

DETECTING AND CLASSIFYING FETAL BRAIN ABNORMALITIES USING DEEP LEARNING

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Technology degree in BIOMEDICAL ENGINEERING

By

MAHALAKSHMI.V (39240024)

RIDDHITA THAKUR (39240032)



**INTERDISCIPLINARY PROJECT REPORT
DEPARTMENT OF BIOMEDICAL ENGINEERING
SCHOOL OF BIO AND CHEMICAL ENGINEERING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF BIOMEDICAL ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Interdisciplinary Project Report is the bonafide work of **Mahalakshmi.v (39240024)** , **Riddhita Thakur (39240032)** who carried out the Inter disciplinary project entitled **"DETECTING AND CLASSIFYING FETAL BRAIN ABNORMALITIES USING DEEP LEARNING"** under our supervision from **December 2021** to **April 2022**.

Guide

(Dr. J. PREM KUMAR)

Head of the Department

(Dr. T. SUDHAKAR)

Submitted for Viva voce Examination held on 30.04.2022

Internal Examiner

External Examiner

DECLARATION

We **MAHALAKSHMI.V ,RIDDHITA THAKUR** hereby declare that the Interdisciplinary Project Report entitled “**DETECTING AND CLASSIFYING FETAL BRAIN ABNORMALITIES USING DEEP LEARNING**” done by us under the guidance of **Dr. J. PremKumar** at Department of Biomedical Engineering, Sathyabama Institute of Science and Technology is submitted in partial fulfilment of the requirements for the award of **Bachelor of Technology** degree in **Biomedical Engineering**.

DATE : 28/4/2022
PLACE : CHENNAI

1. 
2. 

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this Interdisciplinary Project and for completing it successfully. We are grateful to them.

We would like to express our sincere and deep sense of gratitude to our Project Guide Dr. **J. Premkumar** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We convey our thanks to Dr. **T. Sudhakar**, Head of the Department, Department of Biomedical Engineering for providing us necessary support and details at the right time during the progressive reviews.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of Biomedical Engineering who were helpful in many ways for the completion of the project.

	TABLE OF CONTENTS	
CHAPTER NO	TITLE	PAGE
	ABSTRACT	vi
	LIST OF ABBREVIATION	ix
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
3	AIM AND SCOPE	5
	3.1 AIM	5
	3.2 SCOPE	5
	3.3 DOMAIN OVERVIEW	6
4	MATERIALS AND METHODS	7
	4.1 METHODOLOGY	7
	4.1.1 BLOCK DIAGRAM WITH EXPLANATION	7
	4.2 DIGITAL IMAGE PROCESSING	8
	4.2.1 CLASSIFICATION OF IMAGES	8
	4.3 SYSTEM REQUIREMENTS	12
	4.4 SOFTWARE USED	12
	4.4.1 MATLAB	12
	4.4.2 THE MATLAB SYSTEM	14
	4.4.3 STARTING & QUITTING MATLAB	14
	4.4.4 MATLAB DESKTOP	15

	4.4.5 MANIPULATING MATRICES	18
	4.4.6 GUI	20
	4.5 MODULES DESCRIPTION	24
	4.5.1 PRE-PROCESSING	24
	4.5.2 DT-CWT	25
	4.5.3 GLCM	28
	4.5.4 NEURAL NETWORK	30
	4.5.5 ARCHITECTURE OF NN	31
5	RESULTS AND DISCUSSION	46
	5.1 RESULTS	46
	5.2 DISCUSSION	47
6	SUMMARY AND CONCLUSIONS	48
	6.1 SUMMARY	48
	6.2 CONCLUSIONS	49
	REFERENCE	51
	APPENDIXES	60

LIST OF ABBREVIATION

ABBREVIATION	EXPANSION
MRI	Magnetic resonance Imaging
ROI	Region of Interest
CNN	Convolutional Neural Network
SVM	Supervised Machine Learning
CT	Computed Tomography
WHO	World Health Organization
GUI	Graphical User Interface
GLCM	Gray Level Co-Occurrence Matrix
NN	Neural Networks
GRNN	General Regression Neural Network
FCM	Firebase Cloud Messaging
UML	Unified Modelling Language
OMT	Object Modelling Technique
OOSE	Object Oriented Software Engineering
DTL	Data Transformation Language
DWT	Digital Wavelet Transform

ABSTRACT:

In today's world, not only adults, children, teenage suffer from different diseases but also the yet to be born babies suffer from various abnormalities. We hear many cases that the child was born with some disability and because of the delay in the discovery and operation of the disability, the disability becomes permanent. Out of one thousand, three women are pregnant with abnormal child. If somehow, we detect the abnormality in the child when it is in the fetal stage and operate and medicate according to that then the abnormality can be treated very effectively and efficiently. Our paper deals with the same concept of detecting and classifying the brain abnormalities in the fetus using various deep learning techniques and algorithms. There have been previous works on the same but the technique used by others included machine learning and it had some drawbacks which can be resolved using deep learning techniques. Deep learning has greater efficiency and advantage over machine learning. In our method of detection, we take the help of MRI technique to first capture the brain image of the fetus. Then we perform various pre processing steps to extract the ROI . Then we use feature extraction and reduction techniques to obtain more developed and detailed image of the fetus. We compare the image with the normal fetal brain images to classify and detect abnormalities. We make the use of CNN classifier algorithm of the deep learning technique to achieve high level of accuracy. CNN algorithm is better than k means clustering and SVM classifier algorithm of machine learning techniques. Our work has shown higher accuracy than previous models and our future work will involve increasing of classification and data. Detecting and classifying fetal brain abnormalities from magnetic resonance imaging (MRI) is important, as approximately 3 in 1000 women are pregnant with a fetal of abnormal brain. Early detection of fetal brain abnormalities using machine learning techniques can improve the quality of diagnosis and treatment planning. The literature has shown that most of the work made to classify brain abnormalities in very early age is for preterm infants and neonates not fetal. However, research papers that studied fetal brain MRI images have mapped these images with the neonates MRI images to classify an abnormal behavior in newborns not fetal. In this paper, a pipeline process is proposed for fetal brain classification (FBC) which uses machine learning techniques. The main contribution of this paper is the classification of fetal brain abnormalities in early stage, before the fetal is born. The proposed algorithm is capable of detecting and classifying a variety of

abnormalities from MRI images with a wide range of fetal gestational age (GA) (from 16 to 39 weeks) using a flexible and simple method with low computational cost. The novel proposed method consists of four phases; segmentation, enhancement, feature extraction and classification. The results have shown that the proposed method has an area under ROC curve (AUC) of 84%, 86%, 80% and 84.5% for, Linear discriminate analysis (LDA), support vector machine (SVM), K-nearest neighbor (KNN), and Ensemble Subspace Discriminates classifiers respectively. This shows that our proposed has successfully classified fetal brain abnormalities with images of different fetal GA. The results are promising. Future work will be done to improve classification results and increase the dataset.

KEYWORDS:

- MR Image,
- CNN Classifier,
- Fetal brain,
- DWT,
- Preprocessing,
- Segmentation.

CHAPTER- 1

INTRODUCTION

Magnetic resonance imaging (MRI) is an important tool for medical investigation of the brain and is considerably enhanced in uterus fetal brain imaging. Scanning the fetal brain is essential as approximately 3 in 1000 pregnancies have fetuses with different types of brain abnormalities. Moreover, several series of neuropathological variations happen, some related to serious clinical morbidities. Recently, fetal MRI has been considered as a neoteric, non-invasive tool for monitoring fetal brain development. Fetal MRI confers high contrast resolution for the tissue. Therefore, it confers detailed fetal structural information of the brain. Machine learning techniques can be used to detect and classify brain abnormalities and tumors from fetal MRI images at an early stage without any surgical interventions. Early detection of the fetal brain abnormalities will indicate how the pregnancy will be managed, possible treatments that can be taken, and help parents understand and be prepared for dealing with the abnormality. Moreover, the early discovery of these abnormalities can improve the quality of diagnosis and follow-up planning. Machine learning is a well-known field in engineering and computer science. It is a great tool used in pattern recognition and image processing. It can be applied for medical images to help physicians in rendering medical diagnoses. Also, it prevents human-based diagnostic error and the effort during an examination. It can be used as well for fetal brain MRI images for early detection of abnormalities, diagnosis, and classification of certain diseases. Most of the work that used fetal MRI images focused on segmentation. However, few works used fetal MRI images for classification. To our knowledge, papers that used fetal brain images for classification are limited. Although these papers used fetal brain MRI images for classification, they used these images to predict an abnormal neuro-behavior in small for gestational age (SGA) neonates, not fetuses. This means that they classify the SGA abnormality after the fetuses are born and become neonates. The authors of these papers assumed the hypothesis that fetal MRI textural patterns are related to new-born neurobehavioral. Machine learning is an exciting field of research in computer science and engineering. It is considered a branch of artificial intelligence because it enables the extraction of meaningful patterns from examples, which is a component of human intelligence.

CHAPTER- 2

LITERATURE SURVEY

- 1)** They addressed this issue by automatic localization of the fetal anatomy, in particular, the brain which is a structure of interest for many fetal MRI studies. We first extract super pixels followed by the computation of a histogram of features for each super pixel using bag of words based on dense scale invariant feature transform (DSIFT) descriptors. We construct a graph of super pixels and train a random forest classifier to distinguish between brain and non-brain super pixels.
- 2)** The proposed system, at first the self-organizing map neural network trains the features extracted from the discrete wavelet transform blend wavelets and the resultant filter factors are consequently trained by the K-nearest neighbor and the testing process is also accomplished in two stages. The proposed two-tier classification system classifies the brain tumours in double training process which gives preferable performance over the traditional classification method. The proposed system has been validated with the support of real data sets and the experimental results showed enhanced performance .
- 3)** This study, they apply machine-learning methods to compare whole-brain functional connectivity in preterm infants at termequivalent age and healthy term-born neonates in order to test the hypothesis that preterm birth results in specific alterations to functional connectivity by term-equivalent age.
- 4)** They propose a novel multi kernel support vector machine classification framework by using the connectivity features gathered from WM connectivity networks, which are generated via multiscale regions of interest (ROIs) and multiple diffusion statistics such as fractional anisotropy, mean diffusivity, and average fiber length.

