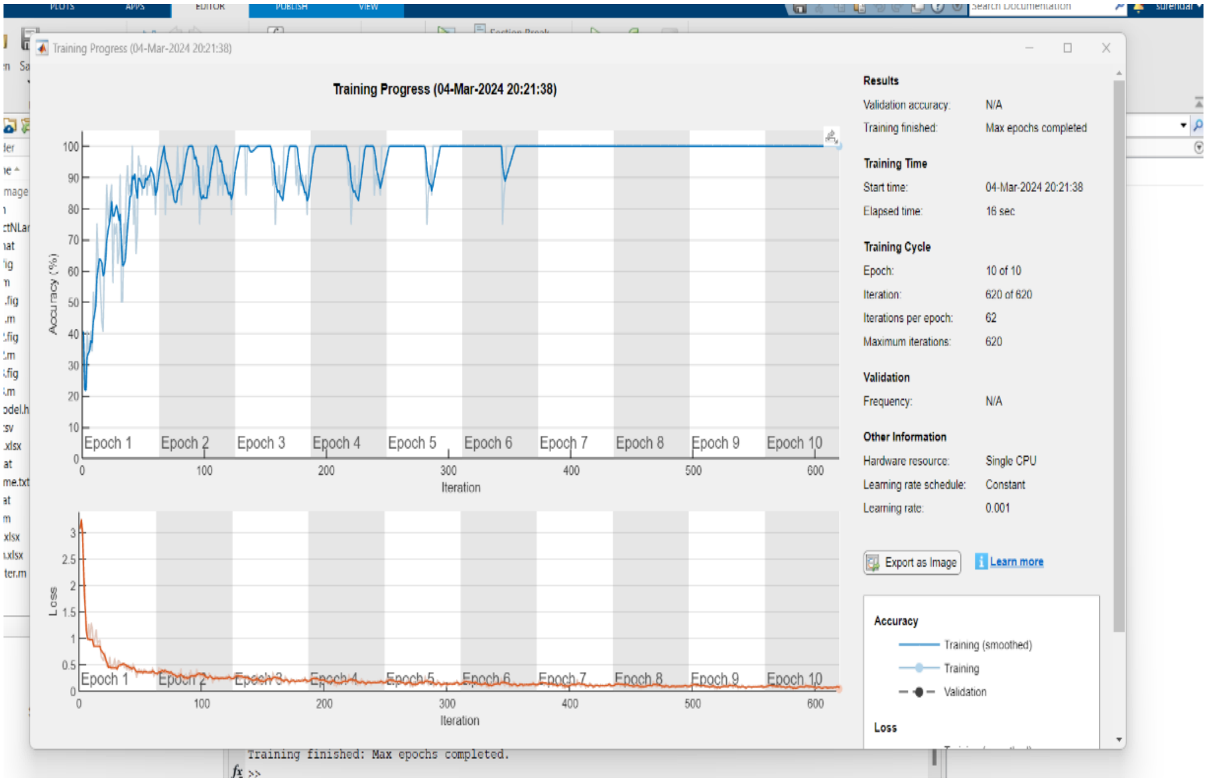


```
YTrain = Y(training(c), :);
XTest = X(test(c), :);
YTest = Y(test(c), :);

% Train the multiclass SVM model
svmModel = fitcecoc(XTrain, YTrain);

```

New to MATLAB? See resources for Getting Started.

```
>> svmbased
Predicted Classes for New Data:
AML-M5
>>
```

TRAINING OF CNN:



Results

Validation accuracy: N/A

Training finished: Max epochs completed

Training Time

Start time: 04-Mar-2024 20:21:38

Elapsed time: 16 sec

Training Cycle

Epoch: 10 of 10

Iteration: 620 of 620

Iterations per epoch: 62

Maximum iterations: 620

Validation

Frequency: N/A

Other Information

Hardware resource: Single CPU

Learning rate schedule: Constant

Learning rate: 0.001

Accuracy

Training (smoothed)

