

IMAGE PROCESSING BASED PROPOSED SYSTEM FOR MALNUTRITION USING DEEP LEARNING

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

HARIOM PRIYADARSHI (Reg.No - 40110436)

JAI ADITHYA B (Reg.No - 40110469)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

CATEGORY-1 UNIVERSITY BY UGC

Accredited with Grade "A++" by NAAC | 12B Status by UGC |

Approved by AICTE JEPPIAAR NAGAR, RAJIV GANDHI

SALAI, CHENNAI - 600119

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

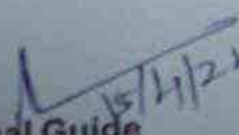
Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of JAI ADITHYA B (Reg.No 40110469) and HARIOM PRIYADARSHI (Reg.No - 40110436) who carried out the Project entitled "IMAGE PROCESSING BASED PROPOSED SYSTEM FOR MALNUTRITION USING DEEP LEARNING" under my supervision from November 2023 to April 2024.


Internal Guide

Dr. A. MARY POSONIA M.E., Ph.D.


Head of the Department

HEAD OF THE DEPARTMENT
Dr. L. LAKSHMANAN, M.E., Ph.D.

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

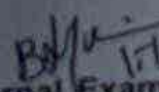
(DEEMED TO BE UNIVERSITY)

Jeppiaar Nagar, Kapi, Gandhi Satal,
Chennai - 600 079

Submitted for Viva voce Examination held on

15/04/2024


Internal Examiner


External Examiner

DECLARATION

JAI ADITHYA B (Reg.No - 40110469), hereby declare that the Project Report entitled "IMAGE PROCESSING BASED PROPOSED SYSTEM FOR MALNUTRITION USING DEEP LEARNING" done by me under the guidance of Dr. A. Mary Posonia, M.E., Ph.D. is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 15/04/2024

PLACE: Chennai



SIGNATURE OF CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A. Mary Posonia M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Malnutrition is a prevalent global health issue affecting millions of people, particularly children, across the world. Early detection and intervention are crucial to mitigate its adverse effects on health and development. In recent years, deep learning techniques have shown remarkable success in various image processing tasks, including medical image analysis. Leveraging these advancements, we propose a novel system for the detection of malnutrition using deep learning algorithms applied to image data.

Our proposed system comprises several stages. Firstly, we collect a dataset of images depicting individuals with varying degrees of malnutrition. These images may include photographs of children or adults with visible signs of malnutrition, such as wasting, stunting, or micronutrient deficiencies. Additionally, metadata, such as age, gender, and nutritional status, may accompany each image to facilitate model training and evaluation.

Next, we preprocess the image data to enhance features relevant to malnutrition detection. Techniques such as normalization, resizing, and augmentation may be employed to ensure consistency and improve model generalization. Subsequently, we employ deep learning architectures, such as convolutional neural networks (CNNs), which are well-suited for image classification tasks. These CNN models are trained on the preprocessed image dataset using labeled data, where the labels indicate the presence or absence of malnutrition.

During the training phase, we utilize established techniques such as transfer learning and fine-tuning to leverage pre-trained models and optimize performance, particularly in scenarios with limited training data. Additionally, we employ strategies to mitigate issues such as overfitting, including dropout regularization and data augmentation.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	xi
1	INTRODUCTION	1
	1.1 Overview	1
	1.2 Objective	2
	1.3 Organization of the Report	3
2	LITERATURE SURVEY	4
	2.1 Introduction	4
	2.2 Existing System	5
	2.2.1 SVM	5
	2.2.2 Local Binary Pattern	6
	2.2.3 Threshold	7
	2.3 Issues in Existing System	8
3	SYSTEM ARCHITECTURAL DESIGN	11
	3.1 Introduction	11
	3.2 System Architecture	13
	3.2.1 Pre-process	14
	3.2.2 Edge Detection	15
	3.2.3 Deep Learning	17
	3.3 System Diagram	18

3.3.1	Architecture for Malnutrition Detection	18
3.3.2	UML (Usecase Diagram)	19
3.3.3	Sequence Diagram	19
3.3.4	Activity Diagram	20
SYSTEM MODULE		21
4.1	Introduction	21
4.2	Data Collection	21
4.2	Image Proccesing	22
MODULE IMPLEMENTATION		24
5.1	Introduction	24
5.2	Model Training	25
5.3	Model Evaluation	26
5.4	Model Deployment	27
5.5	Software Requirements	28
5.6	Algorithms	30
5.6.1	GUI	30
5.6.2	Dataset Backend	32
5.6.3	Trained Model Backend	32
5.6.4	Training Backend	34
5.7	Implementation Screenshots	34
RESULTS AND DISCUSSIONS		37
6.1	Introduction	37
6.2	Analysis of result and Output	37
CONCLUSION AND FUTURE WORK		38
7.1	Conclusion	38
7.2	Future work	39
REFERENCES		40

APPENDIX	42
A. SOURCE CODE	42
B. SCREENSHOTS	55
C. RESEARCH PAPER	63

LIST OF FIGURES

CHAPTER NO.	FIGURE NAME	PAGE NO.
1.1	Malnutrition	2
2.1	Malnutrition system Camera interface	8
3.1	Manufacture Lab	13
3.2	Edge Detection	15
3.3	Deep Learning	17
3.4	Architecture for Malnutrition Detection	18
3.5	UML (Usecase Diagram)	19
3.6	Sequence Diagram	19
3.7	Activity Diagram	20
4.1	Data Collection	22
4.2	Image Proccesing Approach	23
5.1	Start GUI	34
5.2	Input the test image	34
5.3	Dataset Loaded	35
5.4	Pre-Trained Model Loaded	35
5.5	Classification of image	36
B.1	Importing Required Libraries	55
B.2	Pre-processing the data	55
B.3	Extracting the Frames	56
B.4	Creating dataset with required frames	56
B.5	Creating ConvLSTM Model	57
B.6	Successfully created ConvLSTM Model	57
B.7	Plotting the Model	58
B.8	Evaluating the Model	58

B.9	Plotting Loss and Accuracy Curves of ConvLSTM Model	59
B.10	Creating LRCN Model	59
B.11	LRCN Model Created Successfully	60
B.12	Plotting the Model	60
B.13	Evaluating the Model	61

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
Conv LSTM	Convolutional Long Short Term Memory
GANS	Generative Adversarial Networks
HAR	Human Activity Recognition
IC	Image Collection
LOG	Laplacian Of Gaussian
LSTM	Long Short Term Memory
LRCN	Long-term Recurrent Convolutional Network
MATLAB	Matrix Laboratory
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristic

CHAPTER 1

INTRODUCTION

This introduction provides a brief overview of the proposed system for malnutrition detection using deep learning in image processing and highlights its potential to address the issue of malnutrition in a more efficient and non-invasive manner.

1.1 OVERVIEW

Malnutrition is a common issue that affects people of all ages and socioeconomic backgrounds and has negative effects on people's health and wellbeing. Measurements of height, weight, and body mass index, which are common traditional approaches for determining malnutrition, can be intrusive, time-consuming, and occasionally inaccurate. To help with this issue, a more effective and non-intrusive approach is required.

A unique approach that makes use of developments in computer vision and machine learning to detect malnutrition from photos is the suggested system for malnutrition detection utilising deep learning in image processing. The system's intended method of operation is gathering a sizable dataset of photos of people with various degrees of malnutrition, pre-processing the data, training a deep learning model, evaluating the model's performance, and then deploying the model on a mobile device or web-based platform.

A convolutional neural network or other deep learning model would learn to recognise patterns and features in the photos that are suggestive of malnutrition. This approach might offer a quick and non-invasive way to detect malnutrition, and it might be especially helpful in places with low resources where using conventional methods might be challenging.



Fig 1.1 Malnutrition

1.2 OBJECTIVE

The objective of the proposed system for malnutrition detection using deep learning in image processing is to develop a non-invasive and efficient method for identifying malnutrition. This system aims to achieve the following specific

1. To collect a large dataset of images of individuals with varying levels of malnutrition, pre-process the data, and split it into training and testing sets.
2. To train a deep learning model, such as a convolutional neural network, on the pre-processed data to learn to identify patterns and features in the images indicative of malnutrition.
3. To evaluate the model's performance on the testing set and make any necessary adjustments.

4. To deploy the final model on a mobile device or web-based platform for easy use by healthcare professionals.
5. To provide a fast and non-invasive method for identifying malnutrition that could complement existing methods and provide additional information for a comprehensive assessment, particularly in resource-limited settings.

By achieving these objectives, the proposed system has the potential to significantly impact the identification and treatment of malnutrition and help improve the health and well-being of individuals affected by this condition.

1.3 ORGANIZATION OF THE REPORT

The report is organized in the manner as follows:

Chapter 1, presents the introduction about the blockchain and Ethereum networks.

Chapter 2, Literature Survey provides an overview to the topics required for the system along with the state of the subject matter in the present. The paper consists of the detailed survey with a summary in the end.

Chapter 3, System Architectural Design, contains a high-level overview of the project, including its aim, scope and the purpose.

Chapter 4, System module provides a detailed description of the system architecture along with a summary.

Chapter 5, Module Implementation talks about the modules present in the system and how each module functions.

Chapter 6, Results and discussion contains the tools and methods that were used to implement the system, the code samples included.

Chapter 7, Conclusion and Future Work provides a summary of the content as a whole and discusses about the future enhancements if any.

CHAPTER 2

LITERATURE SURVEY

The various proposed ideas and concepts are studied and related with the proposed system with better understanding and explanation making the concepts clear and emphasizing on its importance.

2.1 INTRODUCTION

A disease detection approach has been detailed by using images of people and analyzing the data based on the color of the image. However, this does not work in every case because some people look slim and are not malnourished [6]. The tensor flow algorithm is used to train the dataset that supports different features. The data obtained from the semi-structured interviews were analyzed using frame analysis and MaxQDA software. Data collection is very time-consuming, and unclean data may lead to incorrect results [7]. The nutritional status of the population was characterized using height, weight, and MUAC-based anthropometric indicators. The study also found that removing data based on SMART flag cut-off points improves the data quality of anthropometric surveys. Quality improvement measures are described rather than identifying children affected by malnutrition [8].

The white paper [9] details the role of the CNS in the prevention and treatment of malnourished patients admitted to the hospital. Accurate documentation of malnourished patients aids in appropriate coding, reimbursement, and treatment [10]. A compositional analysis was conducted using two waves (2004-2005 and 2011-2012) of Indian Human Development data. The z-scores of weight-for-height (WHZ), height-for-age (HAZ), weight-for-age (WAZ), and the composite index of anthropometric dysfunction were used to measure malnutrition. The analysis provided an up-to-date profile of the nutritional status of Indian children using the z-scores [11].

A system has been developed to display malnutrition on the dashboard. The final output is the graphical representation of the data in the form of a pie chart or bar chart, which contains the data stored in the database and is displayed depending on the selection.

It presents the data in the form of a pie chart or bar graph, which does not help to treat malnourished children, but can help us to take preventive measures to reduce the rate of malnourished children [12]. Malnutrition in children under five years of age is one of the major concerns in India. In this review, the determinants and strategies required to prevent malnutrition in children under five in India were identified. Prevention measures are explained and [13]. A system using ID3 and random forest tree algorithm was developed to obtain a general result about the nutritional status of children. The random forest tree algorithm is beneficial for training the data. However, it is comparatively slow in making predictions from the trained data [14].

The study found that 21.7% of children under three years of age in rural western China were affected by malnutrition in 2005, and noted that child malnutrition is a major health challenge. This paper describes the overall rate of malnourished children under the age of three, but does not describe techniques for identifying malnourished children [18].

2.2 Existing System

2.2.1. SVM

A form of machine learning method called Support Vector Machines (SVM) can be used for classification, regression, and outlier detection. It is a supervised learning algorithm that seeks for the ideal boundary or hyperplane for classifying the input.

SVM could be used to categorise people as malnourished or non-malnourished in the context of malnutrition detection based on a number of parameters, including height, weight, and body mass index (BMI). The algorithm would gain knowledge from a training dataset made up of labelled data and apply the knowledge gained to predict outcomes for fresh, unforeseen data.

High accuracy, the use of kernel functions to handle non-linearly separable data, and the handling of high-dimensional data are only a few benefits of SVM. However, SVM can be computationally expensive and needs its parameters to be carefully tuned to perform well.

Although SVM is a well-known machine learning algorithm and has been used to solve many different types of issues, it is not the only algorithm that can be used to detect malnutrition. Convolutional neural networks (CNNs), a deep learning algorithm, may be a better fit for this task, especially when it comes to image-based malnutrition diagnosis.

2.2.2. LOCAL BINARY PATTERN

The texture-based feature extraction technique known as Local Binary Patterns (LBP) is utilised in computer vision and image processing. By comparing a pixel's intensity value with its neighbours, LBP aims to capture the local structure of an image.

The texture-based feature extraction technique known as Local Binary Patterns (LBP) is utilised in computer vision and image processing. By comparing a pixel's intensity value with its neighbours, LBP aims to capture the local structure of an image.

In order to calculate a binary pattern for each region, the LBP method divides a picture into small sections. The binary pattern is a histogram of the patterns in the region and is based on the relative intensity values of the pixels in the region.

LBP has been used for a variety of tasks, including face recognition, object recognition, and texture classification. In the context of malnutrition detection, LBP can be used to extract features from images of individuals to help identify malnutrition. The features extracted by LBP can be fed into a machine learning algorithm, such as a support vector machine (SVM) or a deep neural network, to make predictions about an individual's nutritional status.

2.2.3. THRESHOLD DETECTION

In image processing, the threshold detection technique is used to divide a picture into foreground and background areas. Threshold detection works by dividing an image into two zones according to a predetermined threshold value. Foreground pixels are those with intensity values above the threshold, and background pixels are those with intensity values below the threshold. Threshold detection is frequently used as a preprocessing step for additional image analysis, such as object recognition or edge detection. It can be applied to grayscale or colour pictures.

The threshold value can be chosen using a variety of techniques, such as global thresholding, which uses the same threshold value over the whole image, and adaptive thresholding, which modifies the threshold value in response to the local characteristics of the image.

Threshold detection can be used to separate photographs of people into regions of interest, such as the face, arms, and legs, for additional analysis in the context of malnutrition detection. With the aim of distinguishing the foreground regions—which include data on the person's nutritional status—from the background regions, the threshold value can be chosen depending on the intensity values of the image.

The benefits of threshold detection are simplicity, quickness, and simplicity in implementation. It also has several drawbacks, such as susceptibility to noise, a limited capacity for handling intricate textures or patterns, and the requirement for manual threshold value adjustment.