- 5) Machine learning is a technique for recognizing patterns that can be applied to medical images. Although it is a powerful tool that can help in rendering medical diagnoses, it can be misapplied. Machine learning typically begins with the machine learning algorithm system computing the image features that are believed to be of importance in making the prediction or diagnosis of interest. The machine learning algorithm system then identifies the best combination of these image features for classifying the image or computing some metric for the given image region. There are several methods that can be used, each with different strengths and weaknesses. There are open-source versions of most of these machine learning methods that make them easy to try and apply to images. Several metrics for measuring the performance of an algorithm exist; however, one must be aware of the possible associated pitfalls that can result in misleading metrics. More recently, deep learning has started to be used; this method has the benefit that it does not require image feature identification and calculation as a first step; rather, features are identified as part of the learning process. Machine learning has been used in medical imaging and will have a greater influence in the future. Those working in medical imaging must be aware of how machine learning works.
- 6) Recent resting-state functional MRI investigations have demonstrated that much of the large-scale functional network architecture supporting motor, sensory and cognitive functions in older pediatric and adult populations is present in term- and prematurely-born infants. Application of new analytical approaches can help translate the improved understanding of early functional connectivity provided through these studies into predictive models of neuro developmental outcome. One approach to achieving this goal is multivariate pattern analysis, a machine-learning, pattern classification approach well-suited for high dimensional neuro imaging data. It has previously been adapted to predict brain maturity in children and adolescents using structural and resting state-functional MRI data.
- 7) They present a technique for real-time adaptive thresholding using the integral image of the input. Our technique is an extension of a previous method. However, our solution is more robust to illumination changes in the image. Additionally, our method is simple

and easy to implement. Our technique is suitable for processing live video streams at a real-time frame-rate, making it a valuable tool for interactive applications such as augmented reality. Source code is available online.

- 8)** The purpose of this study was to describe fetal brain anomalies identified during nuchal translucency (NT) examination and their clinical management. The current report highlights the evolving ability to detect CNS malformation during NT scan. FTS novel findings expand our embryological understanding of early fetal development. While severe anomalies can be managed by early termination, other defects necessitate comprehensive evaluation and repetitive imaging.
- 9)** We tested the hypothesis whether texture analysis (TA) from MR images could identify patterns associated with an abnormal neurobehavior in small for gestational age (SGA) neonates. Ultra sound .Ultra sound and MRI were performed on 91 SGA fetus at 37 weeks of GA. Frontal lobe, basal ganglia, mesencephalon and cerebellum were delineated from fetal MRIs. SGA neonates underwent NBAS test and were classified as abnormal if \$1 area was ,5th centile and as normal if all areas were .5th centile. Textural features associated with neurodevelopment were selected and machine learning was used to model a predictive algorithm. Fetal brain MRI textural patterns were associated with neonatal neurodevelopment. Brain MRI TA could be a useful tool to predict abnormal neurodevelopment in SGA.
- 10)** In this paper, we present a fully automatic brain tumor segmentation method based on Deep Neural Networks (DNNs). The proposed networks are tailored to glioblastomas

CHAPTER- 3

AIM AND SCOPE

3.1 AIM AND SCOPE OF PRESENT INVESTIGATION :

- The main objective of this paper is to detect the abnormalities present in the fetal brain MRI.
- Automatic defects detection in MR images is very important in many diagnostic and therapeutic applications.
- Because of high quantity data in MR images and blurred boundaries, tumor segmentation and classification are very hard.
- This work has introduced one automatic brain tumor detection method to increase the accuracy and yield and decrease the diagnosis time.
- The goal is classifying the fetal brain and detecting the abnormalities. The diagnosis method consists of four stages, pre-processing of MR images, feature extraction, and classification.
- After pre-processing of image, the features are extracted by the neural network itself. In the classification stage, Convolutional Neural Network are employed to classify the Normal and abnormal brain.

3.2 SCOPE OF PRESENT INVESTIGATION :

The main contributions of this project therefore are

- Data Analysis
- Dataset Pre-processing
- Training the Model
- Testing of Dataset

3.3 DOMAIN OVERVIEW OF PRESENT INVESTIGATION :

- Deep neural networks are now the state-of-the-art machine learning models across a variety of areas, from image analysis to natural language processing, and widely deployed in academia and industry.
- These developments have a huge potential for medical imaging technology, medical data analysis, medical diagnostics and healthcare in general, slowly being realized. We provide a short overview of recent advances and some associated challenges in machine learning applied to medical image processing and image analysis.
- Long before deep learning was used, traditional machine learning methods were mainly used. Such as Decision Trees, SVM, Naïve Bayes Classifier and Logistic Regression.
- These algorithms are also called flat algorithms. Flat here means that these algorithms cannot normally be applied directly to the raw data (such as .csv, images, text, etc.). We need a pre-processing step called Feature Extraction.
- The result of Feature Extraction is a representation of the given raw data that can now be used by these classic machine learning algorithms to perform a task. For example, the classification of the data into several categories or classes.
- Feature Extraction is usually quite complex and requires detailed knowledge of the problem domain. This pre-processing layer must be adapted, tested and refined over several iterations for optimal results.
- On the other side are the artificial neural networks of Deep Learning. These do not need the Feature Extraction step. The layers are able to learn an implicit representation of the raw data directly and on their own. Here, a more and more abstract and compressed representation of the raw data is produced over several layers of an artificial neural-nets. This compressed representation of the input data is then used to produce the result. The result can be, for example, the classification of the input data into different classes.

CHAPTER-4

MATERIALS AND METHODS

4.1 METHODOLOGY

- In our proposed framework, we take the raw MRI image, then we train some of the raw images in order to classify between abnormal brain and normal brain.
- After taking the raw image we first the image in order to remove the noisy data and resizing of the image.
- Then the CNN classifier is there to distinguish the image between the classes. The block diagram of proposed framework is presented Block diagram of Proposed Algorithm Then the database image will be directed to the classifier after the feature extraction phase in order to classify between the classes.
- The database images are the test images that we have not trained

4.1.1 BLOCK DIAGRAM WITH EXPLANATION

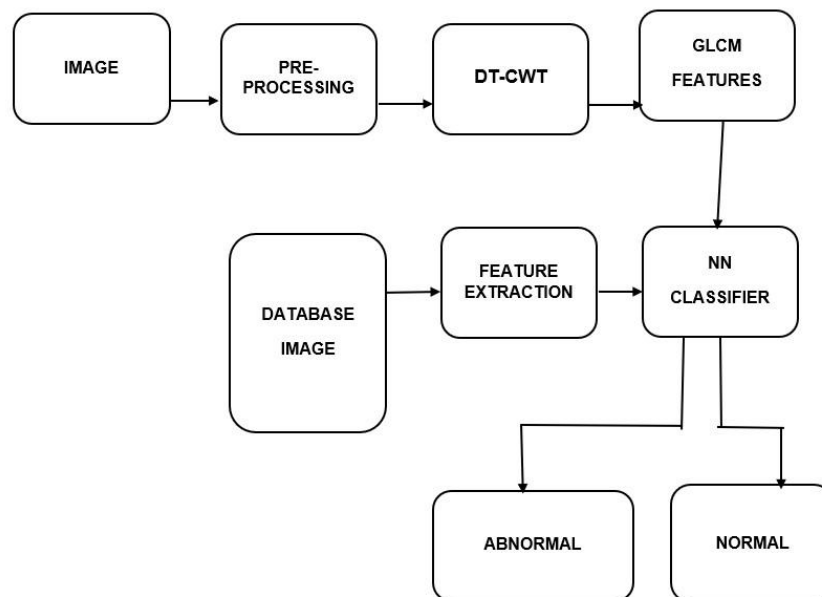


FIG 4.1.1 BLOCK DIAGRAM WITH EXPLANATION

In our proposed framework, we take the raw MRI image, then we train some of the raw images in order to classify between abnormal brain and normal brain. So, after taking the raw image we first the image in order to remove the noisy data and resizing of the image. Then the image will go to DT- CWT phase in order to extract the structure and features. Then The image will pass through the GLCM phase in order to extract texture features of the image like Energy, Entropy, Contrast, Homogeneity etc. Then the CNN classifier is there to distinguish the image between the classes. The block diagram of proposed framework is presented in Fig. 4.1.1. Then the database image will be directed to the classifier after the feature extraction phase in order to classify between the classes. The database images are the test images that we have not trained. Each module of the proposed system is explained in the following sections.

4.2 DIGITAL IMAGE PROCESSING:

4.2.1 CLASSIFICATION OF IMAGES

There are 3 kinds of pix applied in Digital Image Processing.

They are,

- Binary Image
- Gray Scale Image
- Colour Image

Binary picture

A binary photograph is a virtual photo that has most effective viable values for every pixel. Typically the two colorings used for a binary image are black and white despite the truth that any colors may be used. The color used for the item(s) in the photograph is the foreground color while the rest of the picture is the historic past coloration.

This name black and white, monochrome or monochromatic are regularly used for this idea, however may also additionally designate any pics that have best

one pattern in keeping with pixel, together with grayscale photos

Binary pics often upward push up in virtual image processing as masks or because the result of positive operations collectively with segmentation, thresholding, and dithering. Some input/output gadgets, which include laser printers, fax machines, and bi-level laptop shows, can most effective manage bi-degree pix.

- **Gray Scale Image**

A grayscale Image is virtual photo is an picture in which the price of every pixel is a single sample, this is, it consists of satisfactory depth information. Images of this type, also referred to as black-and-white, are composed solely of sun shades of gray(0-255), various from black(0) on the weakest depth to white(255) at the maximum effective.

Grayscale snap shots are extremely good from one-bit black-and-white pix, which in the context of laptop imaging are pix with only the 2 sun sunglasses, black, and white (additionally known as bi-diploma or binary photos). Grayscale snap shots have many sun shades of grey in among. Grayscale photos are also referred to as monochromatic, denoting the absence of any chromatic version.

Grayscale pix are frequently the quit end result of measuring the intensity of moderate at every pixel in a single band of the electromagnetic spectrum (e.G. Infrared, seen mild, ultraviolet, and lots of others.), and in such instances they will be monochromatic proper at the same time as simplest a given frequency is captured. But moreover they may be synthesized from a whole color picture; see the phase approximately changing to grayscale.

- **Color picture**

A (digital) colour image is a virtual photograph that includes shade

records for every pixel. Each pixel has a specific rate which determines its appearing shade. This fee is certified by means of manner of 3 numbers giving the decomposition of the color inside the 3 number one sun shades Red, Green and Blue. Any shade seen to human eye may be represented this manner. The decomposition of a colour inside the 3 number one hues is quantified through some of amongst 0 and 255. For instance, white may be coded as $R = 255$, $G = 255$, $B = 255$; black can be referred to as $(R,G,B) = (0,0,0)$; and say, amazing pink may be : $(255,0,255)$.

In great phrases, an photo is an intensive -dimensional array of colour values, pixels, each of them coded on three bytes, representing the 3 primary shades. This allows the image to encompass a complete of $256 \times 256 \times 256 = 16.8$ million particular shades. This approach is likewise referred to as RGB encoding, and is particularly tailor-made to human imaginative and prescient.