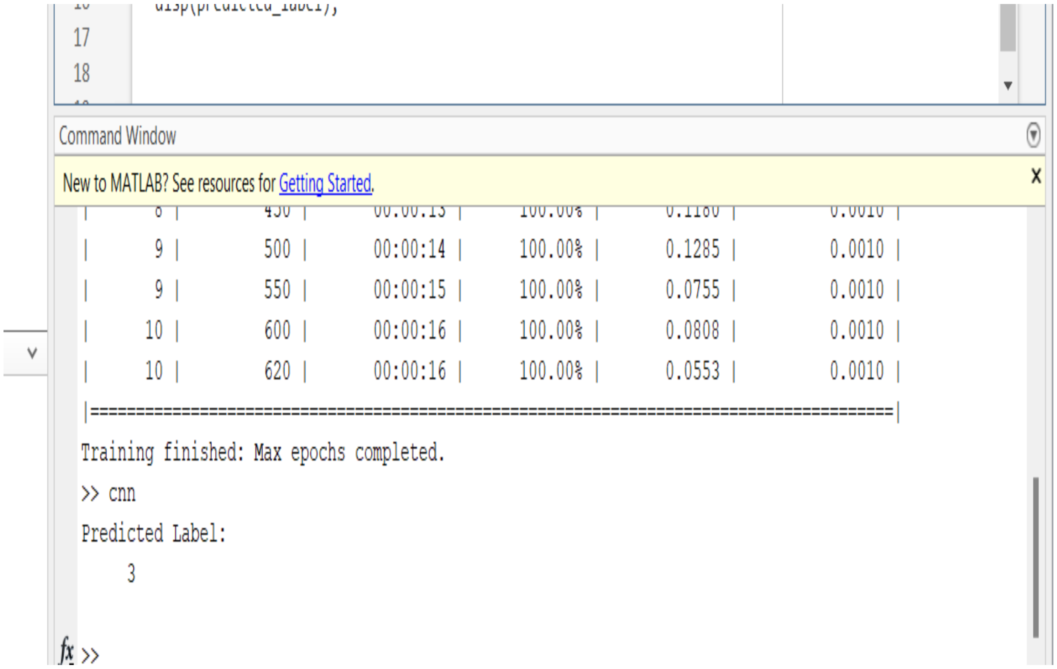
Training

Validation

Loss

```
Training finished: Max epochs completed.
fx >>
```

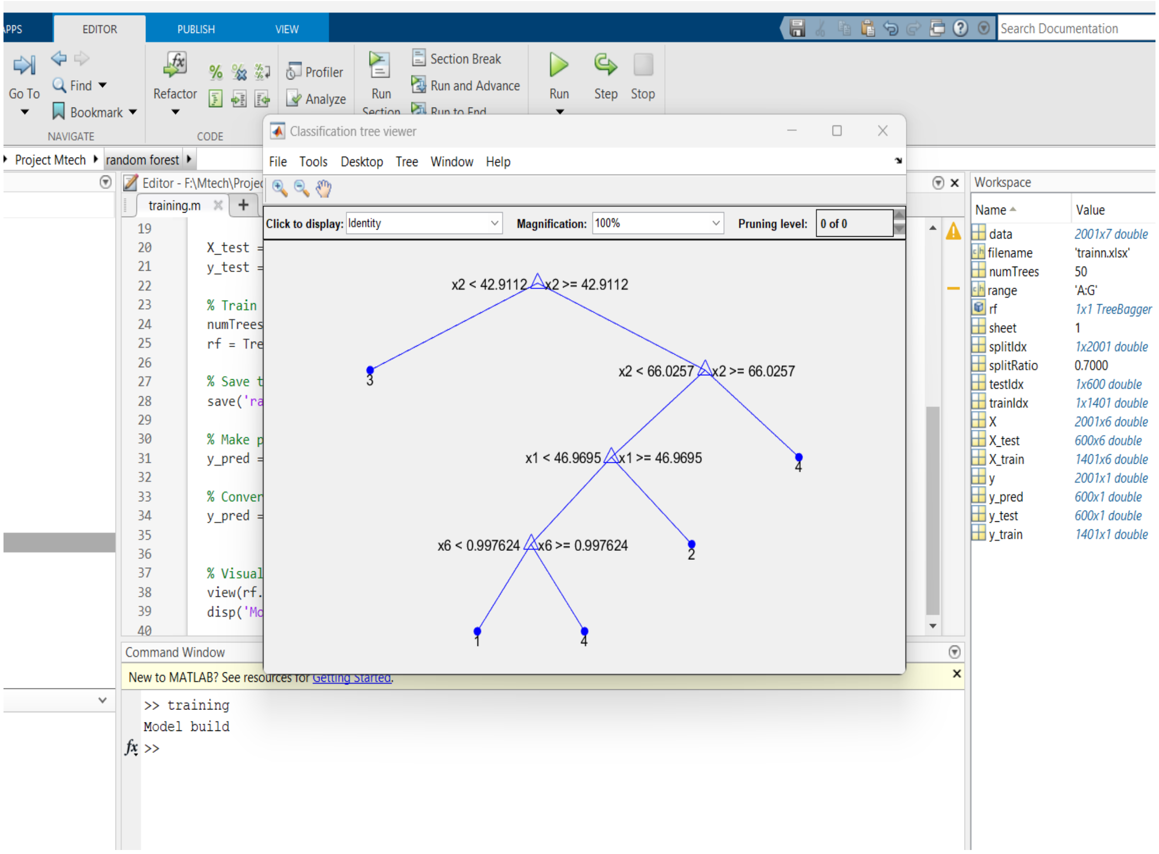
CLASSFICATION OF CNN:



```
disp(predicted_label);
```

```
|       8 |         450 |     00:00:13 |     100.00% |       0.1180 |          0.0010 |
|       9 |         500 |     00:00:14 |     100.00% |       0.1285 |          0.0010 |
|       9 |         550 |     00:00:15 |     100.00% |       0.0755 |          0.0010 |
|      10 |         600 |     00:00:16 |     100.00% |       0.0808 |          0.0010 |
|      10 |         620 |     00:00:16 |     100.00% |       0.0553 |          0.0010 |
|=========================================================================================|
Training finished: Max epochs completed.
>> cnn
Predicted Label:
     3

fx >>
```