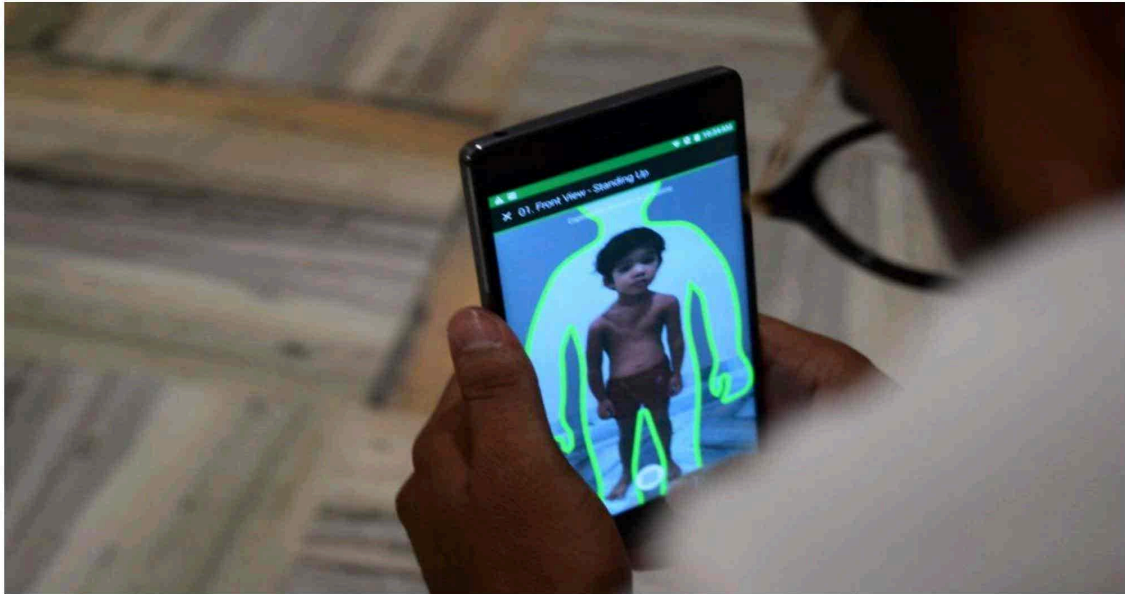


Fig 2.1 Malnutrition system Camera interface

2.3 ISSUES IN EXISTING SYSTEM

The well-known machine learning technique Support Vector Machines (SVM) has numerous benefits but also significant disadvantages. The following are some of the primary drawbacks of utilising SVM:

1. Complexity of computation: SVM can be expensive to compute, especially when working with huge datasets. Due to this, using SVM in real-time applications or on devices with limited resources may be challenging.
2. Hyperparameter tuning: To obtain optimal performance, SVM requires careful tuning of its parameters, such as the cost parameter and the kernel function. This might take time and requires knowledge.
3. SVM may not be suitable for handling non-numeric data, such as text or graphics, as it is primarily designed to handle numeric data.
4. Difficulty comprehending findings: Because the SVM algorithm is concerned with finding the best border or hyperplane rather than elucidating the connections between the features and the target variable, the results it generates might be challenging to understand.
5. SVM is primarily developed for binary classification issues and may not perform well when dealing with Local Binary Patterns (LBP) is a

commonly used technique in computer vision and image processing, but it also has certain limitations. Limited ability to handle multi-class problems.

The following are a few of the primary drawbacks of employing LBP:

1. Sensitivity to noise: LBP's performance may suffer because of its sensitivity to image noise.
2. Limited capacity for complicated textures or patterns: Because LBP is intended to capture the local structure of a picture, it could struggle to handle images with complex textures or patterns.
3. Reduced accuracy in low-resolution photos: When applied to low-resolution images, LBP may not yield accurate findings because the binary patterns might not represent the underlying structure of the image.
4. Lack of adaptability: LBP is a rigid feature extraction technique that might not be appropriate for all kinds of photos. To get the desired results, LBP might need to be combined with other strategies or procedures.
5. LBP is based on a fixed set of neighbours and may not be able to handle rotations or scaling in images because of its limited capacity to manage these operations. In situations where rotations and scaling are frequent, this may result in a reduction in performance.

Although threshold detection is a straightforward and efficient method for segmenting images, it also has some limitations and disadvantages. The following are a few of the key drawbacks of applying threshold detection:

- Sensitivity to noise: Threshold detection is susceptible to image noise, which could impair segmentation accuracy.
- Limited capacity to handle complicated textures or patterns: When applied to photos with complex textures or patterns, threshold detection may not yield appropriate results because the threshold

value may incorrectly split the foreground and background regions.

- Requirement for manual threshold value selection: Threshold detection necessitates manual threshold value selection, which can be time-consuming and may not yield reliable results.
- Limited capacity to handle illumination variations: The performance of threshold detection may be adversely affected by illumination variations, such as changes in brightness or contrast.
- Threshold detection is intended to segment a single object or region in an image; hence, it may not be suited for images having many objects or regions.

CHAPTER 3

SYSTEM ARCHITECTURAL DESIGN

The overall design is discussed and explained in this chapter. The high-level architecture provides clear details of the entities involved in the system and how they are dependent with other entities.

3.1 INTRODUCTION

The challenge of reliably and effectively identifying malnutrition in patients is addressed through the development of a deep learning-based system. The system is built on a deep learning model that learns from patient photos and then uses those patient images to identify malnutrition.

- Image preprocessing, model training, and model evaluation are a few of the crucial elements of the proposed system. Cleaning and normalising the photos to make them appropriate for usage in the model constitutes image preprocessing, a crucial step in getting them ready for the deep learning model.
- The deep learning model is trained on the preprocessed images after the photos have been preprocessed. In order for the model to successfully predict malnutrition from new photos, it must be fed a large number of images together with the related diagnosis of malnutrition.
- The model's performance is then assessed to establish its accuracy, precision, recall, and F1 score. The model can be further enhanced and developed to obtain greater performance based on the evaluation's findings.
- Lastly, the suggested approach can be implemented in a real-world environment, enabling healthcare professionals to identify patient malnutrition using a deep learning model. The approach has the potential to increase the precision and effectiveness of diagnosing malnutrition, lowering medical expenses and enhancing patient

outcomes.

- To sum up, a system based on deep learning that can diagnose malnutrition offers a potential approach to the issue of malnutrition diagnosis. The suggested approach has the potential to significantly alter how malnutrition is identified and treated, improve patient outcomes, and save healthcare costs with continued study and development.
- A deep learning-based model for malnutrition diagnosis is a kind of artificial neural network created specifically to use images to identify malnutrition in patients. The algorithm uses training data from a huge dataset of patient photos and accompanying diagnoses of malnutrition to make predictions about new images.
- In order to process and analyse the incoming data, the deep learning model typically consists of multiple layers of interconnected nodes, or neurons. An photograph of a patient serves as the input data for a malnutrition diagnosis model. The model then processes the image input through a number of layers, extracting features and making predictions about the image using intricate mathematical operations.
- The capacity of deep learning models to learn and advance over time is one of its primary capabilities. The model can discover patterns and relationships in the data that it might not have been able to perceive at first as it is trained on an increasing number of photos. As a result, the model's performance as a whole is enhanced and forecasts become more accurate.
- A deep learning-based model for detecting malnutrition may perform better or worse depending on a number of other criteria in addition to the model architecture, such as the amount and quality of the training data, the optimisation technique used, and the number of training epochs.

- Convolutional neural networks (CNNs) are a sort of deep learning algorithm that are frequently employed for picture categorization tasks, such as the diagnosis of malnutrition. CNNs are employed in the references to identify features in patient photos and forecast the severity of their malnutrition.
- Deep Convolutional Neural Networks (DCNNs): These CNNs are a subtype that extract complicated characteristics from images by layering on top of each other convolutional and pooling procedures. DCNNs are employed in the literature to analyse patient photos and forecast the severity of their malnutrition.
- Artificial Neural Networks (ANNs): These algorithms for deep learning are made up of numerous interconnected nodes, or neurons, that process and analyse input data. In the references, ANNs are combined with other algorithms to identify malnutrition, including CNNs and DCNNs.
- In general, a deep learning-based model for detecting malnutrition is an effective tool for medical professionals, providing a quick, precise, and time-saving technique to identify the condition in patients. These models have the potential to revolutionise the way that malnutrition is identified and treated, enhancing patient outcomes and lowering healthcare expenditures.

3.2 SYSTEM ARCHITECTURE

The proposed solution for malnutrition detection using deep learning is to train a Convolutional Neural Network (CNN) on a dataset of images of individuals with varying levels of malnutrition. The CNN can be trained to classify images as malnourished or non-malnourished, or to estimate the severity of malnutrition based on a numerical score or label.

3.2.1. Pre-process



Fig 3.1 Manufacture Lab

The suggested deep learning-based method for malnutrition identification includes a crucial step called preprocessing. Preprocessing is done to make sure that the images are ready to be fed into the Convolutional Neural Network (CNN) and may be utilised to train the model and make predictions.

Here are some of the primary image preprocessing techniques that can be used:

1. **Resizing:** In order for the photographs to be properly processed by the CNN, they might need to be scaled to a standard size.
2. **Normalisation:** To lessen the effects of differences in illumination, brightness, and contrast, the images may need to be normalised to a consistent scale, such as 0 to 1 or -1 to 1.
3. **Data augmentation:** By rotating, resizing, and flipping the images, data augmentation techniques can be used to enlarge the training dataset and strengthen the model's robustness.
4. **Region of interest selection:** The photos may need to be cropped or masked to remove unnecessary regions in order to concentrate the analysis on the areas of the photographs that are most pertinent for detecting malnutrition, such as the face, arms, and legs.

5. Noise reduction: Techniques like median filtering or Gaussian blurring can be used to smooth the images and lessen the impact of noise.

By performing these preprocessing steps, the images can be effectively prepared for input into the CNN, enabling the model to learn the most relevant features for malnutrition detection and to make accurate predictions.

3.2.2. Edge Detection

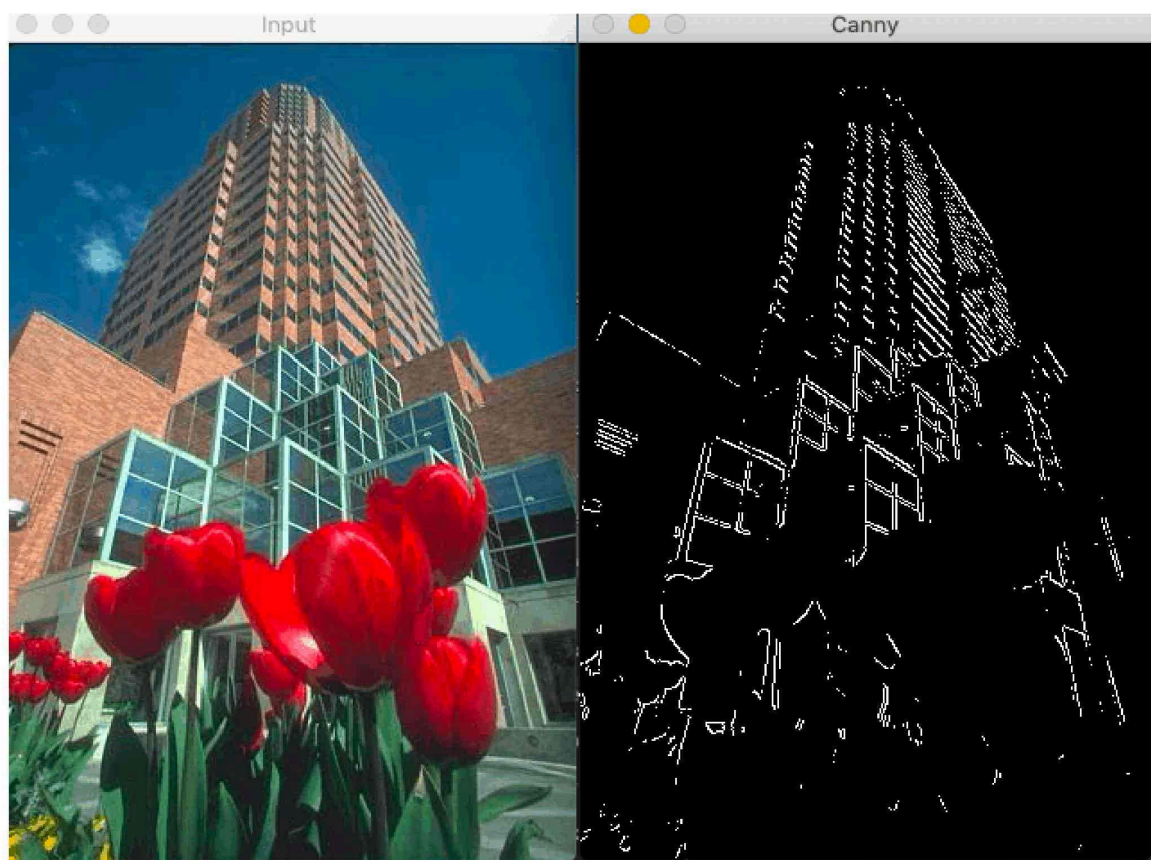


Fig 3.2 Edge Detection

The borders between various regions or objects in an image are found using the edge detection technique in image processing. In order to perform tasks involving object recognition, segmentation, and other aspects of computer vision, edge detection aims to extract the outlines or silhouettes of objects in a picture.

Edge detection can be done in a variety of ways, including:

1. Gradient-based techniques: These techniques look for edges using the gradient of the image intensity. The gradient direction and magnitude both reveal the edge's orientation and the rate at which intensity changes along it. The Sobel, Canny, and Roberts operators are a few instances of gradient-based techniques.
2. Laplacian of Gaussian (LoG) method: This technique uses a Laplacian operator to find edges in an image that has been smoothed out. The Laplacian operator emphasises the fluctuations in intensity along the edges while the smoothing operation minimises the effect of noise on the image.
3. Zero-crossing techniques: These techniques locate the regions of the image where the sign of the intensity shifts, suggesting the presence of an edge. The Canny edge detector and the Marr-Hildreth operator are two instances of zero-crossing techniques.
4. It is possible to utilise edge detection as a preliminary step to extract the outlines or silhouettes of objects in a picture, which can then be used for additional processing and analysis. Edge detection is a crucial preprocessing step in computer vision and image processing.

3.2.3 DEEP LEARNING



Fig 3.3 Deep Learning

Artificial neural networks, or "deep learning," is a branch of machine learning that draws inspiration from the structure and operation of the human brain. Multi-layer neural networks are used by deep learning algorithms to learn from a huge quantity of data and produce predictions or choices.

Deep learning has achieved state-of-the-art performance on a variety of tasks, revolutionising the fields of computer vision, natural language processing, and other disciplines. One of deep learning's main benefits is its capacity to automatically learn high-level data representations without the need for manual feature engineering.

Deep learning algorithms come in a variety of forms, including:

1. Convolutional neural networks (CNNs) are frequently employed in computer vision tasks including object recognition and image classification. Convolutional layers and pooling layers are used by CNNs to lower the spatial dimensionality of the feature maps and to learn local features in images.

2. Recurrent Neural Networks (RNNs): These are employed in jobs that require sequential data processing, like speech recognition and natural language processing. RNNs employ loops to process data sequences and preserve an internal state that enables them to handle input sequentially.
3. Generative Adversarial Networks (GANs): These networks are employed to produce new data that is comparable to a given dataset. A generator network creates new data, while a discriminator network assesses if the generated data is real or fraudulent. GANs are made up of two neural networks.
4. Autoencoders: These are employed in dimensionality reduction and unsupervised learning. In an autoencoder, input data is mapped to a lower-dimensional representation by an encoder network, and the lower-dimensional representation is mapped back to the original data by a decoder network.