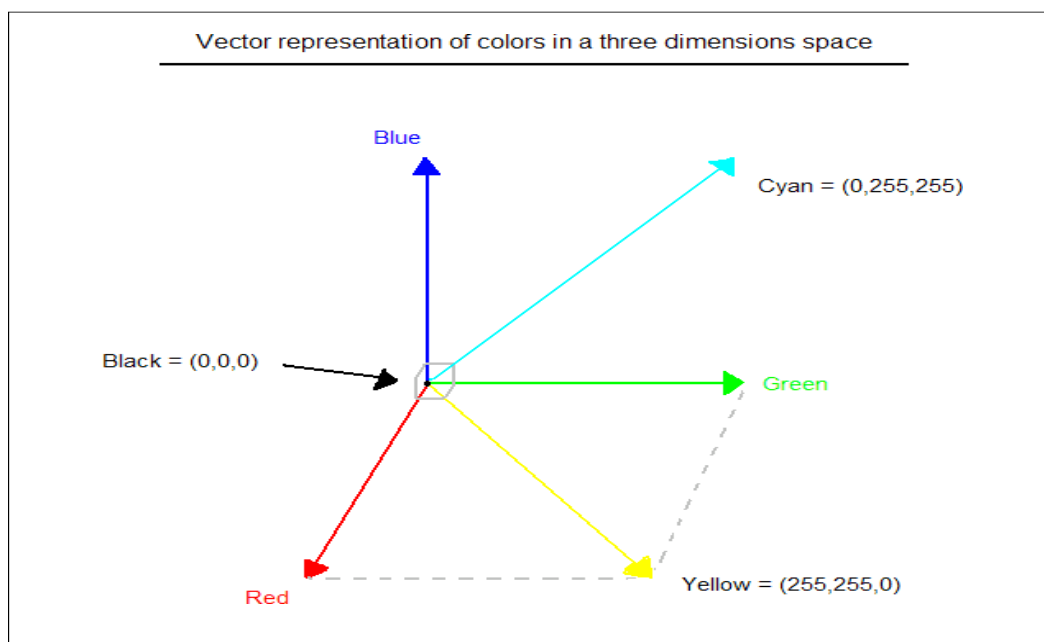


Fig 4.2. Hue Saturation Process of RGB SCALE Image

From the above determine, sun shades are coded on 3 bytes representing their decomposition on the 3 primary colorings. It sounds obvious to a mathematician to right away interpret colors as vectors in a three size area in which every axis stands for one of the number one hues. Therefore we are able to gain of maximum of the geometric mathematical requirements to cope with our colors, which consist of norms, scalar product, projection, rotation or distance. The identification of devices in an picture and this method may additionally possibly begin with photograph processing techniques collectively with noise removal, observed via the usage of (low-diploma) feature extraction to find out lines, areas and probable regions with sure textures.

The smart bit is to interpret collections of those shapes as single devices, e.G. Vehicles on a avenue, boxes on a conveyor belt or cancerous cells on a microscope slide. One motive this is an AI problem is that an item can seem very one-of-a-type while viewed from unique angles or underneath amazing lighting. Another problem is finding out what skills belong to what item and which can be information or shadows and so on. The human visible device plays these responsibilities specifically unconsciously however a laptop calls for skillful programming and masses of processing strength to method human commonplace overall performance. Manipulation of records inside the shape of an photograph through severe viable strategies. An image is generally interpreted as a dimensional array of brightness values, and is maximum. familiarly represented via way of such patterns as the ones of a photographic print, slide, tv display, or movie show display. An image may be processed optically or digitally with a laptop.

✓ IMAGE ACQUISITION

Image Acquisition is to collect a digital photograph. To collect this requires an picture sensor and the functionality to digitize the sign produced thru the sensor. The sensor might be monochrome or coloration TV camera that produces an entire photo of the trouble area each 1/30 sec. The photograph sensor may also be line test virtual digicam that produces a single photo line at a time. In this situation, the gadgets movement beyond the road.

✓ **IMAGE ENHANCEMENT**

Image enhancement is most of the best and maximum appealing regions of digital image processing. Basically, the concept in the back of enhancement strategies is to perform detail that is obscured, or clearly to spotlight certain features of exciting an picture. A familiar instance of enhancement is at the same time as we growth the assessment of an photograph due to the fact “it appears higher.” It is essential to remember the fact that enhancement is a completely subjective place of photo processing.

✓ **IMAGE RESTORATION**

Image recuperation is a place that also deals with enhancing the appearance of an image. However, not like enhancement, that is subjective, photograph recuperation is goal, in the experience that restoration techniques will be predisposed to be based on mathematical or probabilistic fashions of picture degradation.

4.3 SYSTEM REQUIREMENT:

HARD DISK : 1 TB
RAM : 8GB
MONITOR / LAPTOP

4.4 SOFTWARE USED:

MATLAB 2018B

4.4.1 MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or FORTRAN.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB uses software developed by the LAPACK and ARPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

4.4.2 THE MATLAB SYSTEM

The MATLAB system consists of five main parts:

- Development Environment: This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, and browsers for viewing help, the workspace, files, and the search path.
- The MATLAB Mathematical Function Library: This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.
- The MATLAB Language: This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.
- Handle Graphics®: This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.
- The MATLAB Application Program Interface (API): This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

4.4.3 STARTING AND QUITTING MATLAB

This chapter provides a brief introduction to starting and quitting MATLAB, and the tools and functions that help you to work with MATLAB variables and files. For more

information about the topics covered here, see the corresponding topics under Development Environment in the MATLAB documentation, which is available online as well as in print.

STARTING MATLAB

- On a Microsoft Windows platform, to start MATLAB, double-click the MATLAB shortcut icon on your Windows desktop.
- On a UNIX platform, to start MATLAB, type `matlab` at the operating system.

QUITTING MATLAB

- To end your MATLAB session, select Exit MATLAB from the File menu in the desktop, or type `quit` in the Command Window. To execute specified functions each time MATLAB quits, such as saving the workspace, you can create and run a finish `.m` script.

4.4.4 MATLAB DESKTOP

When you start MATLAB, the MATLAB desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB. The first time MATLAB starts, the desktop appears as shown in the following illustration, although your Launch Pad may contain different entries.

You can change the way your desktop looks by opening, closing, moving, and resizing the tools in it. You can also move tools outside of the desktop or return them back inside the desktop (docking). All the desktop tools provide common features such as context menus and keyboard shortcuts.

You can specify certain characteristics for the desktop tools by selecting Preferences from the File menu. For example, you can specify the font characteristics for Command Window text. For more information, click the Help button in the Preferences dialog box.

Desktop Tools:

This section provides an introduction to MATLAB's desktop tools. You can also

use MATLAB functions to perform most of the features found in the desktop tools. The tools are:

Command Window

Use the Command Window to enter variables and run functions and M-files.

Command History

Lines you enter in the Command Window are logged in the Command History window. In the Command History, you can view previously used functions, and copy and execute selected lines. To save the input and output from a MATLAB session to a file, use the diary function.

Running External Programs

You can run external programs from the MATLAB Command Window. The exclamation point character! is a shell escape and indicates that the rest of the input line is a command to the operating system.

Its useful for invoking utilities or running other programs without quitting MATLAB. On Linux, for example! Emacsmagik .m invokes an editor called emacs for a file named magik.m. When you quit the external program, the operating system returns control to MATLAB.

Launch Pad

MATLAB's Launch Pad provides easy access to tools, demos, and documentation.

Help Browser

Use the Help browser to search and view documentation for all your Math Works products. The Help browser is a Web browser integrated into the MATLAB desktop that displays HTML documents. To open the Help browser, click the help button in the toolbar, or type help browser in the Command Window. The Help browser consists of two panes, the

Help Navigator, which you use to find information, and the display pane, where you view the information.

Help Navigator

Use the Help Navigator to find information. It includes:

- **Product filter** - Set the filter to show documentation only for the products you specify.
- **Contents tab** - View the titles and tables of contents of documentation for your products.
- **Index tab** - Find specific index entries (selected keywords) in the MathWorks documentation for your products.
- **Search tab** - Look for a specific phrase in the documentation. To get help for a specific function, set the Search type to Function Name.
- **Favorites tab** - View a list of documents you previously designated as favorites.
- **Browse to other pages** - Use the arrows at the tops and bottoms of the pages, or use the back and forward buttons in the toolbar.
- **Bookmark pages** - Click the Add to Favorites button in the toolbar.
- **Print pages** - Click the print button in the toolbar.
- **Find a term in the page** - Type a term in the Find in page field in the toolbar and click Go.

Other features available in the display pane are: copying information, evaluating a selection, and viewing Web pages.

✓ *Current Directory Browser*

MATLAB file operations use the current directory and the search path as reference points. Any file you want to run must either be in the current directory or on the search path.

✓ *Search Path*

To determine how to execute functions you call, MATLAB uses a search path to find M-files and other MATLAB-related files, which are organized in directories on your file system. Any file you want to run in MATLAB must reside in the current directory or in a directory that is on the search path. By default, the files supplied with MATLAB and MathWorks toolboxes are included in the search path.

✓ *Workspace Browser*

The MATLAB workspace consists of the set of variables (named arrays) built up during a MATLAB session and stored in memory. You add variables to the workspace by using functions, running M-files, and loading saved workspaces.

To view the workspace and information about each variable, use the Workspace browser, or use the functions `who` and `whos`.

To delete variables from the workspace, select the variable and select Delete from the Edit menu. Alternatively, use the `clear` function.

✓ *Array Editor*

Double-click on a variable in the Workspace browser to see it in the Array

Editor. Use the Array Editor to view and edit a visual representation of one- or two-dimensional numeric arrays, strings, and cell arrays of strings that are in the workspace.

✓ *Editor/Debugger*

Use the Editor/Debugger to create and debug M-files, which are programs you write to run MATLAB functions. The Editor/Debugger provides a graphical user interface for basic text editing, as well as for M-file debugging. You can use any text editor to create M-files, such as Emacs, and can use preferences (accessible from the desktop File menu) to specify that editor as the default. If you use another editor, you can still use the MATLAB Editor/Debugger for debugging, or you can use debugging functions, such as `dbstop`, which sets a breakpoint.

4.4.5 MANIPULATING MATRICES:

ENTERING MATRICES

The best way for you to get started with MATLAB is to learn how to handle matrices. Start MATLAB and follow along with each example.

You can enter matrices into MATLAB in several different ways:

- Enter an explicit list of elements.
- Load matrices from external data files.
- Generate matrices using built-in functions.
- Create matrices with your own functions in M-files.

Start by entering Dürer's matrix as a list of its elements. You have only to follow a few basic

conventions:

- Separate the elements of a row with blanks or commas.
- Use a semicolon, ; , to indicate the end of each row.
- Surround the entire list of elements with square brackets, [].