TRAINING OF RF :



Click to display: Identity

Magnification: 100%

Pruning level: 0 of 0

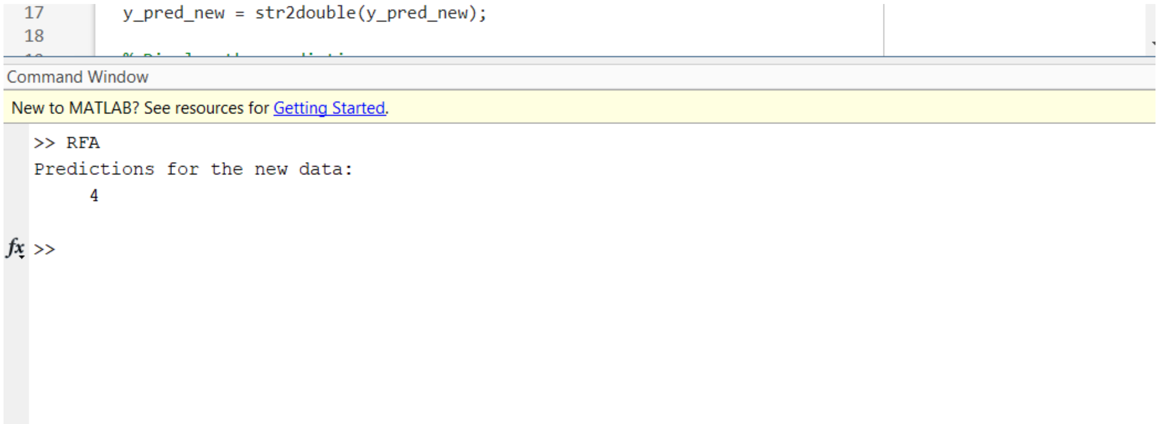
```

X_test =
y_test =

% Train
numTrees
rf = Tre

% Save t
save('ra

% Make p
y_pred =

% Conver
y_pred =


% Visual
view(rf.
disp('Mo

```

New to MATLAB? See resources for Getting Started.

```
>> training
Model build
>>
```

Name	Value
data	2001x7 double
filename	'trainn.xlsx'
numTrees	50
range	'A:G'
rf	1x1 TreeBagger
sheet	1
splitIdx	1x2001 double
splitRatio	0.7000
testIdx	1x600 double
trainIdx	1x1401 double
X	2001x6 double
X_test	600x6 double
X_train	1401x6 double
y	2001x1 double
y_pred	600x1 double
y_test	600x1 double
y_train	1401x1 double

CLASSFICATION OF RF:



```
y_pred_new = str2double(y_pred_new);
```

```
>> RFA
Predictions for the new data:
     4

fx >>
```