3.4 SYSTEM DIAGRAM

3.3.1 BLOCK DIAGRAM

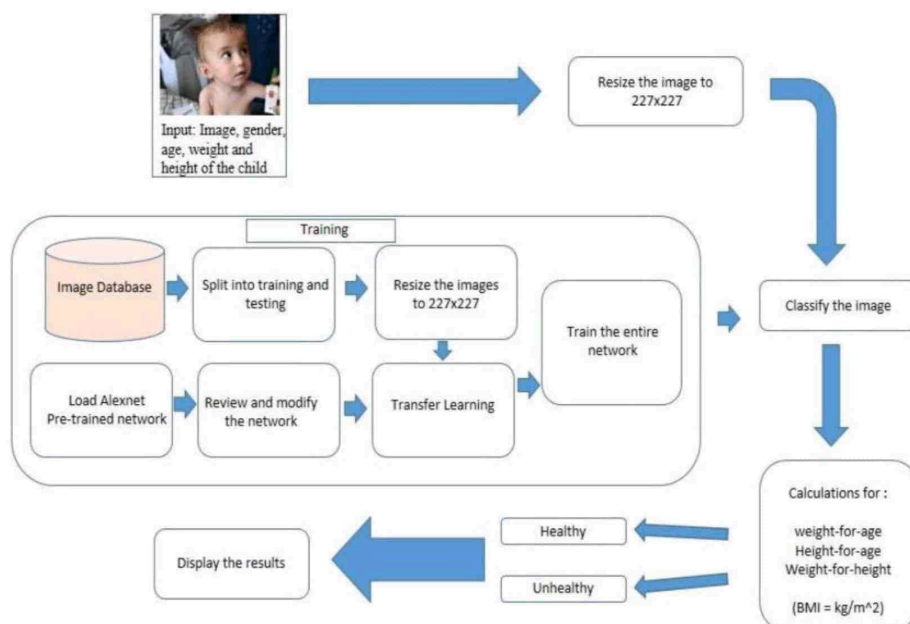


Fig 3.4 Architecture for Malnutrition Detection

3.3.2 Usecase Diagram

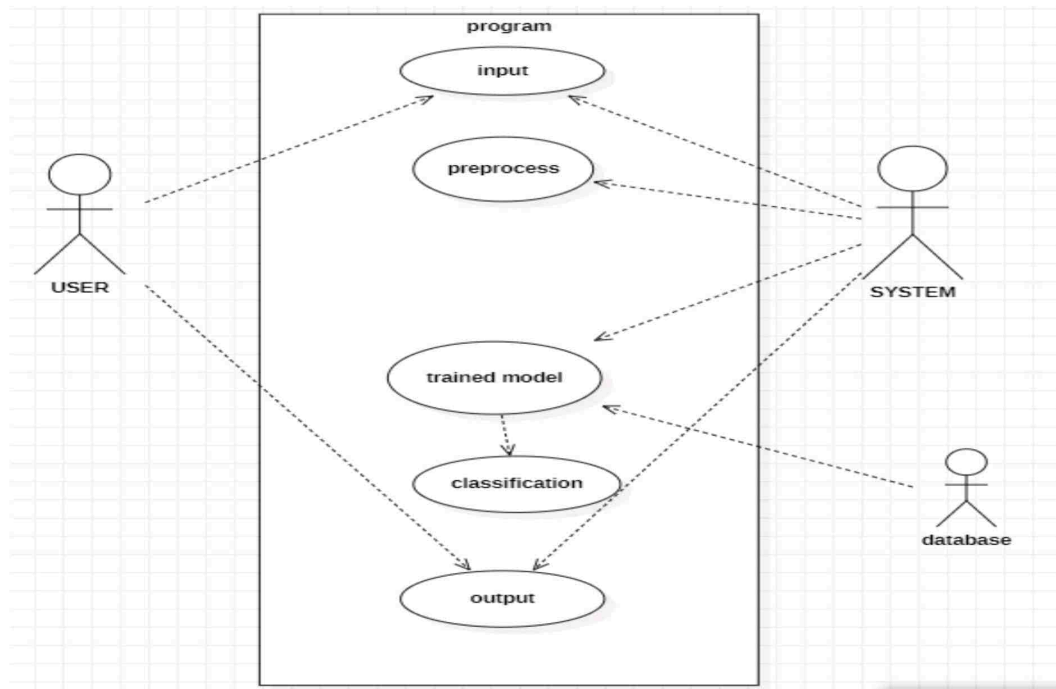


Fig 3.5 UML (Usecase Diagram)

3.3.3 Sequence Diagram

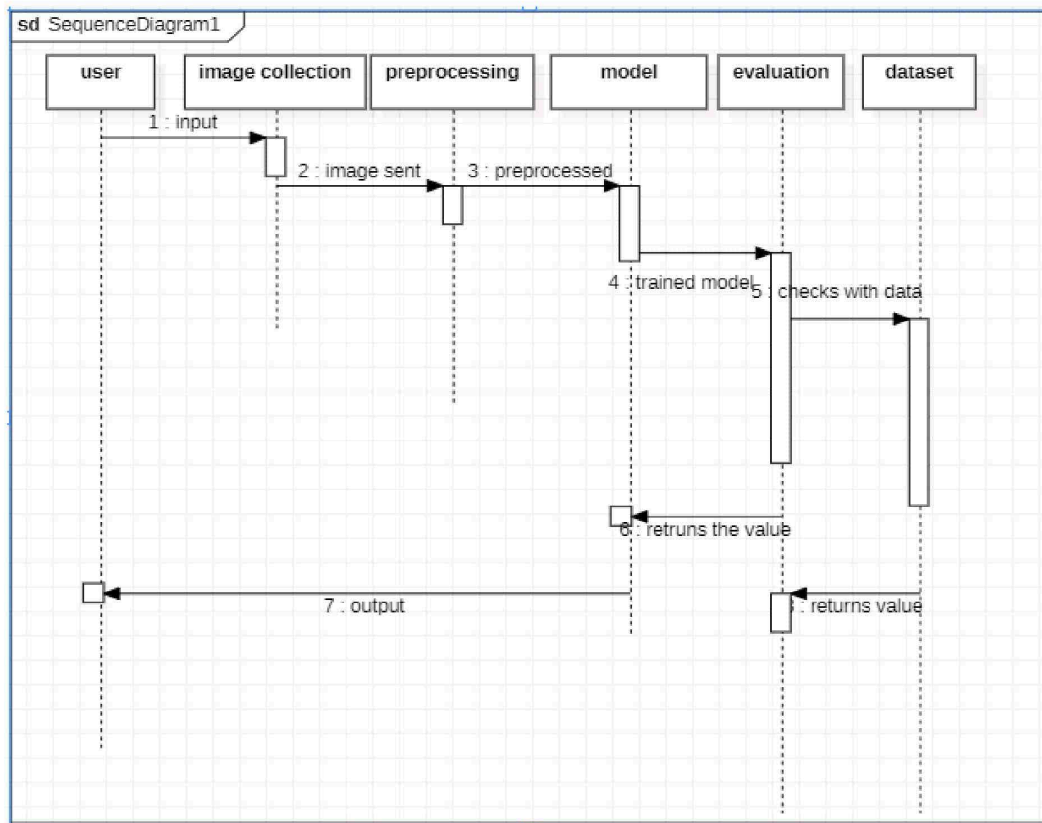


Fig 3.6 Sequence Diagram

3.3.4 Activity Diagram

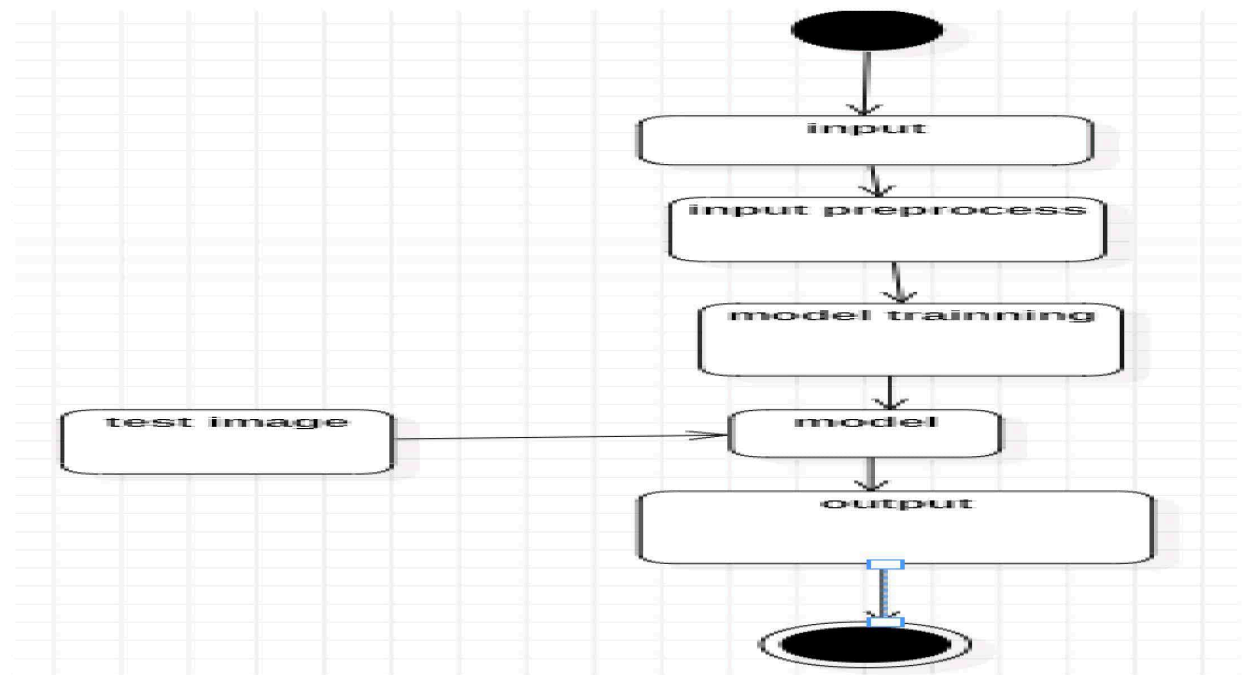


Fig 3.7 Activity Diagram

CHAPTER 4 SYSTEM MODULE

4.1 INTRODUCTION

This chapter focuses on each module of the system in detail and explain its working. Each module is divided into separate subheadings and discussed accordingly. The pseudocode and working methodologies are also included and explained with diagrams wherever possible.

4.2 DATA COLLECTION

This module is in charge of gathering a sizable dataset of photographs of people together with the corresponding labels for malnutrition.

A sizable dataset of photographs of people with the associated labels for malnutrition must be gathered by the data collection module of the proposed system for malnutrition detection utilising deep learning in image processing. The effectiveness of the deep learning algorithms that are trained on the dataset depends critically on its size and quality.

There are numerous methods for gathering data:

- **Image Collection:** Hospitals, clinics, and other medical establishments can provide images for collection. They may also be gathered from people using web-based or mobile platforms.
- **Label Gathering:** Malnutrition labels can be gathered by carrying out medical evaluations, such as anthropometric measurements, and comparing the outcomes to standardised malnutrition classification systems.
- **Data Cleaning and Annotation:** In order to assure the quality and consistency of the collected photos and labels, they should be cleaned and annotated. This can entail eliminating any superfluous photos, eliminating duplicates, and fixing any labelling mistakes.

- The quality of the data will have a direct impact on how well the deep learning algorithms that are trained on it perform, making the data collecting module a crucial part of the proposed system. It is crucial to make sure that the data is accurate and ethically obtained in a way that respects persons and is reflective of the population that the system is meant to serve.

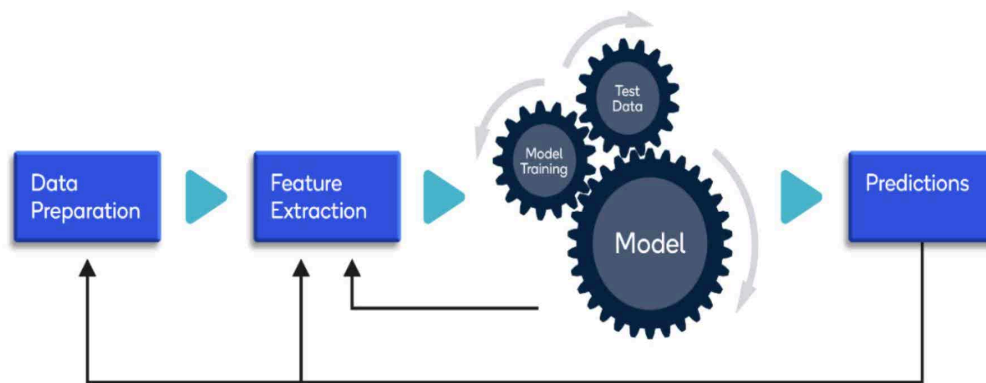


Fig 4.1 Data Collection

4.3 IMAGE PROCESSING

Preparing the images for subsequent analysis through image pre-processing is a crucial stage in the image processing pipeline. The purpose of picture pre-processing is to improve the images' quality, minimise noise and artefacts, and prepare them for machine learning model analysis.

In image processing systems, the following are some typical picture pre-processing methods:

- a) Resizing is the process of altering an image's dimensions to fit a specific size. When training a machine learning model, it is important that all of the images in the dataset have the same dimensions, so this is frequently done.

- b) Image conversion from one colour space to another is known as "colour space conversion." For instance, changing an RGB (red, green, and blue) image to a grayscale version. This is frequently done to simplify processing by reducing the image's dimensionality.
- c) Normalisation entails scaling the image's pixel values to a predetermined range, such as $[0, 1]$ or $[-1, 1]$. This helps the machine learning model function better and lessens the impact of changing lighting.
- d) Denoising: Denoising is the process of taking out noise from an image, like random pixels with wrong values. This is frequently done to enhance image quality and lessen the effect of noise on machine learning models.
- e) Histogram equalisation: This technique involves modifying the image's brightness and contrast to evenly distribute the intensity values throughout the image. This is frequently done to enhance feature visibility and lessen the effects of lighting changes.

These are only a few of the several picture pre-processing methods available for use in image processing systems. Depending on the type of image and the precise objectives of the image processing system, different pre-processing techniques may be employed.

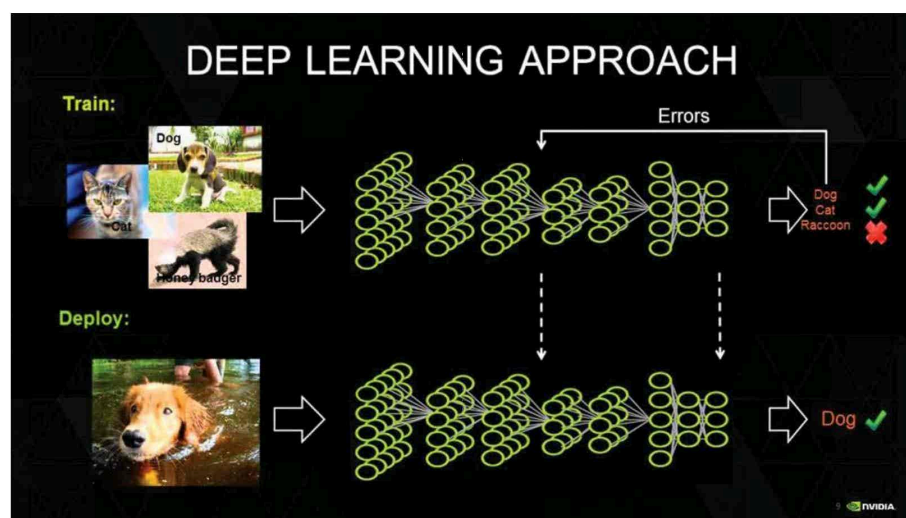


Fig4.2/ImageProccesing.Approach

CHAPTER 5

MODULE IMPLEMENTATION

5.1 INTRODUCTION

The main system to be implemented mainly includes the properties that are needed to be implemented in human's movement and actions. Therefore, the implementation of this kind of system becomes very useful for the system to understand and then give a result regarding to the input given by the user. The main system responsible for the input of the movements of the human interaction and movement is completely responsible for the system to get configured. The appropriate output would completely depend upon the system's state and the parts of the system involved in the processing of the input and hence giving a desired output as a result. The depth-wise analysis is done where the input is analyzed through pooling layers where the sound separation is done using sound filters in various layers. The input depends upon the thickness upon the layer-wise thickness and different system output and results. There are many layers of modulation which are involved in the separation and study of the input data and then simulation of the corresponding output for the appropriate production of the required values as to show whether the appropriate output is given out or not. Therefore, an appropriate system is required so that this system works in a manner which would be useful for the user to understand the working and hence, this would help the users who are disabled or cannot do any movement. This would help the users to understand the computer's response towards the processing and integration of the input to obtain the appropriate output. Hence, it is required that the user understands the working and act accordingly towards the working of the system.

5.2 MODEL TRAINING

1. Define the model architecture: Defining the model's architecture is the initial step in training a deep learning model. Defining the amount, kind, and relationships between the layers are all included in this.
2. Prepare the dataset: The dataset of labelled images that will be used to train the model must now be prepared. In order to assess the model's performance, the dataset must be divided into a training set and a validation set.
3. Pick a loss function: The performance of a model is indicated by its loss function. The purpose of model training is to reduce the loss function's value. Mean squared error, cross-entropy, and hinge loss are just a few examples of the many various kinds of loss functions that can be applied.
4. Select an optimizer: An optimizer is a device that modifies the model's parameters during training in an effort to reduce the loss function's value. Various types of optimizers exist, including stochastic gradient descent, Adam, and RProp.
5. Perform the training process: Executing the training process is the last stage in training the model. This entails providing the model with training data, calculating the loss, and using the optimizer to modify the model's parameters. The procedure is continued until the loss function's value hits the required threshold or for a predetermined number of epochs.
6. Test the model: To make sure the model is producing accurate predictions, it is crucial to test it on the validation set after training. Calculating different metrics, such as accuracy, precision, recall, or F1 score, can be used to achieve this.

