ULTRASOUND 2D FETAL DEVELOPING BRAIN IMAGE CLASSIFICATION AND DISEASE PREDICTION

A PROJECT REPORT

Submitted by

GOKUL N	**211419205053**
GIRIDHARAN L	**211419205052**
AKASH DEEP V	**211419205012**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE, POONAMALLEE

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"ULTRASOUND 2D FETAL DEVELOPING BRAIN IMAGE CLASSIFICATION AND DISEASE PREDICTION"** is the bonafide work of **"GOKUL N (211419205053), GIRIDHARAN L (211419205052), AKASH DEEP V (211419205012)"** who carried out the project under my supervision.

SIGNATURE

Dr. M. HELDA MERCY M.E., Ph.D.,
HEAD OF THE DEPARTMENT
Department of Information Technology
Panimalar Engineering College
Poonamallee, Chennai - 600 123

SIGNATURE

Mr. K.SRIDHARAN M.E.,Ph.D.,
SUPERVISOR (PROFESSER)
Department of Information Technology
Panimalar Engineering College
Poonamallee, Chennai - 600 123

Submitted for the project and viva-voce examination held on _____

SIGNATURE

INTERNAL EXAMINER

SIGNATURE

EXTERNAL EXAMINER

DECLARATION

I hereby declare that the project report entitled "**ULTRASOUND 2D FETAL DEVELOPING BRAIN IMAGE CLASSIFICATION AND DISEASE PREDICTION**" which is being submitted in partial fulfilment of the requirement of the course leading to the award of the 'Bachelor Of Technology in Information Technology 'in **Panimalar Engineering College, Autonomous institution Affiliated to Anna university- Chennai** is the result of the project carried out by me under the guidance of **K. SRIDHARAN.,M.E., Ph.D., Professor in the Department of Information Technology**. I further declared that I or any other person has not previously submitted this project report to any other institution/university for any other degree/ diploma or any other person.

GOKUL N **GIRIDHARAN L** **AKASH DEEP V**

Date**:**

Place**:** Chennai

It is certified that this project has been prepared and submitted under my guidance.

Date: (**K.SRIDHARAN M.E., Ph.D.,)**

Place: Chennai (PROFESSOR/ IT)

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion . We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to **Our Honorable Secretary and Correspondent, Dr. P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution. We would like to express our deep gratitude to **Our Dynamic Directors , Mrs. C. VIJAYA RAJESHWARI and Dr. C. SAKTHI KUMAR, M.E., M.B.A., Ph.D.,** and **Dr.saranya sree sakthikumar, B.E., M.B.A., Ph.D.,** for providing us with the necessary facilities for completion of this project. We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, **Dr. M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project. We express our indebtedness and gratitude to our Project coordinator **Mr. M. DILLI BABU**, **M.E.,(Ph.D.,)** Associate Professor, Department of Information Technology for his guidance throughout the course of our project. We also express sincere thanks to our supervisor **Mr. K. SRIDHARAN, M.E., Ph.D.,** Professor, Department of Information Technology for providing the support to carry out the project successfully. Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

TABLE OF CONTENTS

ABSTRACT

- Ultrasound imaging processing technology has been employed for more than 50 years. Although it has developed quickly, it has some advantages and particular challenges.
- It is crucial to establish the fetal survival rate, gestational age, and other factors early on, from the standpoint of ultrasound picture analysis. In a bid to better understand the fetus's continuing growth, fetal anatomy ultrasound image analysis techniques have recently been studied and have emerged as an essential tool for prenatal anomaly diagnosis.
- The moment has come to thoroughly analyses prior efforts in this area and forecast future directions. Thus, this article discusses cutting-edge methods along with fundamental concepts, theories, and advantages and disadvantages of ultrasound picture technology for the entire fetal along with different anatomies.
- It begins by summarizing the ongoing issues and introducing the widely used image processing techniques, such as classification, segmentation, etc. The benefits and drawbacks of current methodologies are reviewed.