To enter Dürer's matrix, simply type in the Command Window

```
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

MATLAB displays the matrix you just entered.

This exactly matches the numbers in the engraving. Once you have entered the matrix, it is automatically remembered in the MATLAB workspace. You can refer to it simply as A.

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

- ***Expressions***

Like most other programming languages, MATLAB provides mathematical expressions, but unlike most programming languages, these expressions involve entire matrices. The building blocks of expressions are:

- ***Variables***

MATLAB does not require any type declarations or dimension statements. When MATLAB encounters a new variable name, it automatically creates the variable and allocates the appropriate amount of storage. If the variable already exists,

MATLAB changes its contents and, if necessary, allocates new storage. For example, `num_students = 25` Creates a 1-by-1 matrix named `num_students` and stores the value 25 in its single element.¹ Variable names consist of a letter, followed by any number of letters, digits, or underscores. MATLAB uses only the first 31 characters of a variable name. MATLAB is case sensitive; it distinguishes between uppercase and lowercase letters. `A` and `a` are not the same variable. To view the matrix assigned to any variable, simply enter the variable name.

- ***Functions***

MATLAB provides a large number of standard elementary mathematical functions, including `abs`, `sqrt`, `exp`, and `sin`. Taking the square root or logarithm of a negative number is not an error; the appropriate complex result is produced automatically. MATLAB also provides many more advanced mathematical functions, including Bessel and gamma functions. Most of these functions accept complex arguments. For a list of the elementary mathematical functions, type.

4.4.6 GUI

A graphical user interface (GUI) is a user interface built with graphical objects, such as buttons, text fields, sliders, and menus. In general, these objects already have meanings to most computer users. For example, when you move a slider, value changes; when you press an OK button, your settings are applied and the dialog box is dismissed. Of course, to leverage this built-in familiarity, you must be consistent in how you use the various GUI-building components. Applications that provide GUIs are generally easier to learn and use since the person using the application does not need to know what commands are available or how they work. The action that results from a particular user action can be made clear by the design of the interface. The sections that follow describe how to create GUIs with MATLAB. This includes laying out the components, programming them to do specific things in response to user actions, and saving and launching the GUI; in other words, the mechanics of creating GUIs. This documentation does not attempt to cover the "art" of good user interface design, which is an entire field unto itself. Topics covered in this section include:

- *Creating GUIs with GUIDE:*

MATLAB implements GUIs as figure windows containing various styles of GUI

control objects. You must program each object to perform the intended action when activated by the user of the GUI. In addition, you must be able to save and launch your GUI. All of these tasks are simplified by GUIDE, MATLAB's graphical user interface development environment.

- *GUI Development Environment:*

The process of implementing a GUI involves two basic tasks:

- ✓ Laying out the GUI components
- ✓ Programming the GUI component.

GUIDE primarily is a set of layout tools. However, GUIDE also generates an M-file that contains code to handle the initialization and launching of the GUI. This M-file provides a framework for the implementation of the call backs - the functions that execute when users activate components in the GUI.

Features of the GUIDE-Generated Application M-File:

GUIDE simplifies the creation of GUI applications by automatically generating an M-file framework directly from your layout. You can then use this framework to code your application M-file. This approach provides a number of advantages:

The M-file contains code to implement a number of useful features (see Configuring Application Options for information on these features). The M-file adopts an effective approach to managing object handles and executing call back routines (see Creating and Storing the Object Handle Structure for more information). The M-files provides a way to manage global data (see Managing GUI Data for more information).

Beginning the Implementation Process:

To begin implementing your GUI, proceed to the following sections:

- ✓ Getting Started with GUIDE - the basics of using GUIDE
- ✓ *Selecting GUIDE Application Options* - set both FIG-file and M-file options.
- ✓ *Using the Layout Editor* - begin laying out the GUI.

Understanding the Application M-File - discussion of programming techniques

Application Examples - a collection of examples that illustrate techniques Which are useful for implementing GUIs.

Command-Line Accessibility:

When MATLAB creates a graph, the figure and axes are included in the list of children of their respective parents and their handles are available through commands such as `find`, `set`, and `get`. If you issue another plotting command, the output is directed to the current figure and axes.

GUIs are also created in figure windows. Generally, you do not want GUI figures to be available as targets for graphics output, since issuing a plotting command could direct the output to the GUI figure, resulting in the graph appearing in the middle of the GUI.

In contrast, if you create a GUI that contains axes and you want commands entered in the command window to display in this axes, you should enable command-line access.

User Interface Controls:

The Layout Editor Component palette contains the user interface controls that you can use in your GUI. These components are MATLAB uicontrol objects and are programmable via their Call back properties. This section provides information on these components.

Push Buttons:

Push buttons generate an action when pressed (e.g., an OK button may close a dialog box and apply settings). When you click down on a push button, it appears depressed; when you release the mouse, the button's appearance returns to its non depressed state; and its call back executes on the button up event.

Properties to Set:

- ✓ *String* - set this property to the character string you want displayed on the push button.