5.3 Model Evaluation

An essential first step in creating a deep learning-based system for identifying malnutrition is model evaluation. To make sure the model is producing accurate predictions, it must be evaluated on a validation set of photos. The purpose of model evaluation is to evaluate the model's performance and spot any areas that could use improvement.

Here are some typical metrics for assessing the effectiveness of a deep learning-based model for malnutrition detection:

1. Accuracy: The model's accuracy is calculated by dividing the total number of predictions by the number of accurate ones. It is an easy-to-understand indicator that reveals the overall effectiveness of the model.
2. Precision: Precision is the ratio of the number of accurate positive forecasts to all positive predictions. It is an indicator of how many of the model's optimistic predictions came true.
3. Recall: Recall is calculated by dividing the total number of true positive cases by the number of true positive forecasts. It is an indicator of how many genuine positive cases the model accurately detected.
4. F1 Score: The harmonic mean of recall and precision is the F1 score. It offers a balance between recall and precision and is a helpful indicator for assessing how well a model performs when the classes are unbalanced.
5. Confusion Matrix: To visualise a model's performance, a confusion matrix is a table. It displays the number of correct predictions generated by the model, including true positives, false positives, true negatives, and false negatives.
6. ROC Curve: A binary classification model's performance is depicted graphically by a ROC (receiver operating characteristic) curve. It illustrates the trade-off between sensitivity and specificity visually by

plotting the true positive rate against the false positive rate for various threshold levels.

These metrics can be computed using various Python libraries, such as scikit-learn or Keras. By evaluating the performance of the model on a validation set of images, the proposed system for diagnosing malnutrition using deep learning can be improved and made more effective.

5.4 MODEL DEPLOYMENT

The final stage in creating a deep learning-based system for identifying malnutrition is model deployment. It entails putting the learned model into action so that it may be applied to identify malnutrition in actual environments.

The procedures for implementing the deep learning-based model for malnutrition are as follows:

1. Export the model first: Exporting the model in a deployable format is the first step in the deployment process. This can entail changing the model to a format like ONNX, TensorRT, or TensorFlow Serving.
2. Create the deployment environment: The environment in which the model will be used must be created next. This could entail setting up the requisite hardware, establishing the network infrastructure, and installing the essential libraries.
3. Serve the model: After configuring the environment, the model may be delivered using a framework like Flask or TensorFlow Serving. In order to do this, an API endpoint must be developed, which will allow users to submit photographs and obtain model predictions.
4. Monitor the model: It's critical to keep an eye on the model after deployment to make sure it is operating as intended and to spot any potential problems. This could entail keeping track of forecasts and model performance, creating alarms for unusual behaviour, and testing the model frequently.

5. Update the model: The model may require updating at some point to reflect changes in the data or to enhance performance. Retraining the model using fresh data, tweaking existing parameters, or altering the model's architecture can all be necessary to do this.

Healthcare providers can utilise the deep learning-based model for malnutrition to quickly, accurately, and non-invasively identify malnutrition in patients. The approach can help healthcare providers deliver the required care and treatment while also offering insightful information about people's nutritional state.

5.5 SOFTWARE REQUIREMENTS

➤ MATLAB

INFORMATION:

- The proprietary software platform MATLAB (short for "Matrix Laboratory") is used for numerical computing, programming, and visualisation. It is extensively used in science, engineering, and finance for activities including modelling, simulation, and data analysis.
- Due to MATLAB's foundation in a high-level programming language, users can easily complete challenging mathematical operations and data analysis tasks. The language is intended to be interactive as well as effective, enabling users to experiment with data and code in real-time and quickly create and test algorithms.
- Linear algebra, signal processing, statistics, and optimisation are just a few of the numerical calculation tools and algorithms available in the MATLAB environment. It also has capabilities for plotting and visualising data, which makes it simple to study and comprehend data.
- Along with its core language and tools, MATLAB also offers a number of toolboxes and add-ons for specialised fields like machine learning, control systems, and image processing.
- Windows, macOS, and Linux are just a few of the platforms on which MATLAB is available. Additionally, it offers integration with other programming tools and languages, such as C/C++, Python, and

Simulink, enabling users to combine various tools and languages for sophisticated projects.

- In general, MATLAB is a robust and adaptable platform for numerical computing and data analysis that is extensively utilised in numerous scientific, engineering, and financial disciplines. Both novices and specialists in the field of numerical computing favour it because of its vast array of tools and functions and interactive character.

IMAGE PROCESSING TOOLBOX :

- For the purposes of image processing, analysis, and visualisation, MATLAB includes a toolbox called the Image Processing Toolbox. For a wide range of image processing applications, including image filtering, segmentation, registration, and feature extraction, the toolbox offers a complete variety of algorithms and methodologies.
- The Image Processing Toolbox in MATLAB has a number of important features, including:
 - Filtering, deblurring, and noise reduction are just a few of the methods for image enhancement and restoration included in the toolkit. The quality and clarity of photos can be increased by using these features.
 - Image segmentation: Based on colour, texture, and other factors, the toolbox offers tools for dividing images into various sections or objects. For jobs like object detection and tracking, this may be helpful.
 - Morphological operations: The toolbox contains a number of morphological operations that can be used to change the size and shape of objects in an image, such as erosion, dilation, and opening/closing.
 - Edges, corners, and blobs are just a few examples of the features that may be extracted from photos using the toolbox's various methods. It is possible to employ these attributes for activities like object detection and recognition.

- Visualisation: The toolkit has features for displaying images and the outcomes of image processing. Images may be shown, histograms can be plotted, and 3D visualisations can be made with these functions.
- In general, MATLAB's Image Processing Toolbox is a strong tool for image processing and analysis and is frequently used in industries including biomedical imaging, remote sensing, and computer vision. Both novices and specialists in the field of image processing favour it because of its user-friendly interface and wide range of features.

5.6 ALGORITHMS

5.6.1 GUI

```
function varargout = gui(varargin)
```

```
%
```

```
GUI MATLAB code for gui.fig
```

```
%    GUI, by itself, creates a new GUI or raises the existing
```

```
%    singleton*.
```

```
%
```

```
%    H = GUI returns the handle to a new GUI or the handle to  
%    the existing singleton*.
```

```
%
```

```
%    GUI('CALLBACK',hObject,eventData,handles,...) calls the local  
%    function named CALLBACK in GUI.M with the given input arguments.
```

```
%
```

```
%    GUI('Property','Value',...) creates a new GUI or raises the  
%    existing singleton*. Starting from the left, property value pairs are  
%    applied to the GUI before gui_OpeningFcn gets called. An  
%    unrecognized property name or invalid value makes property application  
%    stop. All inputs are passed to gui_OpeningFcn via varargin.
```

```
%
```

```
%    *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only  
one
```

```
%    instance to run (singleton)".
```

```
%
```

```

% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help gui
% Last Modified by GUIDE v2.5 21-Mar-2023 15:56:33

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn',   @gui_OpeningFcn, ...
                  'gui_OutputFcn',    @gui_OutputFcn, ...
                  'gui_LayoutFcn',    [] , ...
                  'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before gui is made visible.
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to gui (see VARARGIN)

% Choose default command line output for gui

% UIWAIT makes gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

```

```
% Get default command line output from handles structure
%varargout{1} = handles.output;
```

5.6.2 DATASET BACKEND

```
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global train
matlabpath='C:\Users\Tejasvi
aryan\Downloads\malnutrition\malnutrition'
data = fullfile(matlabpath,'dataset')

trainimageDatastore(data,
'IncludeSubfolders',true,'LabelSource','foldernames');

count = train.countEachLabel;

msgbox('Dataset Loaded Successfully')

% Update handles structure
guidata(hObject, handles);
```

5.6.3 PRE – TRAINED MODEL BACKEND

```
% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global layers
disp('Pre-Trained Model Loaded...')
net = alexnet;

layers = [ imageInputLayer([236 281 1])

net(2:end-3)

fullyConnectedLayer(2)
```

```

softmaxLayer

classificationLayer()

]
msgbox('Pre-Trained Model Loaded Successfully')

% Update handles structure
guidata(hObject, handles);

```

5.6.4 TRAINING BACKEND

```

function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
%training
global train
matlabpath='C:\Users\Tejasvi
aryan\Downloads\malnutrition\malnutrition'
data = fullfile(matlabpath,'dataset')

train=imageDatastore(data,
'IncludeSubfolders',true,'LabelSource','foldernames');
train=augmentedImageDatastore([236
281],train,'ColorPreprocessing','rgb2gray');
global opt training train layers
opt=trainingOptions('sgdm', ...
'InitialLearnRate', 0.001, ...
'LearnRateSchedule', 'piecewise', ...
'LearnRateDropFactor', 0.1, ...
'LearnRateDropPeriod', 8, ...
'L2Regularization', 0.004, ...
'MaxEpochs', 10, ...
'MiniBatchSize', 20, ...
'Verbose', true, 'Plots','training-progress');
training = trainNetwork(train,trainlayers,opt);
msgbox('Trained Completed')