APPENDIX B

Coding

```
%% Classification using KNN

load FFV
load SR

set(handles.edit1,'string',num2str(FF));

D=xlsread('trainn.xlsx');

training=D(:,1:6);

group =D(:,7);  % Class

C1= knnclassify(FF, training, group);
set(handles.edit2,'string',C1);

if C1 == 1
   A='ALL-L1';
   set(handles.edit3,'string',A);
elseif C1 == 2
   A='ALL-L2';
   set(handles.edit3,'string',A);
elseif C1 == 3
   A='AML-M2';
   set(handles.edit3,'string',A);
elseif C1 == 4
   A='AML-M5';
   set(handles.edit3,'string',A);
```

```
end

axes(handles.axes1);
imshow(NDR);

%Training svm model
% Assuming your data is in a CSV file named 'your_data.csv'
data = readtable('trainn.xlsx');

% Extract features and convert class labels to numeric
X = data(:, 1:6);  % Assuming the features are in the first six columns
Y = grp2idx(table2array(data(:, 7)));  % Assuming the class labels are in the seventh column

% Split the data into training and testing sets
c = cvpartition(Y, 'HoldOut', 0.2); % 80% training, 20% testing
XTrain = X(training(c), :);
YTrain = Y(training(c), :);
XTest = X(test(c), :);
YTest = Y(test(c), :);

% Train the multiclass SVM model
svmModel = fitcecoc(XTrain, YTrain);

save svmModel
disp('model build successfully');

% Assuming your data is in a CSV file named 'your_data.csv'
data = readtable('trainn.xlsx');

% Extract features and convert class labels to numeric
```

```
X = data(:, 1:6);  % Assuming the features are in the first six columns
Y = grp2idx(table2array(data(:, 7)));  % Assuming the class labels are in the seventh column

% Split the data into training and testing sets
c = cvpartition(Y, 'HoldOut', 0.2); % 80% training, 20% testing
XTrain = X(training(c), :);
YTrain = Y(training(c), :);
XTest = X(test(c), :);
YTest = Y(test(c), :);

% Train the multiclass SVM model
svmModel = fitcecoc(XTrain, YTrain);

save svmModel

% Load the new data
newData = readtable('new.csv');

% Extract features
XNew = newData(:, 1:6);  % Assuming the features are in the first six columns

% Make predictions on the new data
newPredictions = predict(svmModel, XNew);

if newPredictions == 1
   newPredictions='ALL-L1';

elseif newPredictions == 2
   newPredictions='ALL-L2';
```

```
elseif newPredictions == 3
   newPredictions='AML-M2';

elseif newPredictions == 4
   newPredictions='AML-M5';

end
% Display the predicted classes
disp('Predicted Classes for New Data:');
disp(newPredictions);



%Training for RF model.
% Load data from the spreadsheet
filename = 'trainn.xlsx';
sheet = 1; % Assuming the data is in the first sheet
range = 'A:G'; % Assuming your data is in columns A to G
data = xlsread(filename, sheet, range);

% Extract features (columns 1 to 6) and labels (column 7)
X = data(:, 1:6);
y = data(:, 7);

% Split the dataset into training and testing sets
splitRatio = 0.7;
splitIdx = randperm(length(y));
trainIdx = splitIdx(1:round(splitRatio * length(y)));
testIdx = splitIdx(round(splitRatio * length(y)) + 1:end);

X_train = X(trainIdx, :);
y_train = y(trainIdx);
```

```
X_test = X(testIdx, :);
y_test = y(testIdx);

% Train a Random Forest using TreeBagger
numTrees = 50;
rf = TreeBagger(numTrees, X_train, y_train, 'Method', 'classification');

% Save the trained model
save('random_forest_model.mat', 'rf');

% Make predictions on the test set
y_pred = predict(rf, X_test);

% Convert predictions to numeric array
y_pred = str2double(y_pred);



% Visualize one of the trees in the forest
view(rf.Trees{1}, 'Mode', 'graph');
disp('Model build');
% Load the saved model
load('random_forest_model.mat');

% Load new data from the new spreadsheet
new_data_filename = 'new.csv';
new_data_sheet = 1;
new_data_range = 'A:F'; % Assuming your new data has 6 features in columns A to F
new_data = xlsread(new_data_filename, new_data_sheet, new_data_range);

% Extract features from the new data
```

```
X_new = new_data;

% Make predictions using the loaded model
y_pred_new = predict(rf, X_new);

% Convert predictions to numeric array
y_pred_new = str2double(y_pred_new);

% Display the predictions
disp('Predictions for the new data:');
disp(y_pred_new);

%model for CNN
% Load data from Excel file
data = xlsread('trainn.xlsx');

% Extract features (columns 1 to 6)
features = data(:, 1:6);

% Extract class labels (7th column)
labels = data(:, 7);

% Ensure that labels are column vectors
labels = labels';  % Transpose labels to make it a column vector

% Convert labels to categorical
categorical_labels = categorical(labels);

% Reshape features to be compatible with fully connected layers
features = reshape(features', [1, 1, size(features, 2), size(features, 1)]);
```

```
% Define CNN architecture
layers = [
   imageInputLayer([1 1 6]) % Assuming 6 features
   fullyConnectedLayer(64)
   reluLayer
   fullyConnectedLayer(32)
   reluLayer
   fullyConnectedLayer(4) % Adjust the number of neurons to match the number of classes
   softmaxLayer
   classificationLayer
];

% Set the training options
options = trainingOptions('sgdm', ...
   'MaxEpochs', 10, ...
   'MiniBatchSize', 32, ...
   'InitialLearnRate', 0.001, ...
   'Plots', 'training-progress');

% Train the CNN
net = trainNetwork(features, categorical_labels, layers, options);

% Save the trained neural network model
save('mymodel.h2.mat', 'net');

% Load the trained model
loaded_model = load('mymodel.h2.mat');
net = loaded_model.net;

% Load test data from Excel file
```

```
test_image = xlsread('new.csv');

% Reshape the test image for prediction
test_image = reshape(test_image, [1, 1, 6]);
% Make predictions using the trained network

predicted_label = classify(net, test_image);

% Display the predicted label
disp('Predicted Label:');
disp(predicted_label);
```