- ✓ *Tag* - GUIDE uses the Tag property to name the call-back sub function in the application M-file. Set Tag to a descriptive name (e.g., close button) before activating the GUI.

Programming the Call-back:

When the user clicks on the push button, its call back executes. Push buttons do not return a value or maintain a state toggle buttons.

Toggle Buttons:

Toggle buttons generate an action and indicate a binary state (e.g., on or off). When you click on a toggle button, it appears depressed and remains depressed when you release the mouse button, at which point the call-back executes. A subsequent mouse click returns the toggle button to the non depressed state and again executes its call back.

Programming the Call back:

The call back routine needs to query the toggle button to determine what state it is in. MATLAB sets the Value property equal to the Max property when the toggle button is depressed (Max is 1 by default) and equal to the Min property when the toggle button is not depressed (Min is 0 by default).

Radio Buttons:

Radio buttons are similar to checkboxes, but are intended to be mutually exclusive within a group of related radio buttons (i.e., only one button is in a selected state at any given time). To activate a radio button, click the mouse button on the object. The display indicates the state of the button.

Checkboxes:

Check boxes generate an action when clicked and indicate their state as checked or not checked. Check boxes are useful when providing the user with a number of independent choices that set a mode (e.g., display a toolbar or generate call back function prototypes).

Edit Text:

Edit text controls are fields that enable users to enter or modify text strings. Use edit text when you want text as input. The String property contains the text entered by the user.

Axes:

Axes enable your GUI to display graphics (e.g., graphs and images). Like all graphics objects, axes have properties that you can set to control many aspects of its behaviour and appearance. See Axes Properties for general information on axes objects.

Figure:

Figures are the windows that contain the GUI you design with the Layout Editor. See the description of figure properties for information .

4.5 MODULES DESCRIPTION:

- **PRE-PROCESSING**
- **DT-CWT METHOD**
- **GLCM FEATURE EXTRACTION**
- **NEURAL NETWORK**

4.5.1 PRE-PROCESSING

Digital image processing deals with manipulation of digital images through a digital computer. It is a subfield of signals and systems but focus particularly on images. DIP focuses on developing a computer system that is able to perform processing on an image. The input of that system is a digital image and the system process that image using efficient algorithm.

It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and distortion during processing.

- i. Importing the image via image acquisition tools;
- ii. Analyzing and manipulating the image;
- iii. Output in which result can be altered image

Image Pre-processing is a common name for operations with images at the lowest level of abstraction. Its input and output are intensity images. The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.

- The aim of pre-processing is an improvement of the signal data that suppresses unwanted distortions or enhances some signal features important for further processing.
- The high-pass pre-emphasis filter can then be applied to equalize the energy between the low and high frequency components of speech.
- In face recognition the pre-processing suppresses some of the unwanted pixels and

colour conversion in finger print detection and resize the pixels and apply some of the filter to reduce the noise over it.

- Image pre-processing is the operation of taking a corrupted/noisy image and estimating the clean original image. Corruption may come in many forms such as motion blur, noise, and camera mis focus.

4.5.2 DT-CWT Discrete Wavelet Transform (DWT)

The discrete wavelet transform (DWT) became superior to use the wavelet rework to the digital international. Filter banks are used to approximate the behavior of the non- prevent wavelet transform. The signal is decomposed with a immoderate-skip smooth out and a low-bypass clear out. The coefficients of these filters are computed using mathematical evaluation and made to be had to you. See Appendix B for more records about those computations.

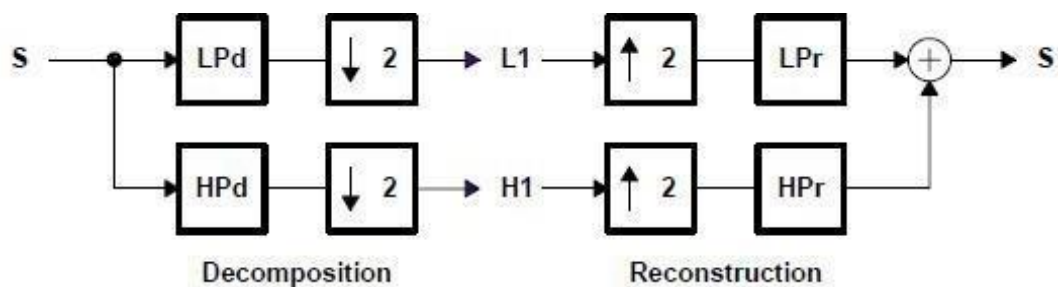


Fig 4.5.2 Discrete Wavelet Transform

The wavelet literature offers the filter coefficients to you in tables. An example is the Daubechies filters for wavelets. These filters rely upon a parameter p called the vanishing 2nd.

▪ Wavelets Image Processing

Wavelets have located a massive type of programs within the photo processing discipline. The JPEG 2000 famous uses wavelets for photo compression. Other photo processing programs which incorporates noise reduction, issue detection, and finger print evaluation have additionally been investigated inside the literature.

▪ Wavelet Decomposition of Images

In wavelet decomposing of an photo, the decomposition is achieved row thru row after which column through column. For example, proper here is the technique for an $N \times M$ photograph. You filter out every row after which down-sample to acquire $N \times (M/2)$ pictures. Then clear out each column and subsample the filter output to attain four $(N/2) \times (M/2)$ pictures of the four sub snap shots obtained as visible in Figure 12, the most effective acquired through the use of low-pass filtering the rows and columns is referred to as the LL image.

The one acquired with the aid of way of low-skip filtering the rows and high-pass filtering the columns is referred to as the LH snap shots. The one obtained thru immoderate-bypass filtering the rows and espresso-skip filtering the columns is referred to as the HL photo. The sub image received by using manner of the use of immoderate- bypass filtering the rows and columns is referred to as the HH photograph. Each of the sub snap shots obtained on this fashion can then be filtered and sub sampled to acquire 4 extra sub pictures. This machine can be persisted till the popular sub band form is acquired.

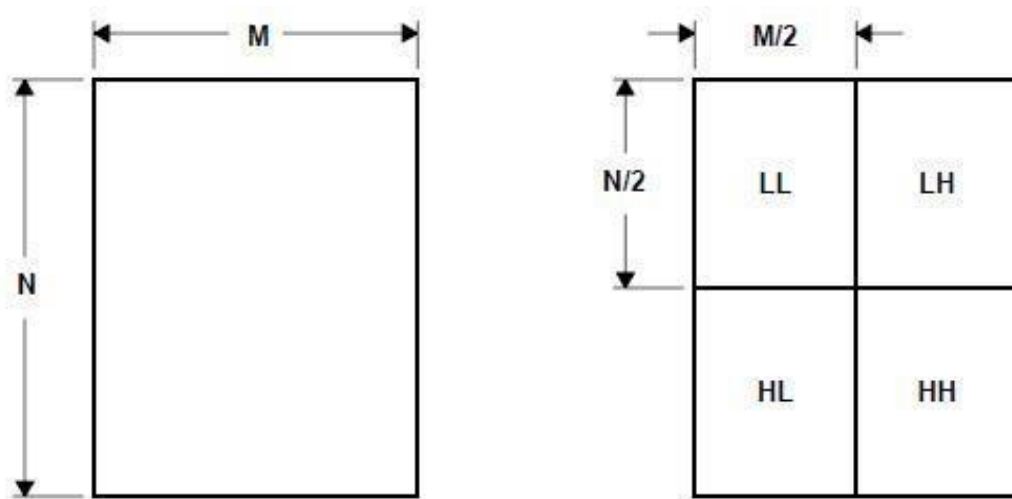


Fig 4.5.3 Original Image one level 2-D Decomposition

Three of the maximum well-known strategies to decompose an photograph are: pyramid, space, and wavelet packet, as shown in Figure below.

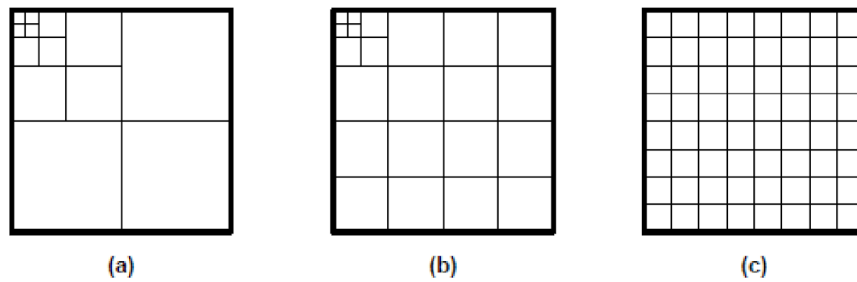


Fig 4.5.4 Three popular wavelet decomposition structures on image

- In the form of pyramid decomposition, most effective the LL subimage is decomposed after each decomposition into four greater subimages.
- In the form of wavelet packet decomposition, every subimage(LL, LH,HL, HH) is decomposed after every decomposition.