```

```
% Update handles structure  
guidata(hObject, handles);
```

5.7 Implementation Screenshots

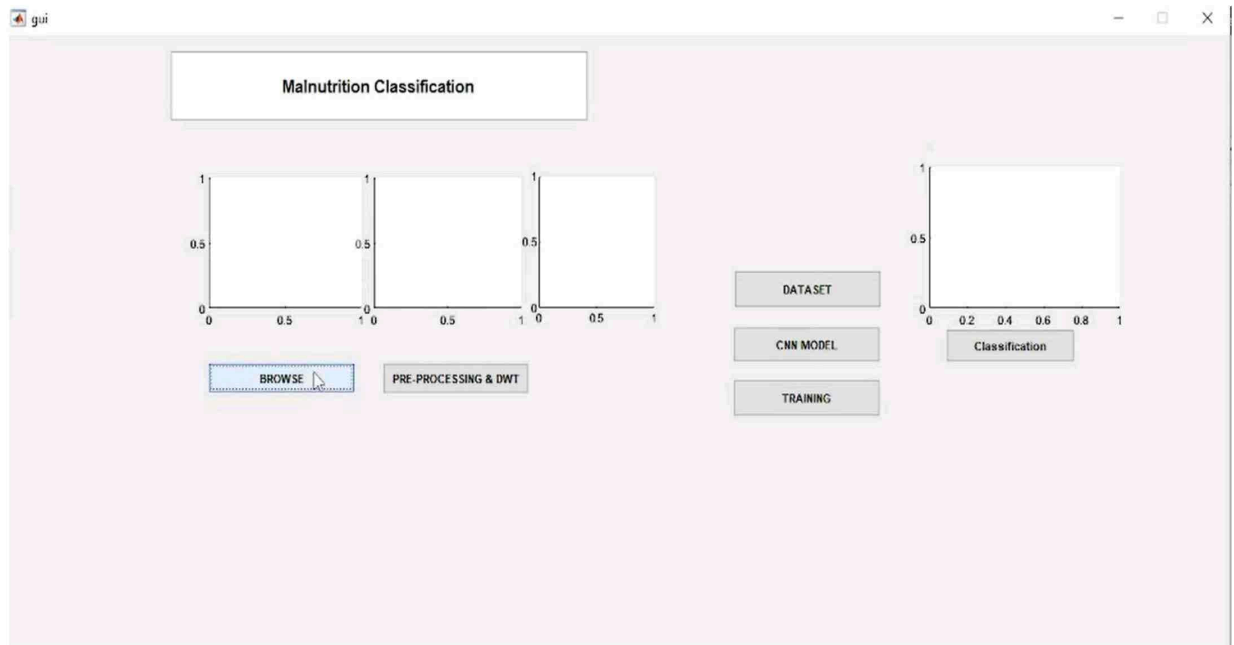


Fig 5.1 Start GUI

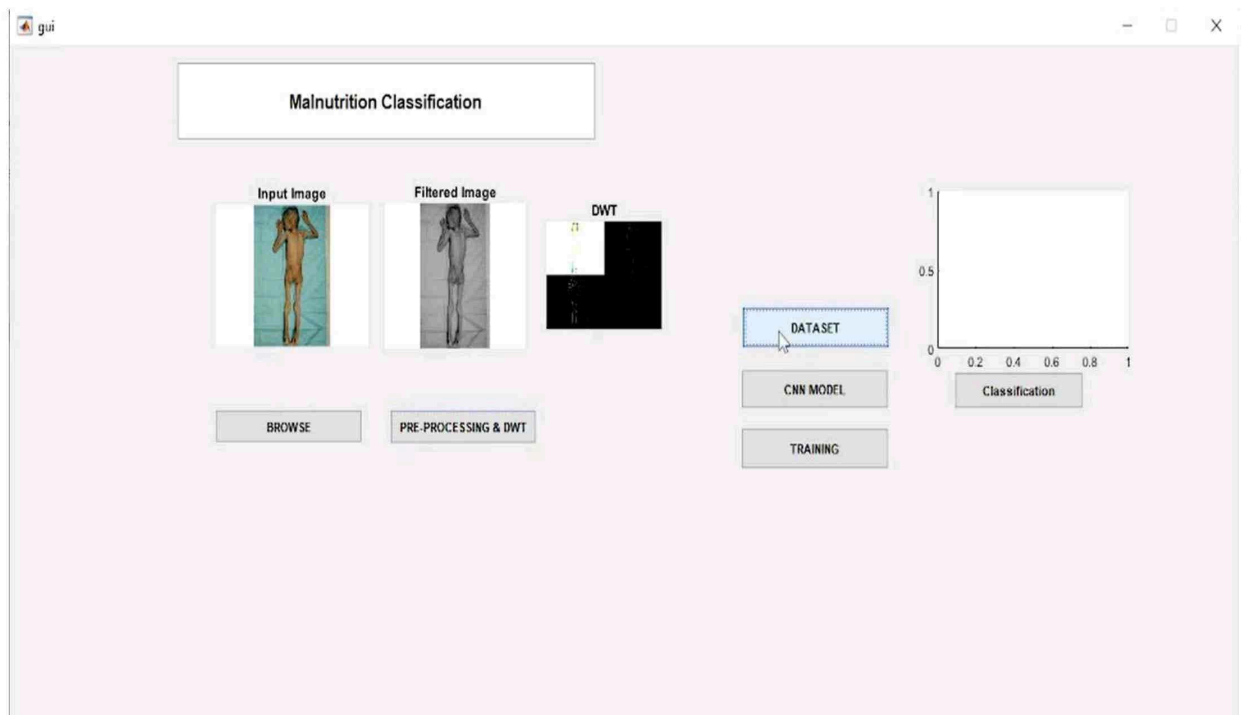


Fig 5.2 Input the test image

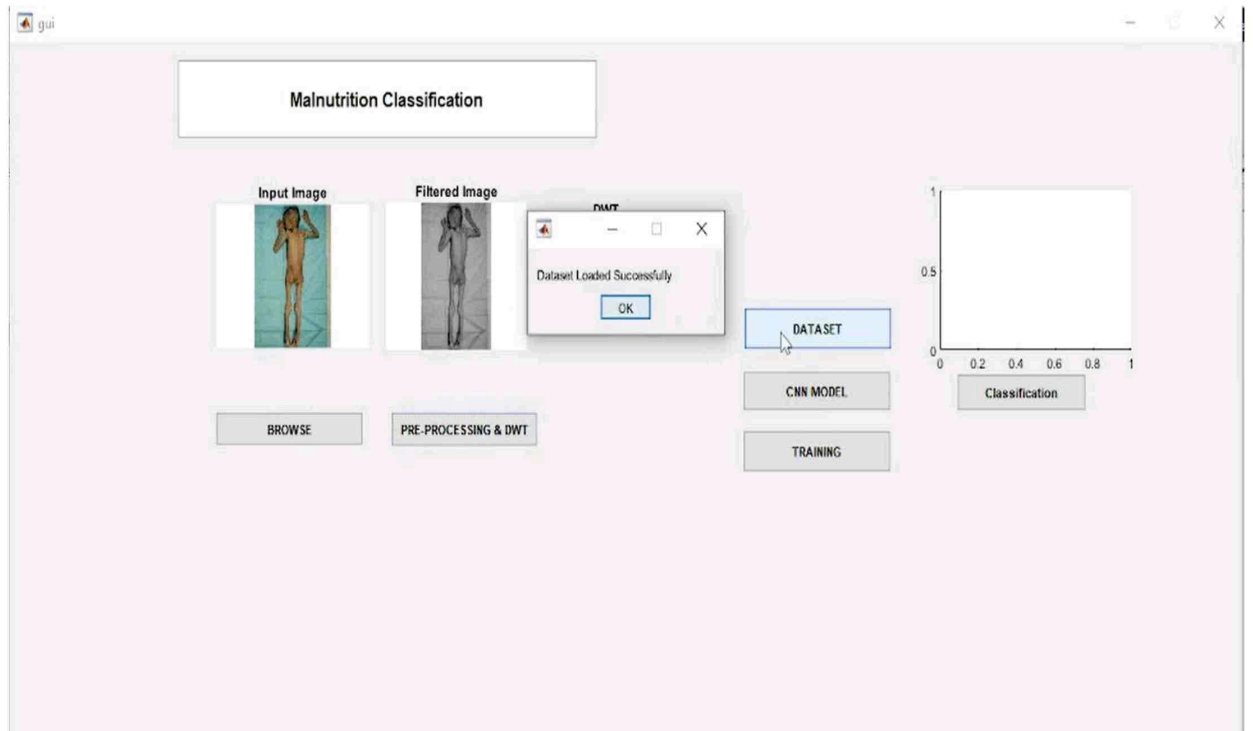


Fig 5.3 Dataset Loaded

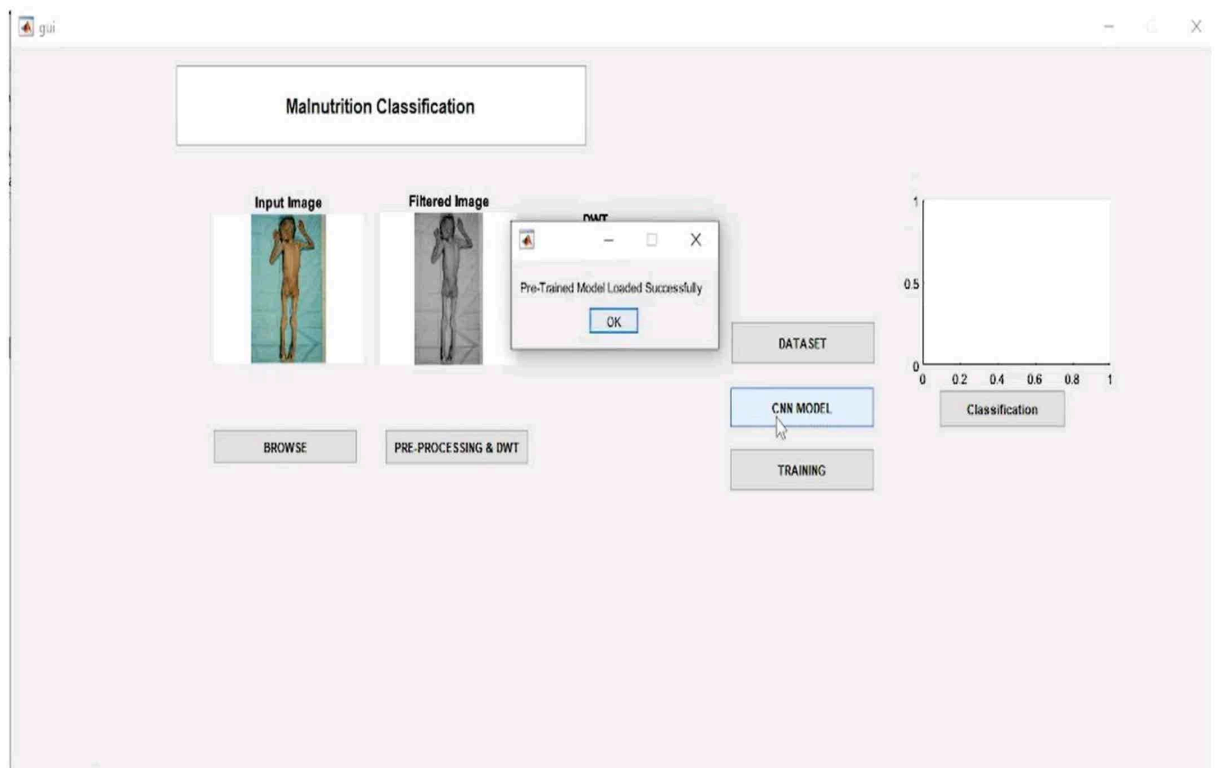


Fig 5.4 Pre-Trained Model Loaded

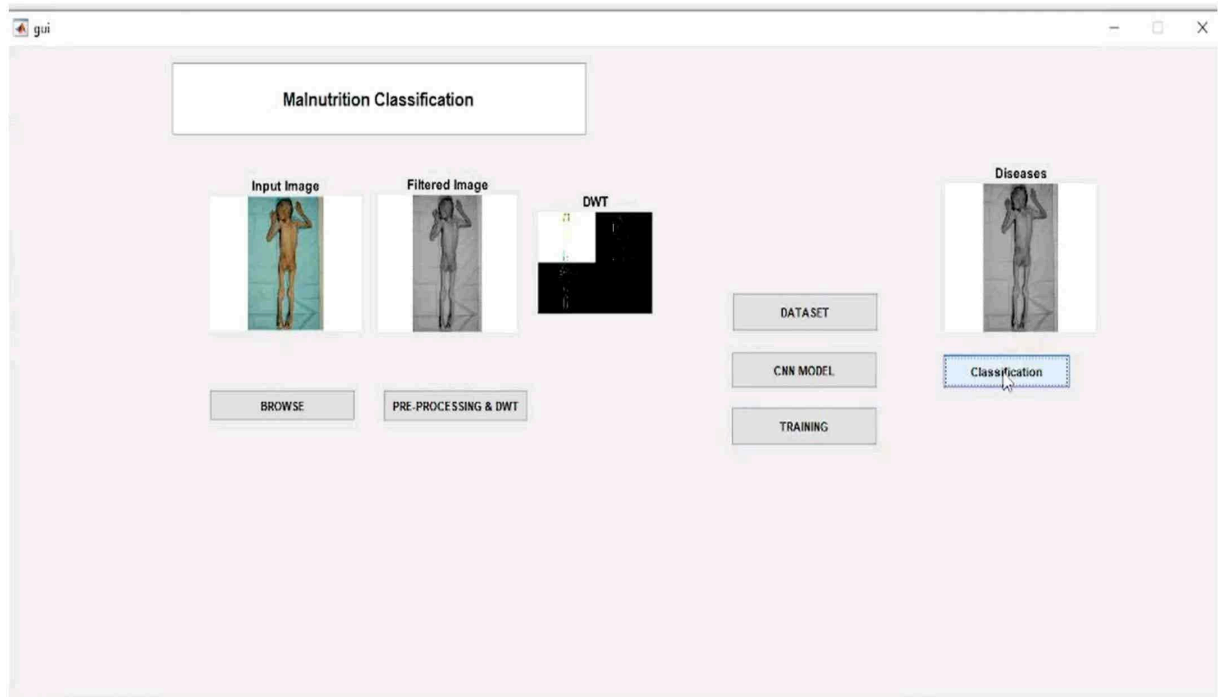


Fig 5.5 Classification of image

CHAPTER 6

RESULTS AND DISCUSSION

6.1 INTRODUCTION

The results and discussion section is an important part of a deep learning-based system for diagnosing malnutrition. It provides an analysis of the performance of the model and a discussion of the implications of the results.

Here are some key points to consider when discussing the results of the proposed system for diagnosing malnutrition:

6.2 ANALYSIS OF RESULT AND OUTPUT

1. Key results summary: The conclusion needs to list the main conclusions of the proposed system, together with its strengths and weaknesses.
2. Contribution to healthcare: The relevance of the suggested system and its possible influence on healthcare should be highlighted in the conclusion. Improved malnutrition diagnosis accuracy and effectiveness, cost and time savings over conventional techniques, and better patient outcomes may all result from this.
3. Implications for future work: The discussion of the suggested system's implications for research on deep learning and the diagnosis of malnutrition should also be included in the conclusion. This could entail enhancing the model's performance, looking into different deep learning architectures, or incorporating extra data sources to strengthen the model.
4. Concluding remarks: The conclusion ought to offer a statement that highlights the significance of the proposed system for the healthcare industry while summarising the main conclusions and contributions of the system.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

The main conclusions and contributions of the suggested system are outlined in the conclusion of a deep learning-based system for diagnosing malnutrition. The importance of the suggested system and its potential influence on healthcare are highlighted.

Here are some important ideas to bear in mind when drafting the conclusion of a suggested method for identifying malnutrition:

1. A list of the main findings The important conclusions of the proposed system, including its effectiveness and limits, should be outlined in the conclusion.
2. Contribution to healthcare: The relevance of the suggested system and its possible influence on healthcare should be highlighted in the conclusion. Improved malnutrition diagnosis accuracy and effectiveness, cost and time savings over conventional techniques, and better patient outcomes may all result from this.
3. Implications for future work: The discussion of the suggested system's implications for research on deep learning and the diagnosis of malnutrition should also be included in the conclusion. This could entail enhancing the model's performance, looking into different deep learning architectures, or incorporating extra data sources to strengthen the model.
4. Concluding remarks: The conclusion ought to offer a statement that highlights the significance of the proposed system for the healthcare industry while summarising the main conclusions and contributions of the system.

In conclusion, a deep learning-based system for diagnosing malnutrition has the potential to revolutionize the way that malnutrition is diagnosed and treated. By leveraging the power of deep learning, the proposed system can provide fast, accurate, and non-invasive malnutrition diagnosis, improving patient outcomes and reducing healthcare costs. The proposed system represents a major step forward in the field of deep learning and malnutrition diagnosis and has the potential to transform healthcare practices in this area.

7.2 FUTURE WORK

A deep learning-based system for diagnosing malnutrition may be improved in the future using a variety of techniques designed to boost the model's functionality and robustness.

These potential areas for future improvement are listed below:

- **Enhanced training data** Expanding the training data set to incorporate a larger variety of photos and demographic data is one potential area for improvement. This might aid the model's performance and allow it to more accurately generalise to fresh cases of malnutrition.
- **Real-time surveillance** The creation of a real-time monitoring system for malnutrition might enable medical professionals to keep track of patients' nutritional status changes over time and monitor them continually. Through early detection and intervention, malnutrition may be prevented before it worsens.
- These are only a few of the many potential areas where a deep learning-based system for identifying malnutrition could be improved in the future. The suggested approach has the potential to significantly alter how malnutrition is identified and treated, improve patient outcomes, and save healthcare costs by continuing to develop and enhance the model

REFERENCES

- [1] 'Levels and trends in child malnutrition', UNICEF/WHO/World Bank Group – Joint Child Malnutrition Estimates 2019 edition. <https://www.who.int/nutgrowthdb/jme-2019-key-findings.pdf>
- [2] The State of the World's Children 2019: Children, Food and Nutrition, UNICEF. <https://data.unicef.org/resources/state-of-theworlds-children-2019/>
- [3] Malnutrition is a world health crisis, [2019], WHO. <https://www.who.int/nutrition/topics/world-food-day-2019-malnutrition-world-health-crisis/en/>
- [4] Nutrition in India, National Family Health Survey (NFHS-3) India 2005-06. <https://dhsprogram.com/pubs/pdf/frind3/frind3- vol1andvol2.pdf>
- [5] India Fact Sheet, The National Family Health Survey (NFHS-4) 2015-16. <https://dhsprogram.com/pubs/pdf/FR339/FR339.pdf>
- [6] Kadam, N., Dabhade, V., Baravkar, R., Saravade, V., & Mankar, C. (2019). Detect Malnutrition in Underage Children by using TensorFlow Algorithm of Artificial Intelligence. (Vol. 06, Issue 12)
- [7] Mohseni, M., Aryankhesal, A., & Kalantari, N. (2019). Prevention of malnutrition among children under 5 years old in Iran: A policy analysis. PloS one, 14(3), e0213136.
- [8] Grellety, E., & Golden, M. H. (2018). Change in quality of malnutrition surveys between 1986 and 2015. Emerging themes in epidemiology, 15(1), 1-13.
- [9] Susan Smith, Tracy B. Chamblee, Karie Falder-Saeed, Kelly Haight, Melanie Horbal Shuster, Mary J. Waldo and Hea Lan Yoon, [January 2017], 'MALNUTRITION IN HOSPITALIZED ADULT PATIENTS', National Association of Clinical Nurse Specialists(NACNS). <https://nacns.org/wpcontent/uploads/2017/01/Malnutrition-Report.pdf>
- [10] Kellett, J., Kyle, G., Itsiopoulos, C., Naunton, M., & Luff, N. (2016). Malnutrition: the importance of identification, documentation, and coding in the acute care setting. Journal of nutrition and metabolism, 2016. (Vol. 2016, Article ID 9026098) <https://doi.org/10.1155/2016/9026098>

- [11] Nie, P., Rammohan, A., Gwozd, W., & Sousa-Poza, A. (2016). Developments in Undernutrition in Indian Children Under Five: A Decompositional Analysis. <http://ftp.iza.org/dp9893.pdf>
- [12] Madhuri Arya, Pooja Chavhan and Ujjwala Chaudhry, [2015], 'Malnutrition Detection and Management System', International Journal of Computer Science and Information Technology Research (Vol. 3, Issue 1, pp: (364-368))
- [13] Sahu, S. K., Kumar, S. G., Bhat, B. V., Premarajan, K. C., Sarkar, S., Roy, G., & Joseph, N. (2015). Malnutrition among under-five children in India and strategies for control. Journal of natural science, biology, and medicine, 6(1), 18.
- [14] Markos, Z., Doyore, F., Yifiru, M., & Haidar, J. (2014). Predicting Under nutrition status of under-five children using data mining techniques: The Case of 2011 Ethiopian Demographic and Health Survey. J Health Med Inform, 5, 152.
- [15] Markos, Z., Doyore, F., Yifiru, M., & Haidar, J. (2014). Predicting Under nutrition status of under-five children using data mining techniques: The Case of 2011 Ethiopian Demographic and Health Survey. J Health Med Inform, 5, 152.
- [16] Dezhi, X., & Ganegoda, G. U. (2011). Rule based classification to detect malnutrition in children. International Journal on Computer Science and Engineering (IJCSE)Vol,3

APPENDIX

A. SOURCE CODE

```
function varargout = gui(varargin)
% GUI MATLAB code for gui.fig
%   GUI, by itself, creates a new GUI or raises the existing
%   singleton*.
%
%   H = GUI returns the handle to a new GUI or the handle to
%   the existing singleton*.
%
%   GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in GUI.M with the given input arguments.
%
%   GUI('Property','Value',...) creates a new GUI or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before gui_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to gui_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
%   one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui

% Last Modified by GUIDE v2.5 21-Mar-2023 15:56:33

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @gui_OpeningFcn, ...
                  'gui_OutputFcn', @gui_OutputFcn, ...
```

```

        'gui_LayoutFcn', [] , ...
        'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before gui is made visible.
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui (see VARARGIN)

% Choose default command line output for gui

% UIWAIT makes gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
%varargout{1} = handles.output;

function pushbutton4_Callback(hObject, eventdata, handles)

```



```

% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global train
matlabpath ='C:\Users\jaiad\OneDrive\Desktop\malnutrition'
data = fullfile(matlabpath,'dataset')

train                                =                                imageDatastore(data,
'IncludeSubfolders',true,'LabelSource','foldernames');

count = train.countEachLabel;

msgbox('Dataset Loaded Successfully')

% Update handles structure
guidata(hObject, handles);

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global layers
disp('Pre-Trained Model Loaded...')
net = alexnet;

layers = [ imageInputLayer([236 281 1])

net(2:end-3)
fullyConnectedLayer(2)
softmaxLayer
classificationLayer()
]
msgbox('Pre-Trained Model Loaded Successfully')

% Update handles structure
guidata(hObject, handles);