- In the shape of spacl, after the primary diploma of decomposition, each subimage is decomposed into smaller subimages, after which only the LL subimage is decomposed.

Fig 4.5.5 shows a three-diploma decomposition image of pyramid form.

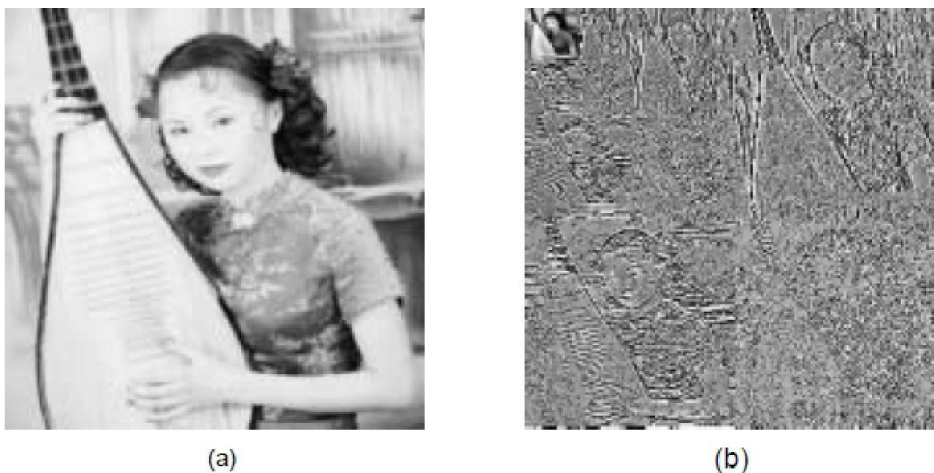


Fig 4.4.5 a) original image

b) three level pyramid structure decomposition.

In the thing I development level, the JPEG 2000 modern-day lets in the pyramid decomposition form. In the destiny all three structures may be supported.

For dimensions, the C55x IMGLIB offers capabilities for pyramid and packet

decomposition and reconstruction. Complete data approximately those functions may be determined within the C55x IMGLIB.

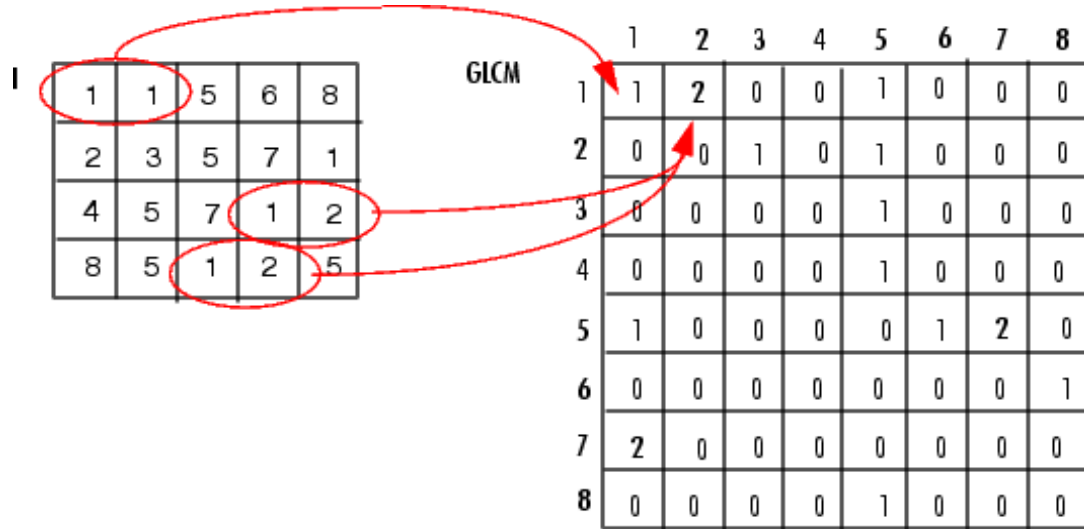
4.5.3 Gray-Level Co-Occurrence Matrix:

To create a GLCM, use the `graycomatrix` function. The `graycomatrix` function creates a gray-level co-occurrence matrix (GLCM) by calculating how often a pixel with the intensity (gray-level) value i occurs in a specific spatial relationship to a pixel with the value j .

default, the spatial relationship is defined as the pixel of interest and the pixel to its immediate right (horizontally adjacent), but you can specify other spatial relationships between the two pixels. Each element (i,j) in the resultant GLCM is simply the sum of the number of times that the pixel with value i occurred in the specified spatial relationship to a pixel with value j in the input image.

The processing required to calculate a GLCM for the full dynamic range of an image is prohibitive, `graycomatrix` scales the input image. By default, `graycomatrix` uses scaling to reduce the number of intensity values in gray scale image from 256 to eight. The number of gray levels determines the size of the GLCM. To control the number of gray levels in the GLCM and the scaling of intensity values, using the `Num Levels` and the `Gray Limits` parameters of the `graycomatrix` function.

The gray-level co-occurrence matrix can reveal certain properties about the spatial distribution of the gray levels in the texture image. For example, if most of the entries in the GLCM are concentrated along the diagonal, the texture is coarse with respect to the specified offset. To illustrate, the following figure shows how `graycomatrix` calculates the first three values in a GLCM. In the output GLCM, element $(1,1)$ contains the value 1 because there is only one instance in the input image where two horizontally adjacent pixels have the values 1 and 1, respectively.



To create multiple GLCMs, specify an array of offsets to the `graycomatrix` function. These offsets define pixel relationships of varying direction and distance. For example, you can define an array of offsets that specify four directions (horizontal, vertical, and two diagonals) and four distances. In this case, the input image is represented by 16 GLCMs. When you calculate statistics from these GLCMs, you can take the average.

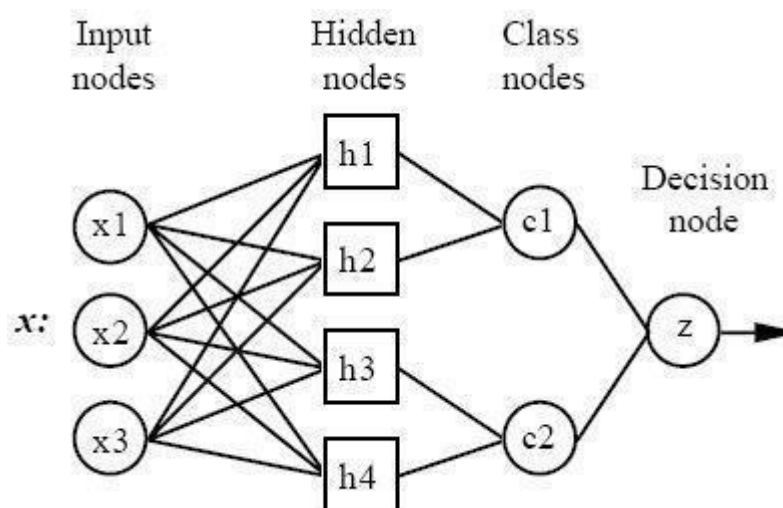
You specify these offsets as a p -by-2 array of integers. Each row in the array is a two- element vector, `[row_offset, col_offset]`, that specifies one offset. Row offset is the number of rows between the pixel of interest and its neighbor. Col_offset is the number of columns between the pixel of interest and its neighbor. This example creates an offset that specifies four directions and 4 distances for each direction. After you create the GLCMs, you can derive several statistics from them using the `graycoprops` function. These statistics provide information about the texture of an image. Statistic such a as Contrasts, Correlation, Energy.

4.5.4 Neural Networks (NN):

Neural Network (NN) and General Regression Neural Networks (GRNN)

have similar architectures, but there is a fundamental difference: networks perform classification where the target variable is categorical, whereas general regression neural networks perform regression where the target variable is continuous. If you select a NN/GRNN network, DTREG will automatically select the correct type of network based on the type of target variable.

4.5.1 Architecture of a NN:



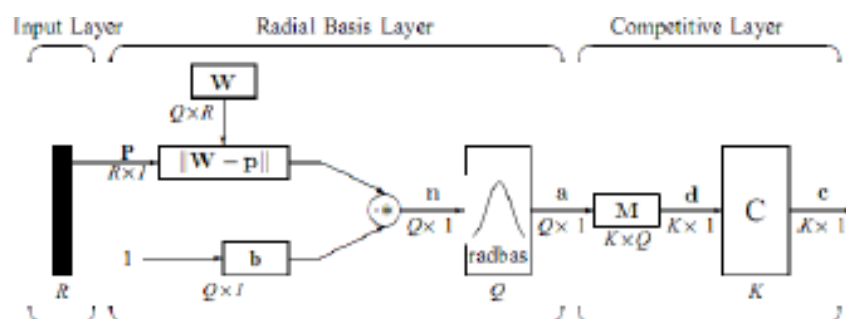
All NN networks have four layers:

- Input layer — There is one neuron in the input layer for each predictor variable. In the case of categorical variables, $N-1$ neurons are used where N is the number of categories. The input neurons (or processing before the input layer) standardizes the range of the values by subtracting the median and dividing by the interquartile range. The input neurons then feed the values to each of the neurons in the hidden layer.
- Hidden layer — This layer has one neuron for each case in the training data set. The neuron stores the values of the predictor variables for the case along with the target value. When presented with the x vector of input values from the input layer, a hidden

neuron computes the Euclidean distance of the test case from the neuron's center point and then applies the RBF kernel function using the sigma value(s). The resulting value is passed to the neurons in the pattern layer.

- Pattern layer / Summation layer — The next layer in the network is different for NN networks and for GRNN networks. For NN networks there is one pattern neuron for each category of the target variable. The actual target category of each training case is stored with each hidden neuron; the weighted value coming out of a hidden neuron is fed only to the pattern neuron that corresponds to the hidden neuron's category. The pattern neurons add the values for the class they represent (hence, it is a weighted vote for that category). For GRNN networks, there are only two neurons in the pattern layer. One neuron is the denominator summation unit the other is the numerator summation unit. The denominator summation unit adds up the weight values coming from each of the hidden neurons. The numerator summation unit adds up the weight values multiplied by the actual target value for each hidden neuron.
- Decision layer — The decision layer is different for NN and GRNN networks. For NN networks, the decision layer compares the weighted votes for each target category accumulated in the pattern layer and uses the largest vote to predict the target category.

For GRNN networks, the decision layer divides the value accumulated in the numerator summation unit by the value in the denominator summation unit and uses the result as the predicted target value.



1. Input Layer