```



```

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%training
global train
matlabpath ='C:\Users\jaiad\OneDrive\Desktop\malnutrition'
data = fullfile(matlabpath,'dataset')
train = imageDatastore(data,
'IncludeSubfolders',true,'LabelSource','foldernames');
train = augmentedImageDatastore([236 281],train,'ColorPreprocessing',
'rgb2gray');
global opt training train layers
opt=trainingOptions('sgdm', ...
    'InitialLearnRate', 0.001, ...
    'LearnRateSchedule', 'piecewise', ...
    'LearnRateDropFactor', 0.1, ...
    'LearnRateDropPeriod', 8, ...
    'L2Regularization', 0.004, ...
    'MaxEpochs', 10, ...
    'MiniBatchSize', 20, ...
    'Verbose', true, 'Plots','training-progress');

training = trainNetwork(train,layers,opt);
msgbox('Trained Completed')
% Update handles structure
guidata(hObject, handles);
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%       str2double(get(hObject,'String')) returns contents of edit1 as a double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global inp
global img
cd input
[file path] = uigetfile('.bmp;.png;*.jpg','Pick an Image File');

if isequal(file,0)
    warndlg('File not selected');
else

inp = imread(file);

cd ..
axes(handles.axes1);
imshow(inp);
title('Input Image');
img=inp;
if size(inp,3)>1
    Freg = im2gray(inp);
end
handles.img=img;
end
% Update handles structure

```

```

guidata(hObject, handles);

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global inp J
img = im2gray(inp);

axes(handles.axes2);
inp=imresize(img,[236 281])

imshow(inp);
title('Filtered Image');
% Update handles structure
guidata(hObject, handles);
% --- Executes on button press in pushbutton4.

img = handles.img;

wavename = 'haar';
[cA,cH,cV,cD] = dwt2(im2double(img),wavename);
[cAA,cAH,cAV,cAD] = dwt2(cA,wavename); % Recompute Wavelet of
Approximation Coefs.
%Level2=[cAA,cAH; cAV,cAD]; %contacinat
%

axes(handles.axes4);
imshow([cAA,cAH;cAV,cAD],'Colormap',gray);
title('DWT');

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

global training inp out
out = classify(training,inp);
axes(handles.axes3);
imshow(inp);
title(string(out));
% Update handles structure
guidata(hObject, handles);

function varargout = gui(varargin)
% GUI MATLAB code for gui.fig
%   GUI, by itself, creates a new GUI or raises the existing
%   singleton*.
%
%   H = GUI returns the handle to a new GUI or the handle to
%   the existing singleton*.
%
%   GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in GUI.M with the given input arguments.
%
%   GUI('Property','Value',...) creates a new GUI or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before gui_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to gui_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui

% Last Modified by GUIDE v2.5 21-Mar-2023 15:56:33

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;

```

```

gui_State = struct('gui_Name',      mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @gui_OpeningFcn, ...
                  'gui_OutputFcn', @gui_OutputFcn, ...
                  'gui_LayoutFcn', [] , ...
                  'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui is made visible.
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui (see VARARGIN)


% Choose default command line output for gui


% UIWAIT makes gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB

```

```

% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
%varargout{1} = handles.output;

function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global train
matlabpath ='C:\Users\jaiad\OneDrive\Desktop\malnutrition'
data = fullfile(matlabpath,'dataset')

train                                =                                imageDatastore(data,
'IncludeSubfolders',true,'LabelSource','foldernames');

count = train.countEachLabel;

msgbox('Dataset Loaded Successfully')

% Update handles structure
guidata(hObject, handles);

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global layers
disp('Pre-Trained Model Loaded...')
net = alexnet;

layers = [ imageInputLayer([236 281 1])

net(2:end-3)

fullyConnectedLayer(2)

```

```

softmaxLayer

classificationLayer()

]
msgbox('Pre-Trained Model Loaded Successfully')

% Update handles structure
guidata(hObject, handles);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%training
global train
matlabpath ='C:\Users\jaiad\OneDrive\Desktop\malnutrition'
data = fullfile(matlabpath,'dataset')

train = imageDatastore(data,
'IncludeSubfolders',true,'LabelSource','foldernames');
train = augmentedImageDatastore([236 281],train,'ColorPreprocessing',
'rgb2gray');

global opt training train layers
opt=trainingOptions('sgdm', ...
    'InitialLearnRate', 0.001, ...
    'LearnRateSchedule', 'piecewise', ...
    'LearnRateDropFactor', 0.1, ...
    'LearnRateDropPeriod', 8, ...
    'L2Regularization', 0.004, ...
    'MaxEpochs', 10, ...
    'MiniBatchSize', 20, ...
    'Verbose', true, 'Plots','training-progress');

training = trainNetwork(train,layers,opt);

```

```

msgbox('Trained Completed')

% Update handles structure
guidata(hObject, handles);

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global inp
global img
cd input
[file path] = uigetfile('*.bmp;*.png;*.jpg','Pick an Image File');

if isequal(file,0)

```



```

        warndlg('File not selected');
else

inp = imread(file);

cd ..
axes(handles.axes1);
imshow(inp);
title('Input Image');
img=inp;
if size(inp,3)>1
    Freg = im2gray(inp);
end
handles.img=img;
end

% Update handles structure
guidata(hObject, handles);

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global inp J
img = im2gray(inp);

axes(handles.axes2);
inp=imresize(img,[236 281])

imshow(inp);
title('Filtered Image');
% Update handles structure
guidata(hObject, handles);
% --- Executes on button press in pushbutton4.
img = handles.img;
wavename = 'haar';

```

```

[cA,cH,cV,cD] = dwt2(im2double(img),wavename);
[cAA,cAH,cAV,cAD] = dwt2(cA,wavename); % Recompute Wavelet of
Approximation Coefs.
%Level2=[cAA,cAH; cAV,cAD]; %concatinat
%
axes(handles.axes4);
imshow([cAA,cAH;cAV,cAD],'Colormap',gray);
title('DWT');
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global training inp out
out = classify(training,inp);
axes(handles.axes3);
imshow(inp);
title(string(out));
% Update handles structure
guidata(hObject, handles);

```

B. SCREENSHOTS

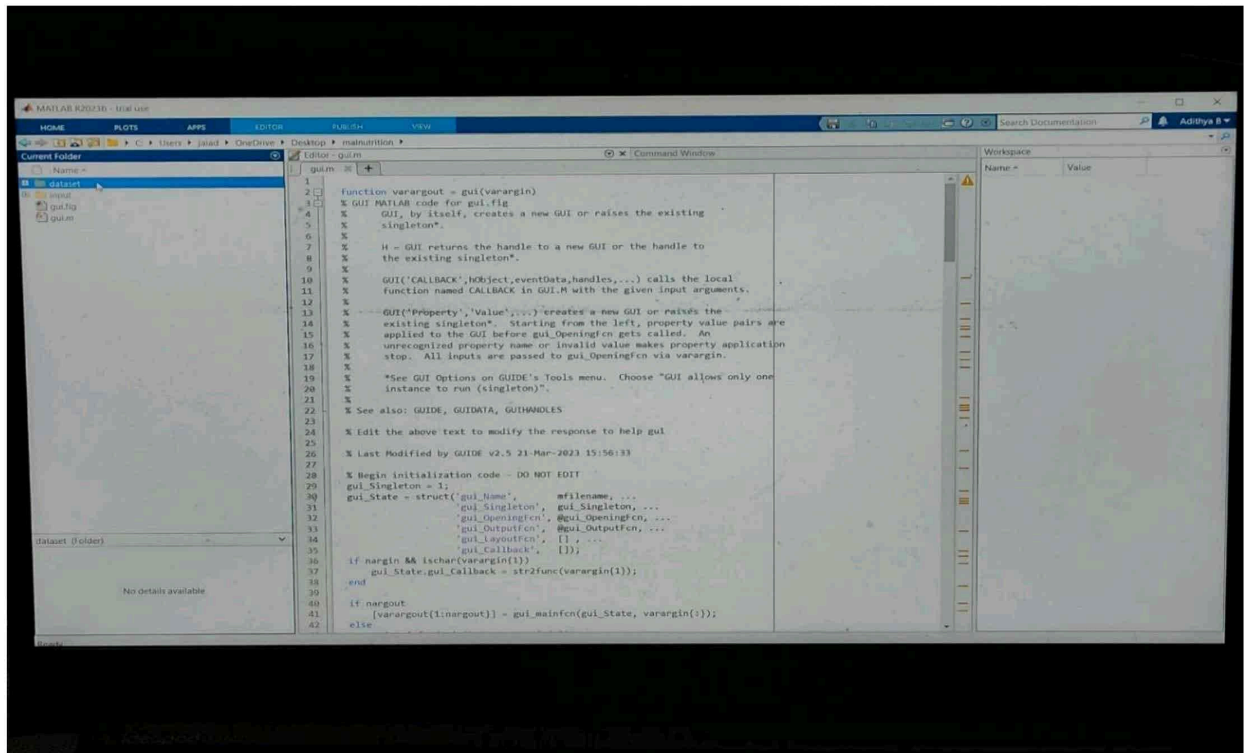


Fig. B.1: Importing Required Libraries

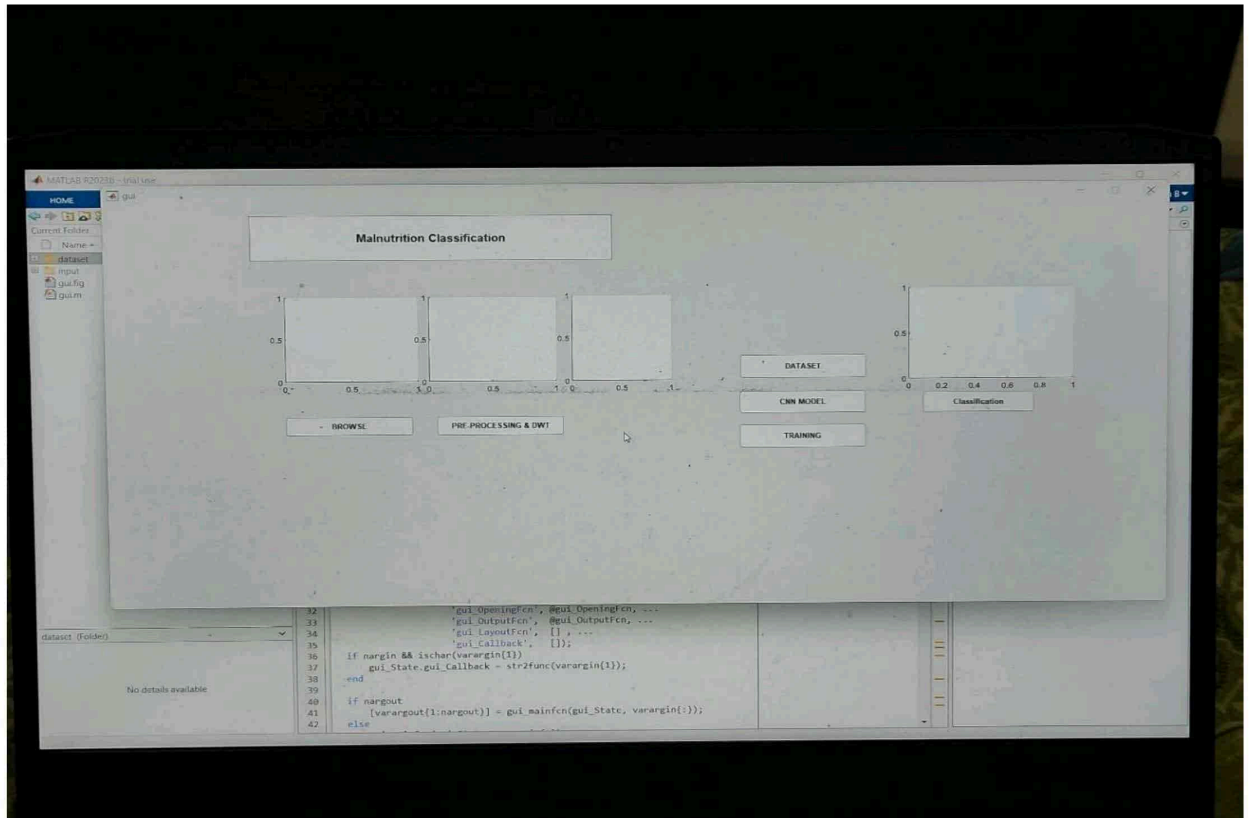


Fig. B.2: Pre-processing the data

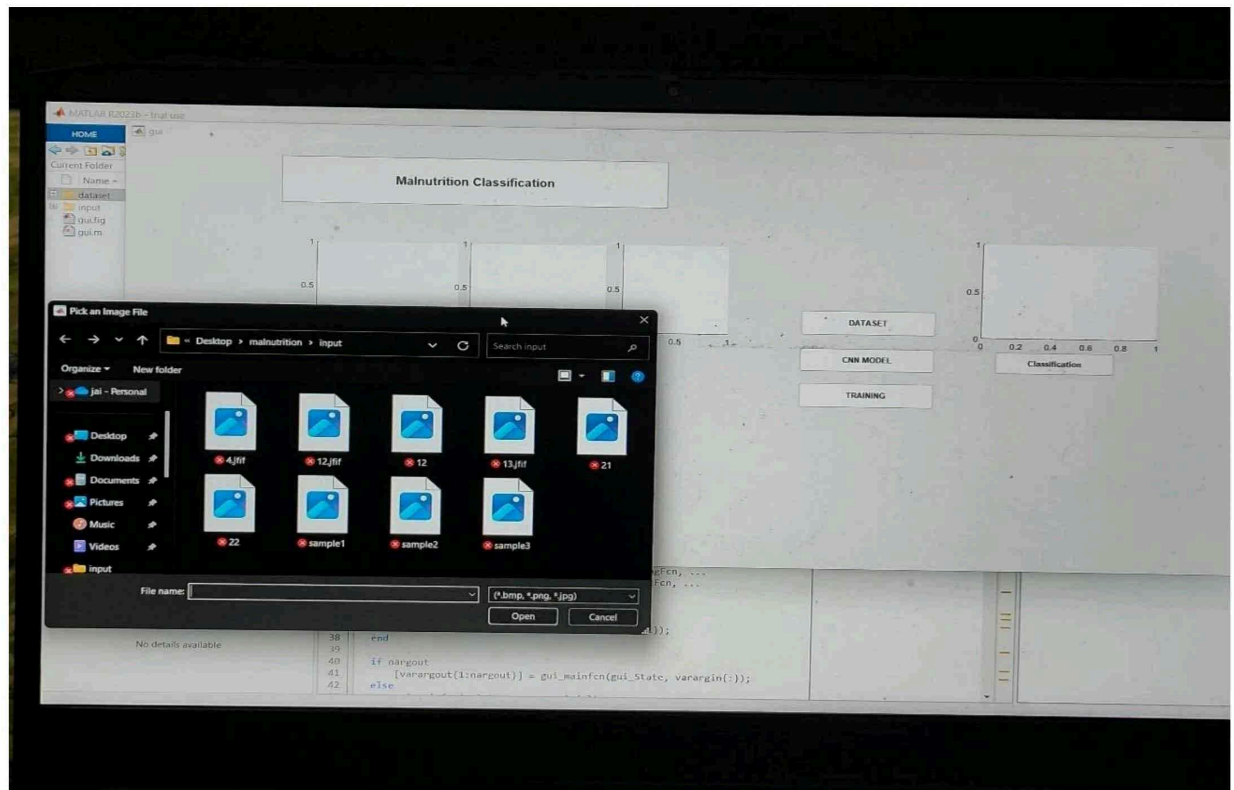


Fig. B.3: Extracting the Frames

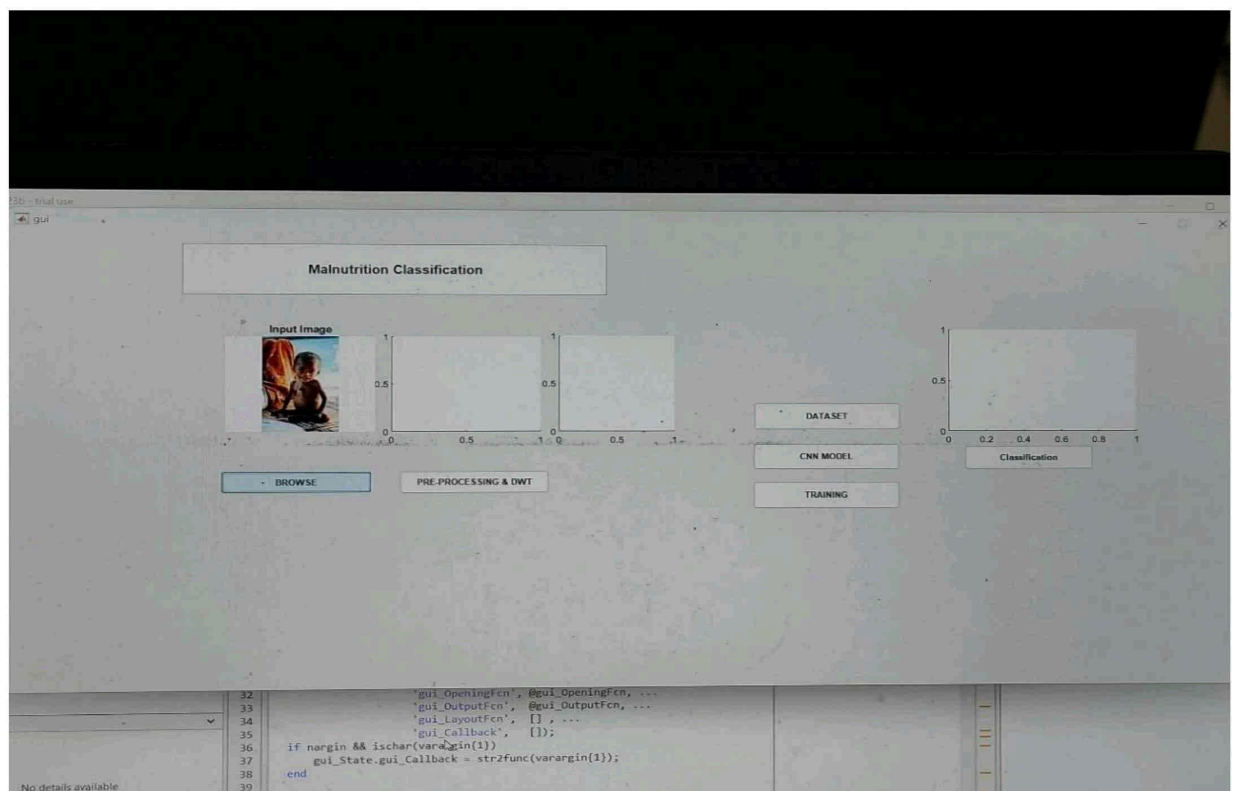


Fig. B.4: Creating dataset with required frames

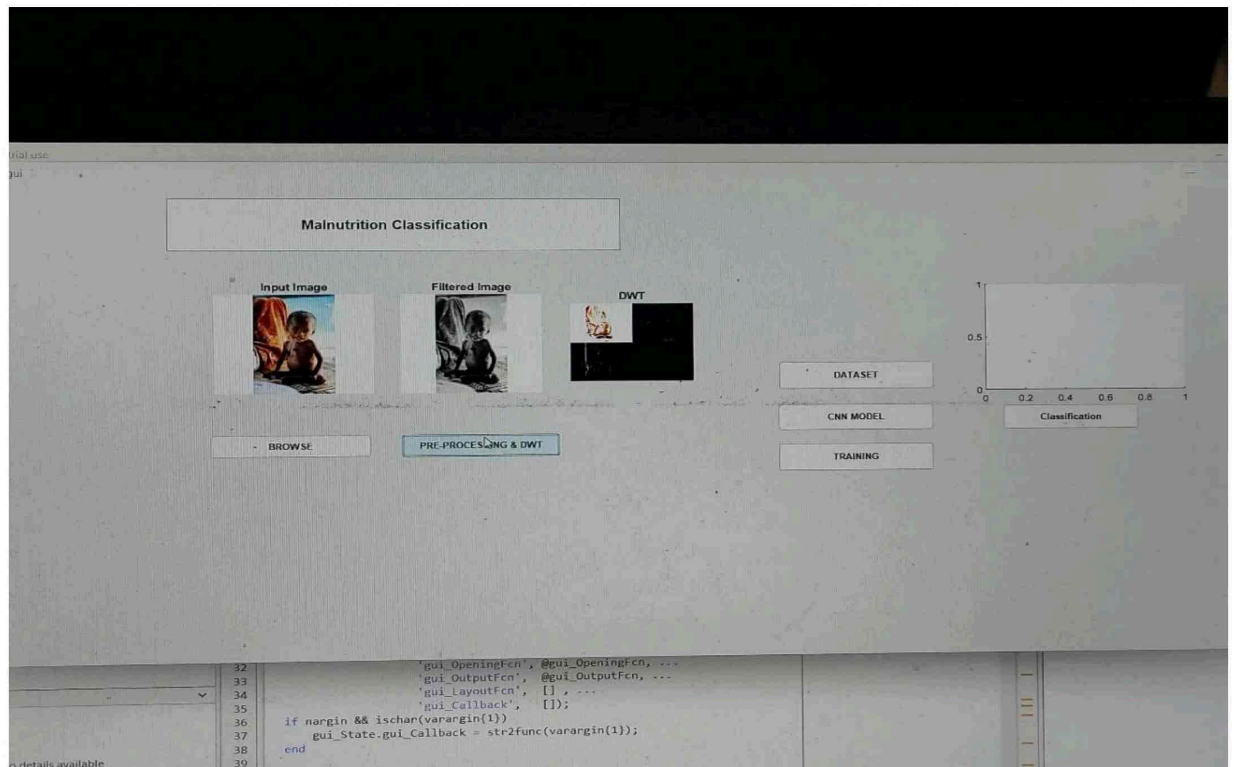


Fig. B.5: Creating ConvLSTM Model

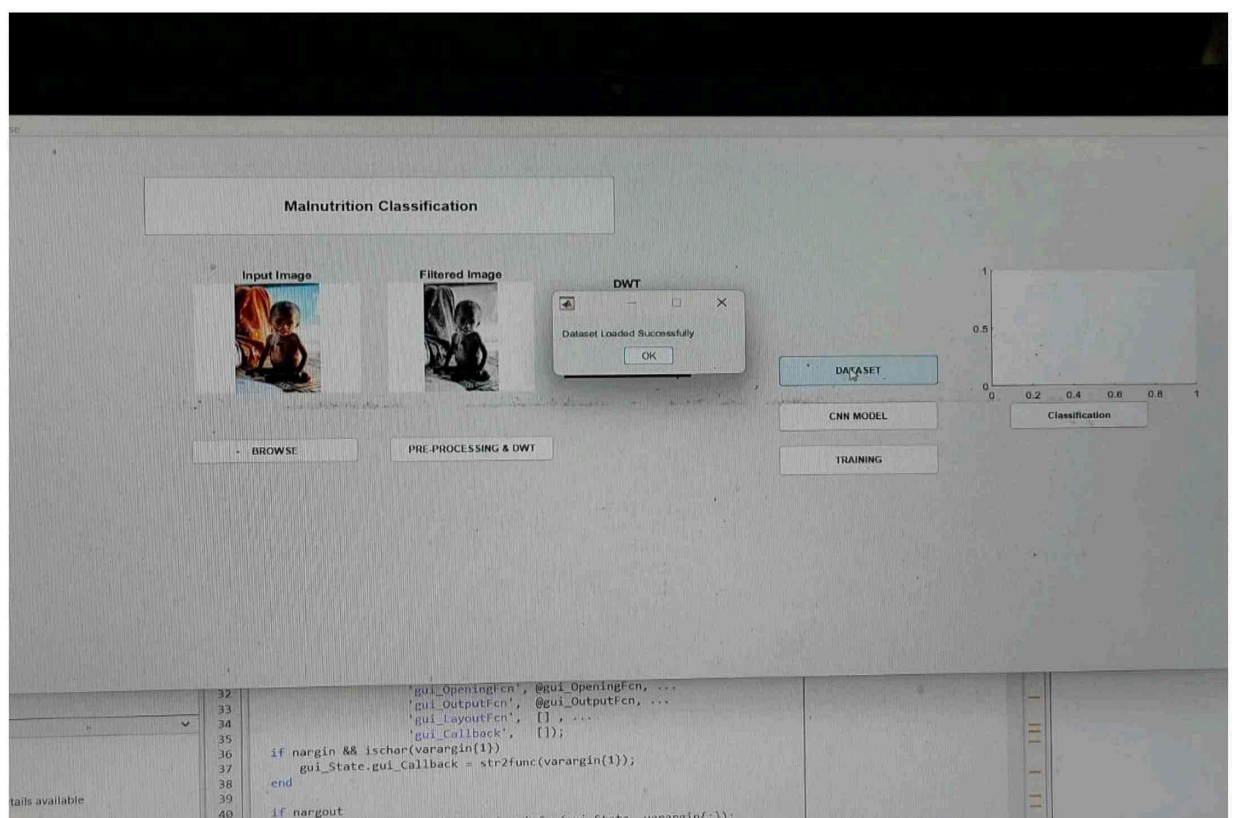


Fig. B.6: Successfully created Conv LSTM Model