produces the i -th element of the distance vector $\|W-p\|$, whose dimension is $Q \times 1$. The minus symbol, “-”, indicates that it is the distance between vectors. Then, the bias vector b is combined with $\|W-p\|$ by an element-by-element multiplication, .The result is denoted as $n = \|W-p\| \cdot b$. The transfer function in NN has built into a distance criterion with respect to a center. In this paper, it is defined as $radbas(n) = \frac{1}{1 + e^{-2n}}$ (1) Each element of n is substituted into Eq. 1 and produces corresponding element of a , the output vector of Radial Basis Layer. The i -th element of a can be represented as $a_i = radbas(\|W_i - p\| \cdot b_i)$

2. Radial Basis Layer:

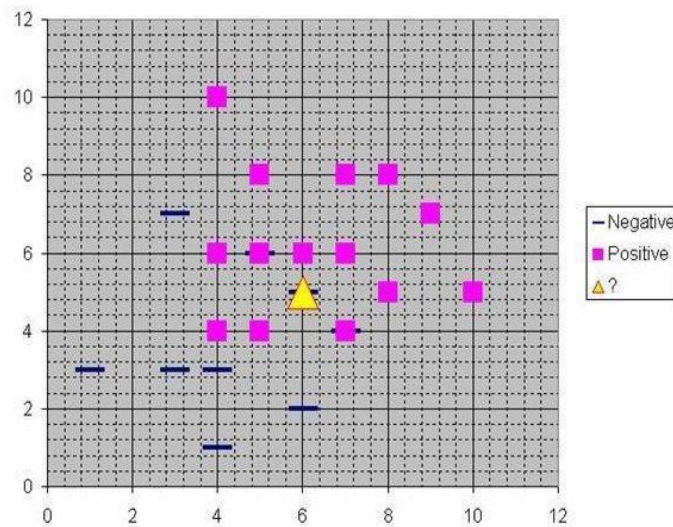
In Radial Basis Layer, the vector distances between input vector p and the weight vector made of each row of weight matrix W are calculated. Here, the vector distance is defined as the dot product between two vectors [8]. Assume the dimension of W is $Q \times R$. The dot product between p and the i -th row of W .

3. Competitive Layer:

There is no bias in Competitive Layer. In Competitive Layer, the vector a is firstly multiplied with layer weight matrix M , producing an output vector d . The competitive function, denoted as C in Fig. 2, produces a 1 corresponding to the largest element of d , and 0's elsewhere. The output vector of competitive function is denoted as c . The index of 1 in c is the number of tumor that the system can classify. The dimension of output vector, K , is 5 in this paper.

4.5.2 How NN network work:

Although the implementation is very different, neural networks are conceptually similar to *K-Nearest Neighbor* (k-NN) models. The basic idea is that a predicted target value of an item is likely to be about the same as other items that have close values of the predictor variables. Consider this figure:



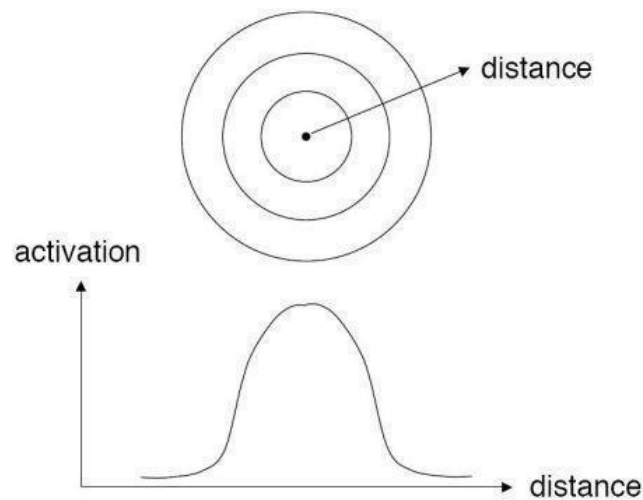
Assume that each case in the training set has two predictor variables, x and y .

The cases are plotted using their x, y coordinates as shown in the figure. Also assume that the target variable has two categories, *positive* which is denoted by a square and *negative* which is denoted by a dash. Now, suppose we are trying to predict the value of a new case represented by the triangle with predictor values $x=6$, $y=5.1$. Should we predict the target as positive or negative?

Notice that the triangle is positioned almost exactly on top of a dash representing a negative value. But that dash is in a fairly unusual position compared to the other dashes which are clustered below the squares and left of center. So it could be that the underlying negative value is an odd case.

The nearest neighbor classification performed for this example depends on how many neighboring points are considered. If 1-NN is used and only the closest point is considered, then clearly the new point should be classified as negative since it is on top of a known negative point. On the other hand, if 9-NN classification is used and the closest 9 points are considered, then the effect of the surrounding 8 positive points may overbalance the close negative point.

A neural network builds on this foundation and generalizes it to consider all of the other points. The distance is computed from the point being evaluated to each of the other points, and a *radial basis function* (RBF) (also called a *kernel function*) is applied to the distance to compute the weight (influence) for each point. The radial basis function is so named because the radius distance is the argument to the function.



$$\text{Weight} = \text{RBF}(\text{distance})$$

The further some other point is from the new point, the less influence it has.

Radial Basis Function

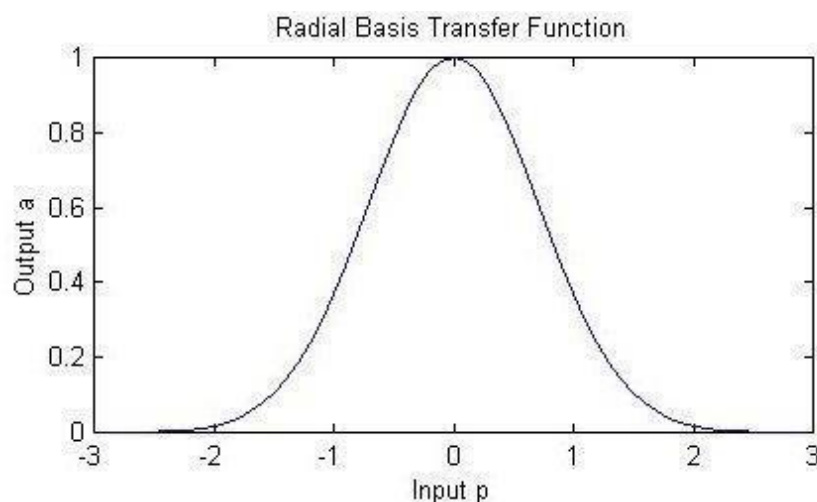
Different types of radial basis functions could be used, but the most common is the Gaussian function:

4.5.3 Advantages and disadvantages of NN networks:

- It is usually much faster to train a NN/GRNN network than a multilayer perceptron network.
- NN/GRNN networks often are more accurate than multilayer perceptron networks.
- NN/GRNN networks are relatively insensitive to outliers (wild points).
- NN networks generate accurate predicted target probability scores.
- NN networks approach Bayes optimal classification.
- NN/GRNN networks are slower than multilayer perceptron networks at classifying new cases.
- NN/GRNN networks require more memory space to store the model.

4.5.4 Removing unnecessary neurons

One of the disadvantages of NN models compared to multilayer perceptron networks is that NN models are large due to the fact that there is one neuron for each training row. This causes the model to run slower than multilayer perceptron networks when using scoring to predict values for new rows.



DTREG provides an option to cause it remove unnecessary neurons from the model after the model has been constructed.

Removing unnecessary neurons has three benefits:

- The size of the stored model is reduced.
- The time required to apply the model during scoring is reduced.
- Removing neurons often improves the accuracy of the model.

The process of removing unnecessary neurons is an iterative process. Leave-one- out validation is used to measure the error of the model with each neuron removed. The neuron that causes the least increase in error (or possibly the largest reduction in error) is then removed from the model. The process is repeated with the remaining neurons until the stopping criterion is reached.

When unnecessary neurons are removed, the “Model Size” section of the analysis report shows how the error changes with different numbers of neurons. You can see a graphical chart of this by clicking Chart/Model size.

- There are three criteria that can be selected to guide the removal of neurons: Minimize error – If this option is selected, then DTREG removes neurons as long as the leave-one-out error remains constant or decreases. It stops when it finds a neuron whose removal would cause the error to increase above the minimum found.
- Minimize neurons – If this option is selected, DTREG removes neurons until the leave-one-out error would exceed the error for the model with all neurons.
- neurons – If this option is selected, DTREG reduces the least significant neurons until only the specified number of neurons remain.

CHAPTER-5

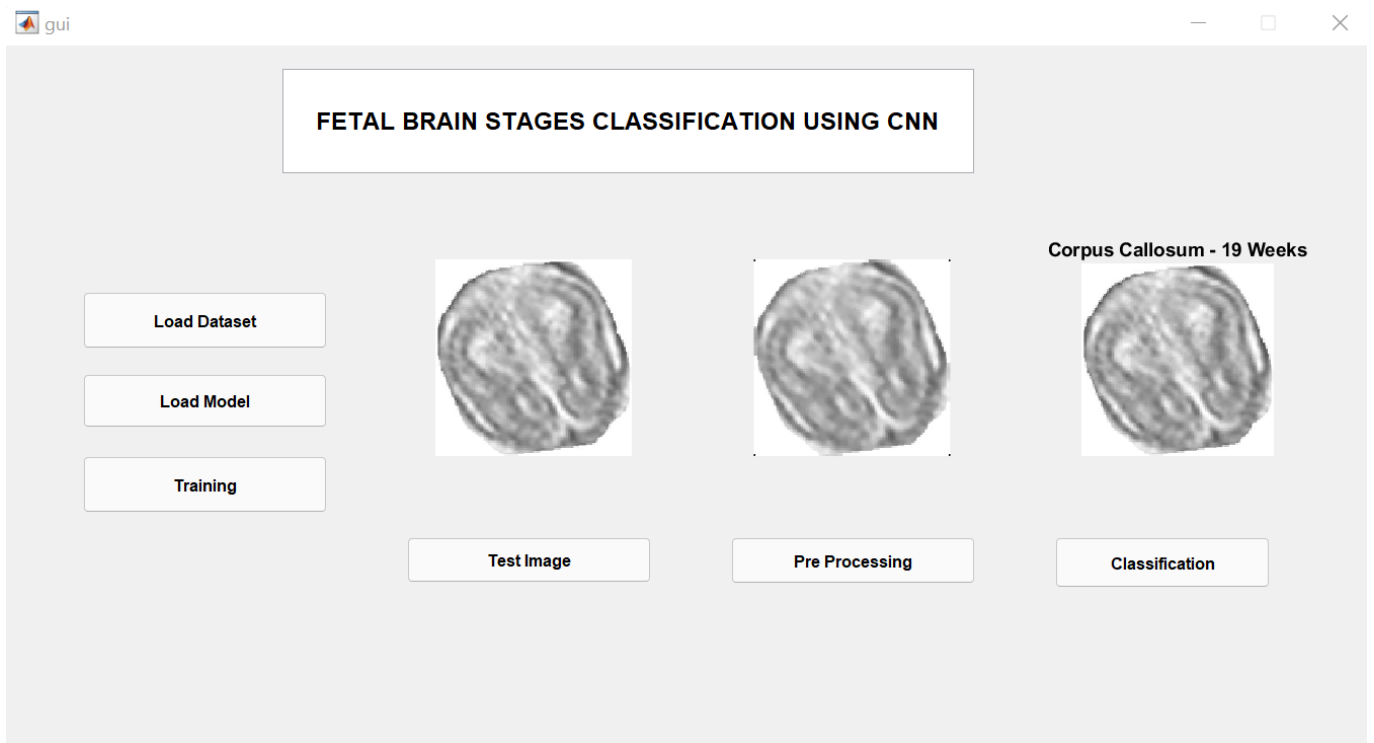
RESULT & STIMULATION

5.1 RESULT

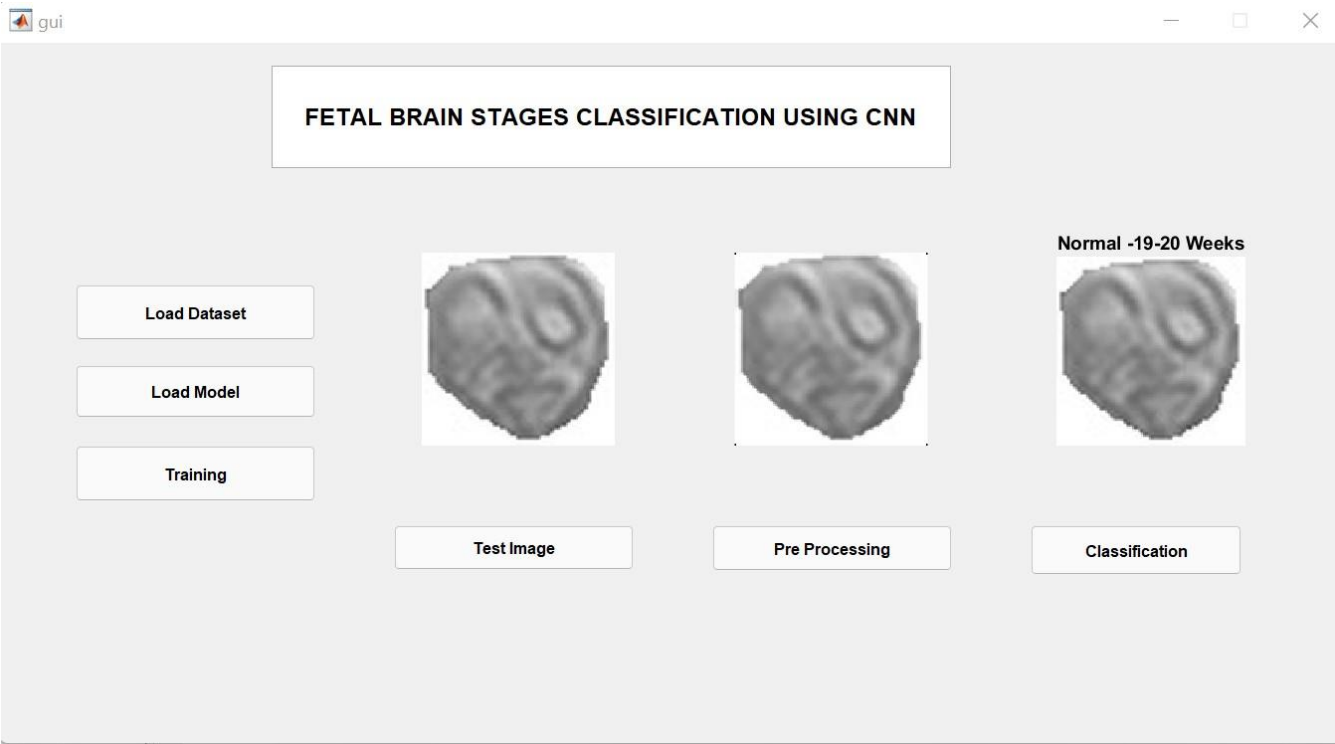
We run the proposed framework in MATLAB, and we get the simulation results in GUI (Graphical User Interface) of MATLAB. Fig 6.1 shows the full simulation result of proposed framework. Firstly, the raw MR image will be browsed then the step of the image will be done. In image, segmentation part will be done. It also presents segmentation phases of the image. After, DWT phase is there i.e. to compress the image Then the GLCM features will be extracted like entropy, contrast etc. After clicking DATABASE button, the test image will be checked with the training images then after classification button the result will come i.e., the fetal image is normal or abnormal.

5.2 STIMULATION:

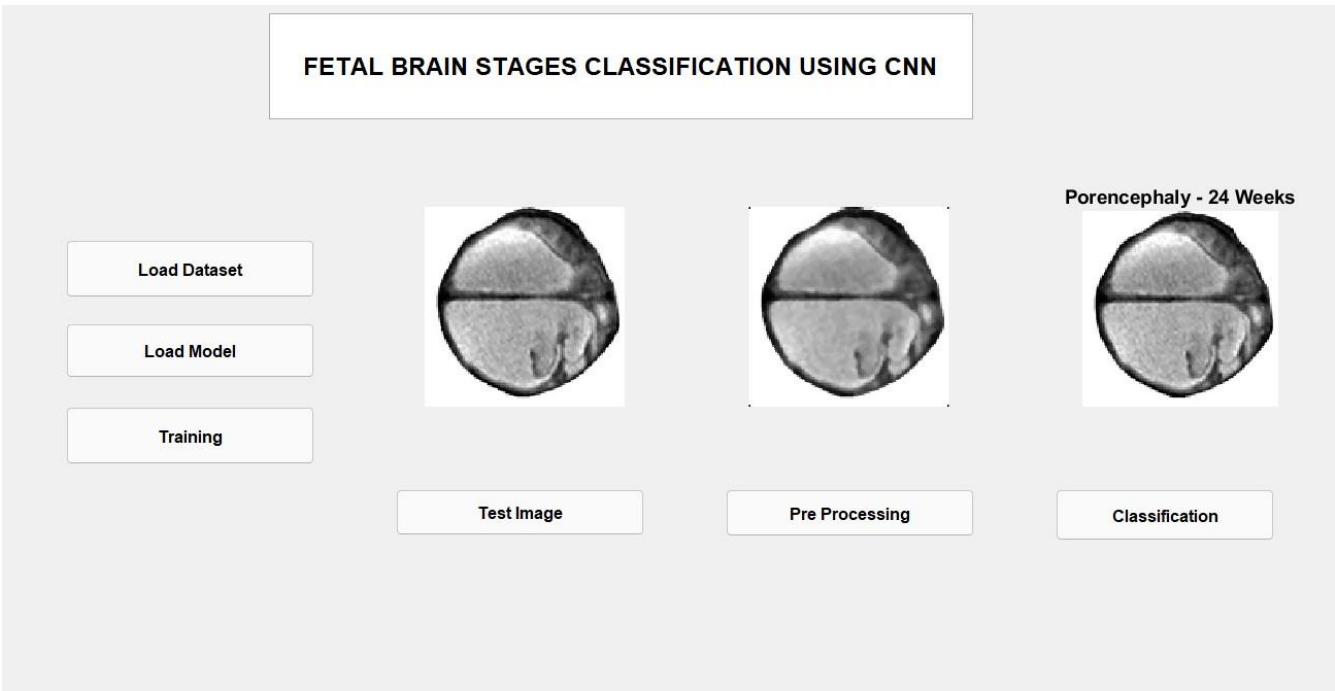
Eg:1



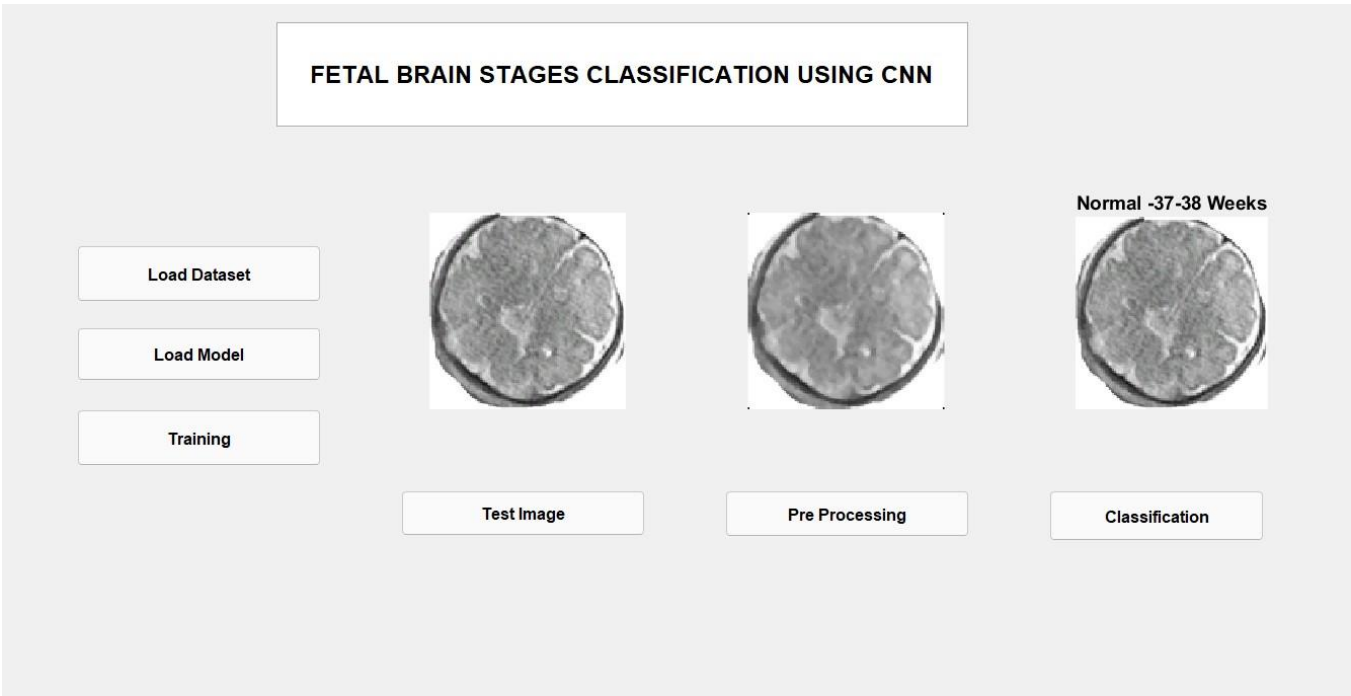
Eg:2



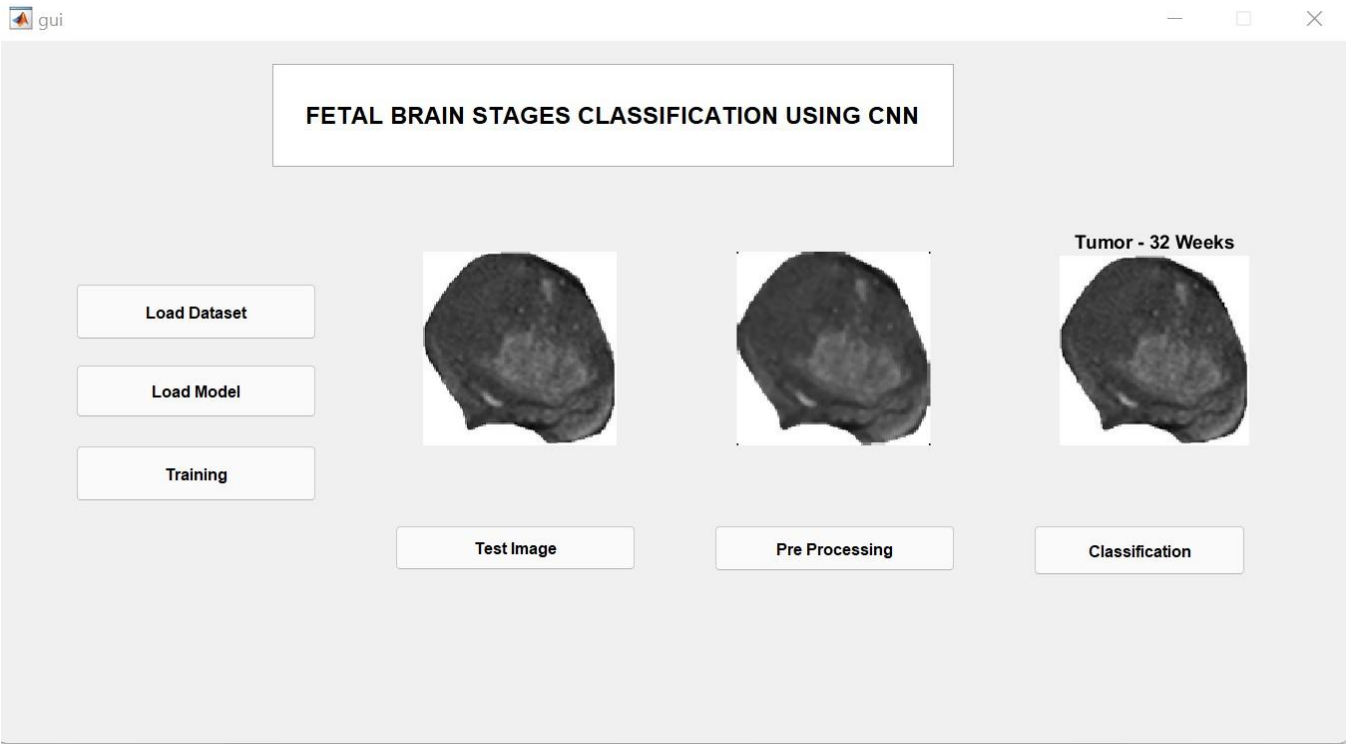
Eg:3



Eg:4



Eg:5



CHAPTER- 6

SUMMARY AND CONCLUSION

6.1 SUMMARY

- Today, Deep learning is a man-made consciousness work that mirrors the operations of the human brain in preparing information and making designs for use in dynamic operations and images. Deep Learning is a sub set of machine learning .
- It uses Neural Networks which process the visual image information at different layers or levels. The lower levels process small information and higher levels process complex information.
- As according to a research approximately 3 among every 1000 ladies are pregnant with abnormal fetal brain. Early discovery of fetal brain variations from the norm utilizing ML methods (deep learning) can improve the nature of determination and treatment arranging.
- We found that majority of the work done regarding detection of brain abnormalities was done after the child was born and not during the fetal stage.
- MRI is an important tool in for brain abnormality detection.
- We have proposed a method for the detection of any abnormality or disease in the brain when the child is in the fetal stage so that if any strange phenomenon is observed, it can be treated and operated accordingly and can be rectified at an early stage .
- We use the deep learning techniques and algorithms unlike other research work done using machine learning . These algorithms are fast and much more efficient than machine learning techniques

6.2 CONCLUSION

- We proposed a method for the recognition and classification of brain abnormalities in the fetus.
- We used deep learning techniques and algorithms for this purpose such as the CNN

algorithm. Unlike other researches that used machine learning techniques and algorithms for the same purpose, our proposed work has shown higher accuracy and efficient results.

- We achieved accuracy of around 92%-93% which is higher than any other previous work. Also, we were able to predict and identify different brain abnormalities with images of different gestational ages. The CNN classifier, we used here has outperformed any other algorithm in terms of decreasing computational cost and increasing image processing speed. We want to achieve more efficiency in the future work. And we will also increase the data sets. And we will classify different abnormal diseases like tumor, cerebellar hypoplasia Etc.,

REFERENCES

- 1) Attallah.O, Gadelkarim,H, Sharkas, M.A. Detecting and Classifying Fetal Brain Abnormalities Using Machine Learning Techniques. In Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 17–20 December 2018; pp. 1371–1376.
- 2) Ball.G, Aljabar.P, Arichi.T, Cox.D, Merchant.N, Nongena,P Hajnal,J.V, EdwardsA.D. Counsell, S.J. Machine-learning to characterize neonatal functional connectivity in the preterm brain. Neuroimage 2016, 124, 267–275.
- 3) Balaji.V, Kanaga.S Suba Raja.S, Aparna.J ,Haritha.M, Lakshmi Kanya (2019) 'Smart Assistance for Asperger Syndrome using Raspberry Pi',International Journal of Innovative Technology and Exploring Engineering, ISSN: 2278–3075, Volume 9, Issue - 1, pp. 2680-2683.
- 4) Carlos Arizmendi ; Alfredo Vellido ; Enrique Romero; Binary classification of brain tumors using a Discrete Wavelet Transform and energy criteria,2011
- 5) Levman,J, Takahashi.E., Multivariate analyses applied to fetal, neonatal and pediatric MRI of neurodevelopmental disorders. Neuroimage Clin. 2015, 9, 532–544.
- 6) Rajaram K., Usha Kiruthika S. (2010) Dynamic Contract Generation and Monitoring for B2B Applications with Composite Services. In: Das V.V., Vijaykumar R. (eds) Information and Communication Technologies. ICT 2010. Communications in Computer and Information Science, vol 101. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-15766-0_55
- 7) Raja, K. S., & Kiruthika, U. (2015). An energy efficient method for secure and reliable data transmission in wireless body area networks using RelAODV. Wireless Personal Communications, 83(4), 2975-2997
- 8) Shahla Najafi; Maehdi Chehel Amirani;Zahra Sedghi; A New Approach to MRI Brain Images Classification. 2011.

- 9)** Smyser, C.D.; Dosenbach, N.U.; Smyser, T.A.; Snyder, A.Z.; Rogers, C.E.; Inder, T.E.; Schlaggar, B.L.; Neil, J.J. Prediction of brain maturity in infants using machine-learning algorithms. *NeuroImage* 2016, 136, 1–9.
- 10)** Sanz-Cortés, M.; Ratta, G.A.; Figueras, F.; Bonet- Carne, E.; Padilla, N.; Arranz, A.; Bargalló, N.; Gratacós, E. Automatic Quantitative MRI Texture Analysis in Small-for-Gestational-Age Fetuses Discriminates Abnormal Neonatal Neurobehavior. *PLoS ONE* 2013, 8, e69595.
- 11)** Sanz-Cortes, M.; Figueras, F.; Bonet-Carne, E.; Padilla, N.; Tenorio, V.; Bargallo, N.; Amat- Roldan, I.; Gratacós, E. Fetal brain MRI texture analysis identifies different microstructural patterns in adequate and small for gestational age fetuses at term. *Fetal Diagn. Ther.* 2013, 33, 122–129

APPENDIXES

CODE FOR STIMULATION:

```
% 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 03-Jul-2020 14:56:25
```

```
% 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 nargin
```

```
    [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
handles.output = hObject;

a=ones([256 256]);
axes(handles.axes1);
imshow(a);

axes(handles.axes2);
imshow(a);

axes(handles.axes3);
imshow(a);

% Update handles structure
guidata(hObject, handles);

% 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\MAHA VENKATESH\Desktop\Fetal Brain Code\Fetal Brain Code\'
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 = googlenet;

```

```

layers = [ imageInputLayer([120 120 3])
net(2:end-3)
fullyConnectedLayer(20)
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 opt training train layers
opt = trainingOptions('adam','Maxepoch',20,'InitialLearnRate', 0.001)
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

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);
img=inp;
if size(inp,3)>1
    Freg = rgb2gray(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 = rgb2gray(inp);
J = medfilt2(img);
axes(handles.axes2);
title('Filtered Image');
imshow(J);
% Update handles structure
guidata(hObject, handles);
% --- Executes on button press in pushbutton4.
% --- 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);
```