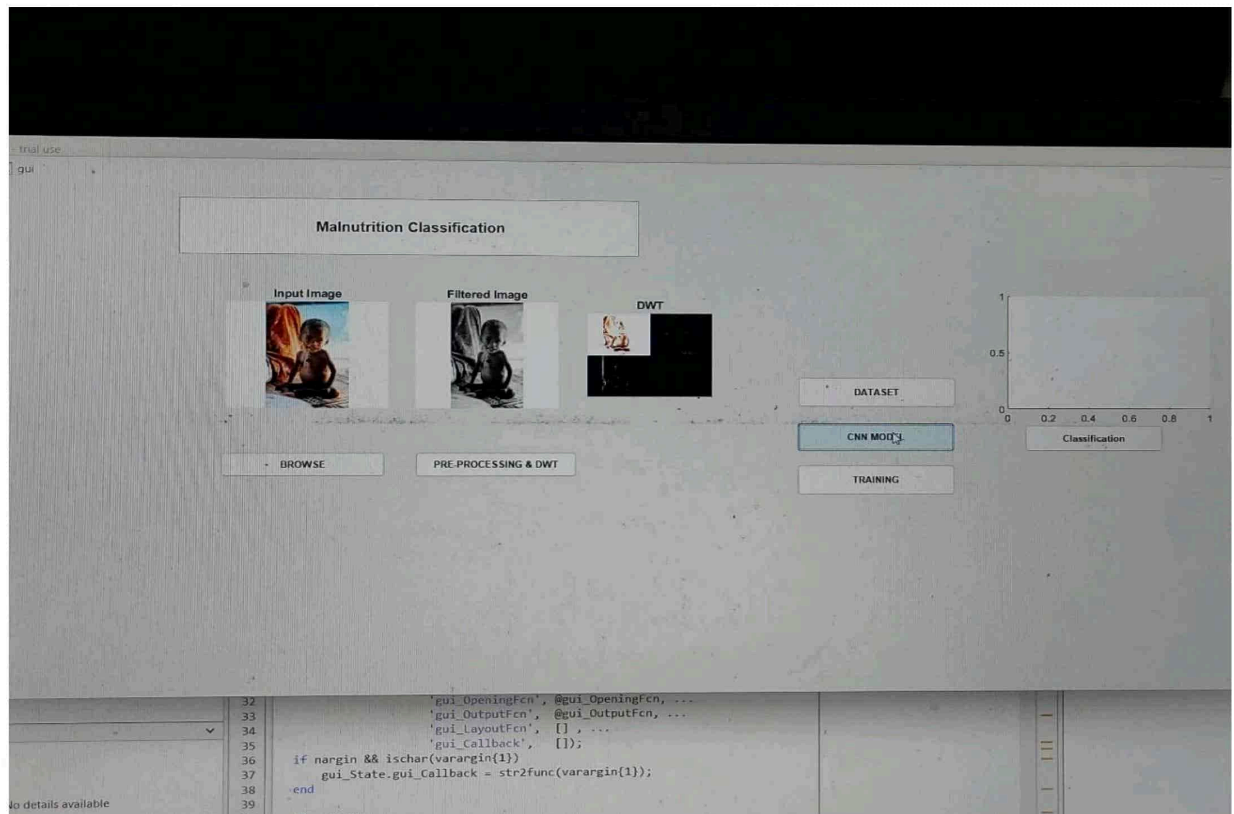


Fig. B.7: Plotting the Model

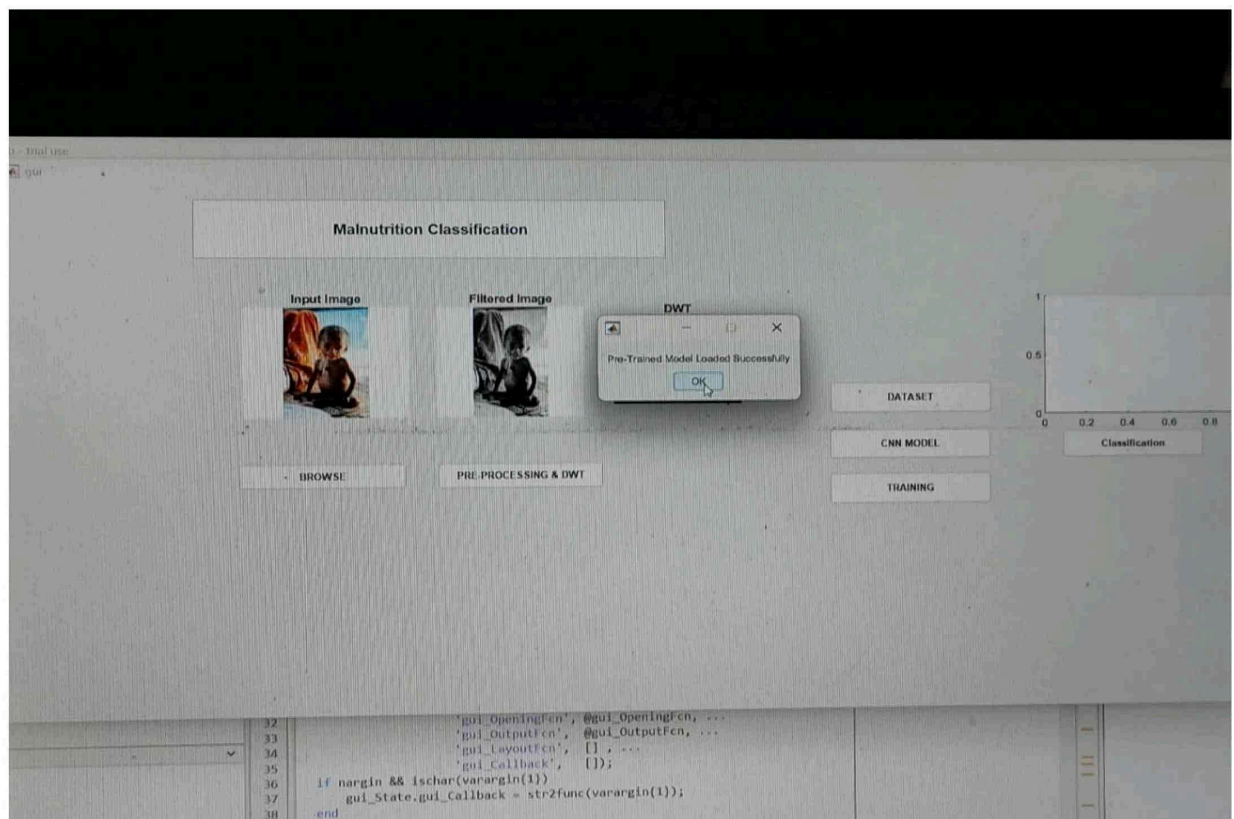


Fig. B.8: Evaluating the Model

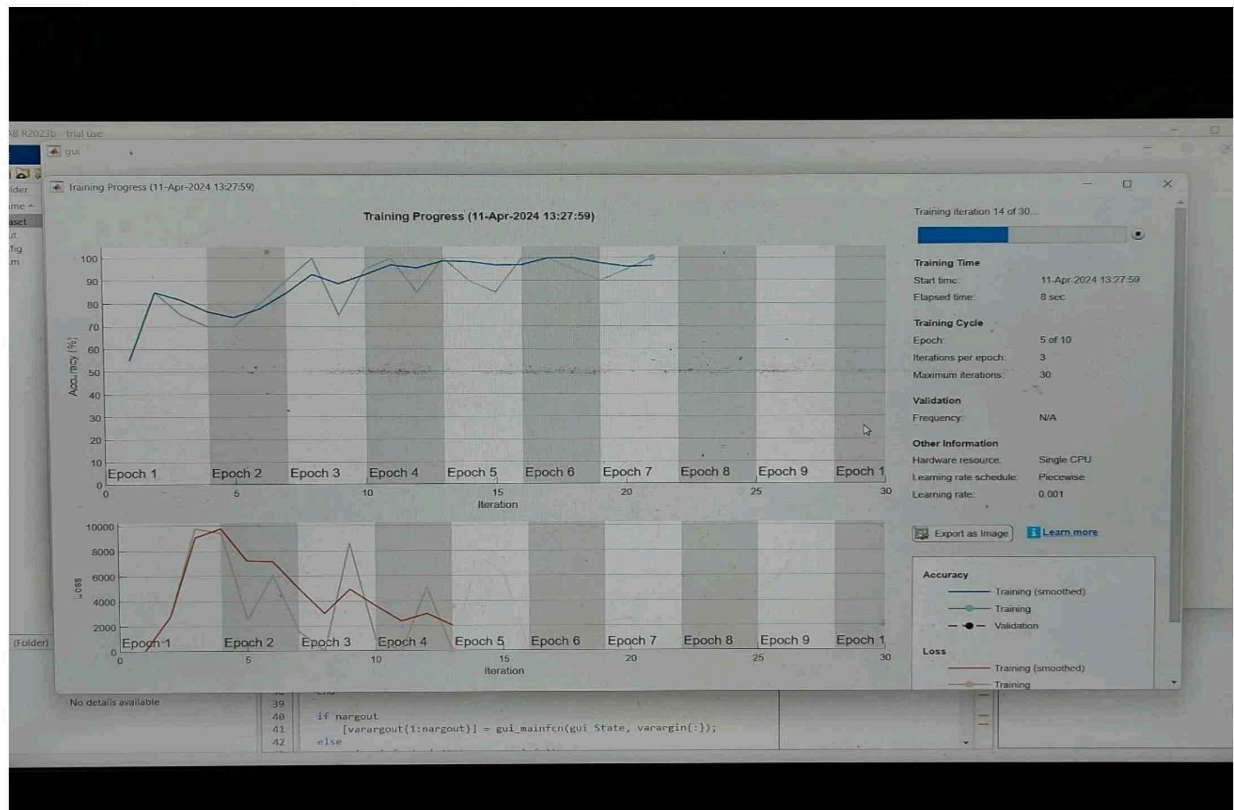


Fig. B.9: Plotting Loss and Accuracy Curves of ConvLSTM Model

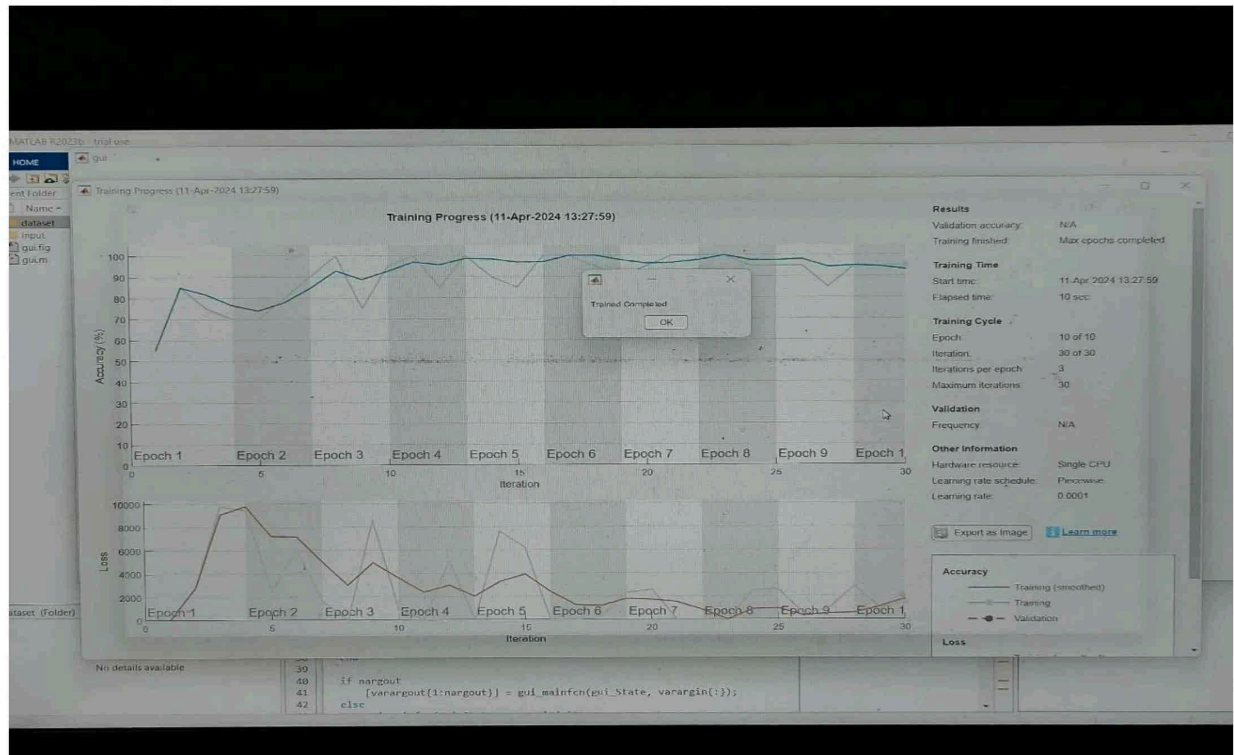


Fig. B.10: Creating LRCN Model

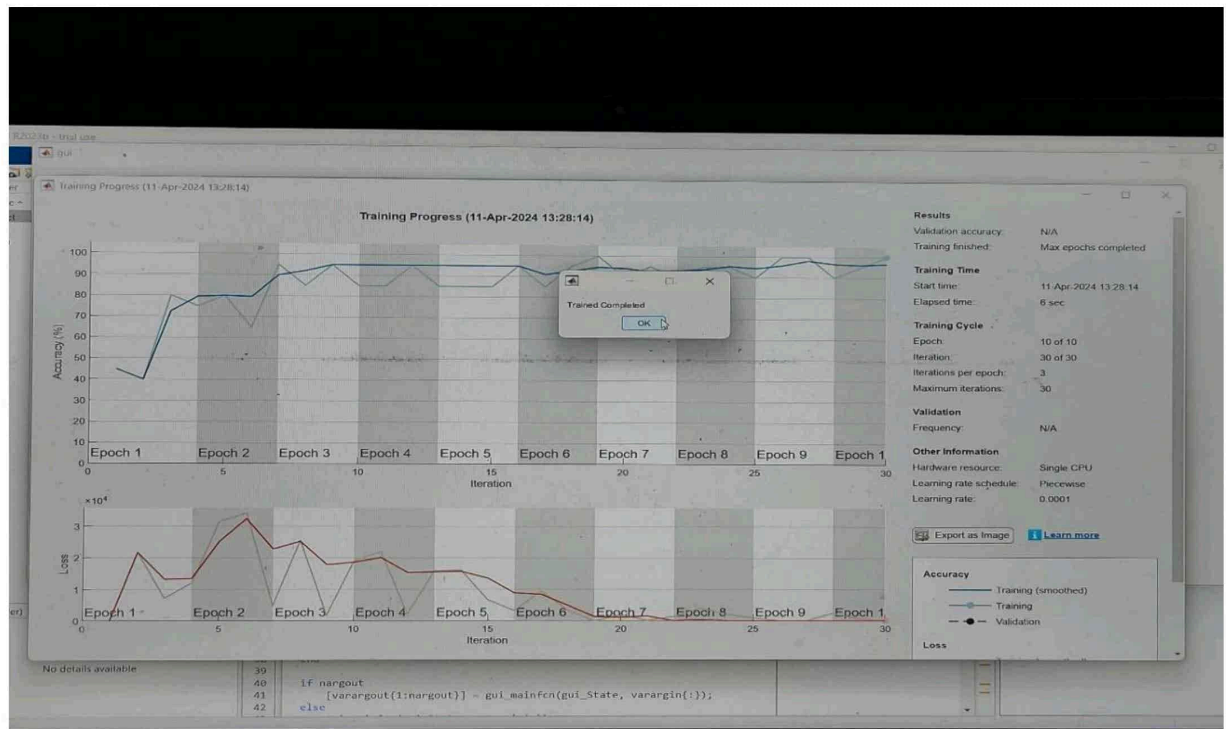


Fig. B.11: LRCN Model Created Successfully

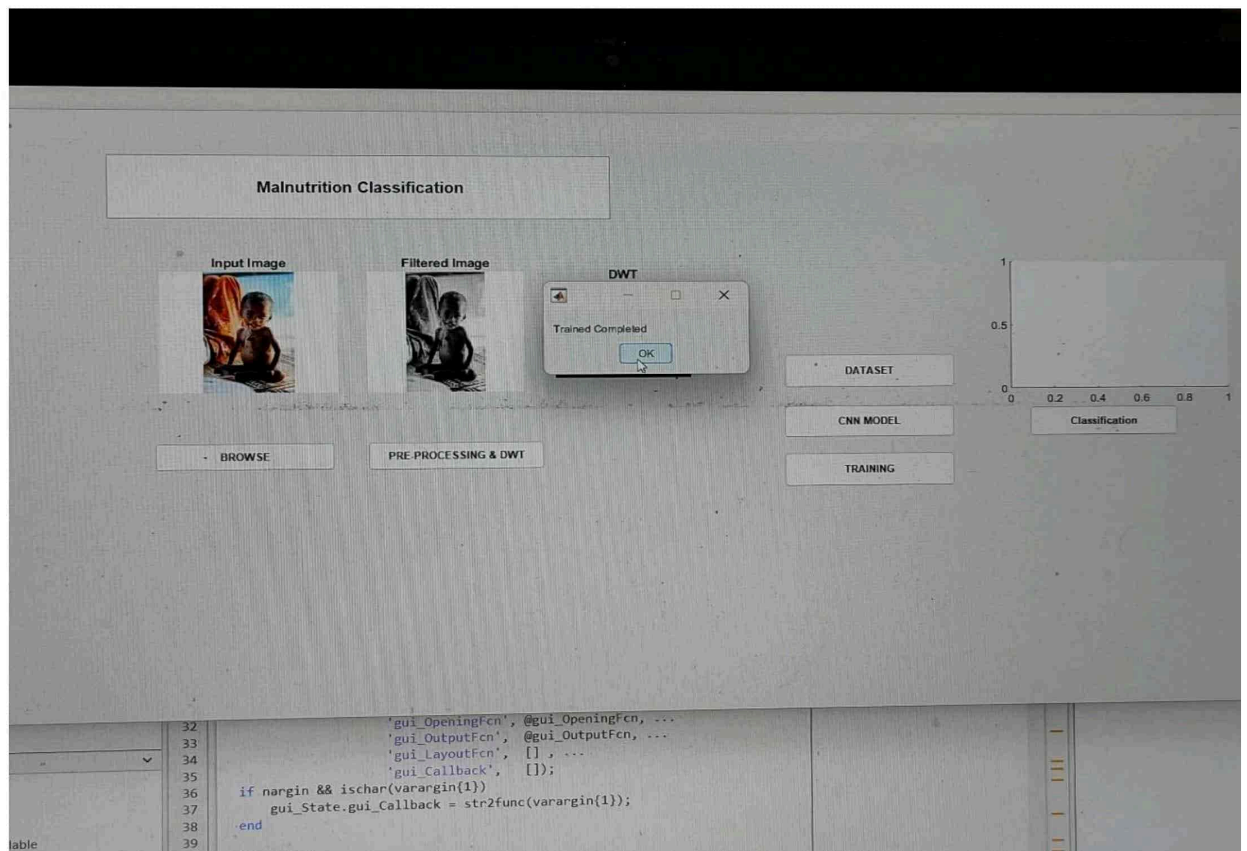


Fig. B.12: Plotting the Model

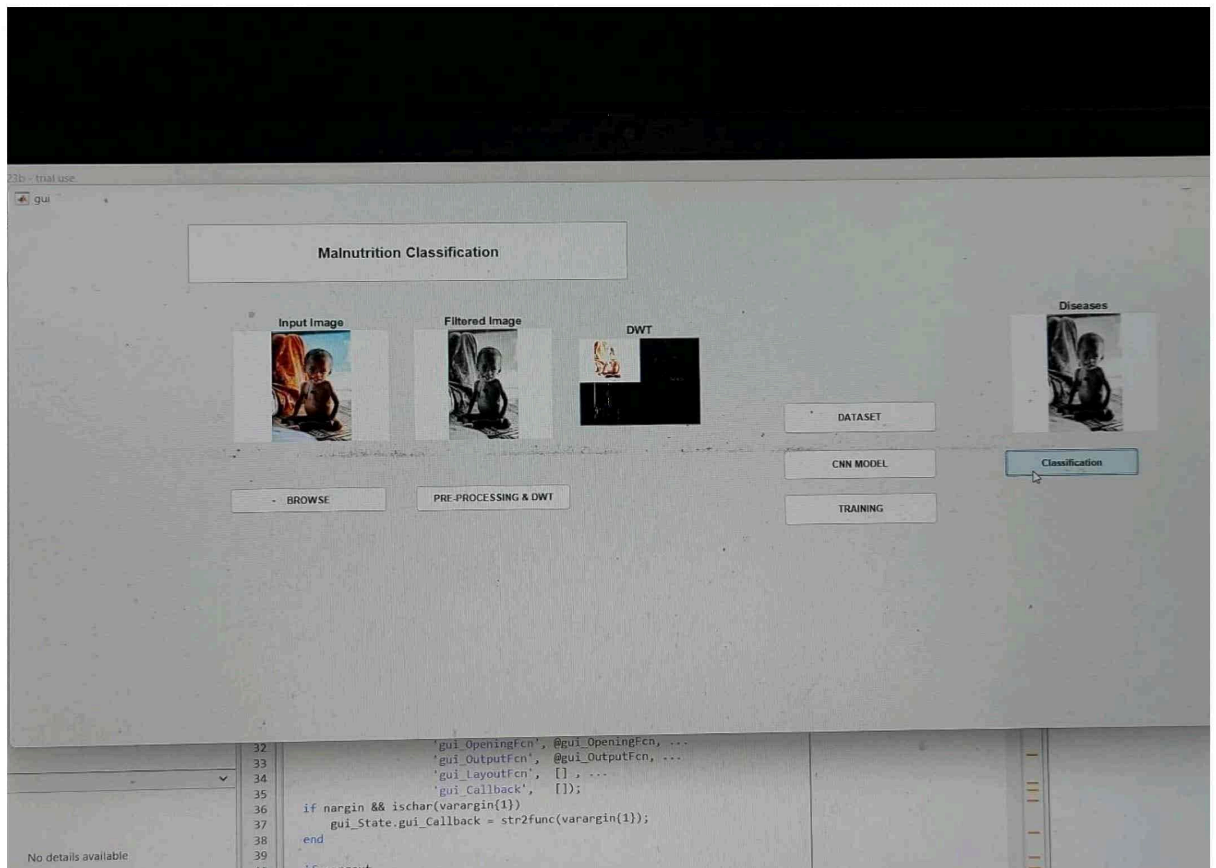


Fig.B.13:Plotting.the-Model