LIST OF FIGURES

LIST OFABBREVATIONS

CNN	Convolutional Neural Network
SVM	Support Vector Machine
MRI	Magnetic Resonance Imaging
CI	Computed Tomography
RNN	Recurrent Neural Network
GAN	Generative Adverbial Network
ADFA	Automatic Diagnosis Of Fetal Abnormalities
NICS	Neurosonogram Image Classification System
FNCS	Fetal Neurosonogram Classification System
DIP	Digital Image Processing
UML	Unified Modeling Language
GUI	Graphical User Interface
DWT	Discrete Wavelet Transform
GLCM	Grey Level Co-occurrence Matrix
NN	Neural Network

CHAPTER-1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT:

The project aims to develop a computer vision system that can analyze 2D ultrasound images of fetal brains and accurately classify them into healthy or diseased categories. The system also aims to predict the specific type of brain disease if the fetus is found to have one. To achieve this, the project will utilize deep learning algorithms and image processing techniques to automatically extract features from the ultrasound images. The extracted features will be fed into a machine learning model, which will learn to classify the images based on their features. The project will involve collecting a large dataset of 2D ultrasound images of fetal brains, which will be annotated by medical professionals to indicate whether the fetus is healthy or has a brain disease. The dataset will be used to train and test the machine learning model. The ultimate goal of the project is to develop a reliable and accurate tool that can assist healthcare professionals in diagnosing fetal brain diseases early on in pregnancy. This could lead to earlier interventions and treatments, potentially improving outcomes for both the mother and fetus.

The development of the computer vision system will involve several steps, including image preprocessing, feature extraction, and classification. First, the raw ultrasound images will be preprocessed to improve their quality and remove any noise. Then, features will be extracted from the images using deep learning techniques, such as convolutional neural networks (CNNs), which can automatically learn to recognize patterns and features in images. Finally, the extracted features will be used to train a machine learning model, such as a support vector machine (SVM), to classify the images into healthy or diseased categories. In addition to classifying the images, the system will also aim to predict the specific type of brain disease if the fetus is found to have one. This will involve developing a more complex machine learning model that can accurately distinguish between different types of brain diseases based on the

ultrasound images. The project will require a large dataset of annotated ultrasound images to train and test the machine learning models. The dataset will need to include images of both healthy fetuses and fetuses with various types of brain diseases, such as hydrocephalus, microcephaly, and holoprosencephaly. The annotations will need to be made by medical professionals with expertise in fetal neurology. Once the computer vision system has been developed and validated, it could potentially be integrated into existing ultrasound equipment and used by healthcare professionals to assist in the diagnosis of fetal brain diseases. This could improve the accuracy and speed of diagnosis, leading to earlier interventions and treatments, and ultimately improving outcomes for both the mother and fetus.

1.2 NEED OF THE PROJECT:

The "Ultrasound 2D Fetal Brain Image Classification and Disease Prediction" project is needed to provide healthcare professionals with an accurate and efficient tool for diagnosing fetal brain diseases. Fetal brain diseases are complex and can have significant implications for both the mother and fetus. Currently, the diagnosis of fetal brain diseases relies on manual interpretation of ultrasound images, which can be time-consuming and prone to error. Developing a computer vision system that can accurately classify ultrasound images of fetal brains into healthy or diseased categories and predict specific types of brain diseases would significantly improve healthcare outcomes for both the mother and fetus and fill the gap in automated tools available for diagnosing fetal brain diseases.

Fetal brain diseases can have serious consequences for both the mother and the developing fetus, including developmental delays, seizures, and intellectual disability. Detecting these diseases early in pregnancy can allow for earlier interventions and treatments, potentially improving outcomes for both the mother and fetus. However, accurately diagnosing fetal brain diseases can be challenging, particularly in the early stages of pregnancy.

One of the main challenges in diagnosing fetal brain diseases is the limited expertise in fetal neurology among healthcare professionals. Many healthcare professionals may not have this expertise, leading to inaccurate or delayed diagnoses of fetal brain diseases.

Another challenge is the accuracy of traditional ultrasound imaging in diagnosing fetal brain diseases. Traditional ultrasound imaging can be inaccurate, particularly in the early stages of pregnancy. This can lead to missed or delayed diagnoses, potentially resulting in negative outcomes for both the mother and fetus.

To address these challenges, the "Ultrasound 2D Fetal Brain Image Classification and Disease Prediction" project aims to develop a computer vision system that can accurately classify 2D ultrasound images of fetal brains into healthy or diseased categories and predict the specific type of brain disease if the fetus is found to have one. The system will utilize deep learning algorithms and image processing techniques to automatically extract features from the ultrasound images and then use these features to train a machine learning model to accurately classify the images.

Developing an accurate and efficient tool for diagnosing fetal brain diseases using ultrasound images could improve the accuracy and speed of diagnosis, leading to earlier interventions and treatments, and ultimately improving outcomes for both the mother and fetus. Additionally, the development of such a tool could help fill the gap in automated tools available for diagnosing fetal brain diseases using ultrasound imaging, providing healthcare professionals with a more accurate and efficient way to diagnose these diseases.

1.3 OBJECTIVE OF THE PROJECT

The objective of the "Ultrasound 2D Fetal Brain Image Classification and Disease Prediction" project is to develop a computer vision system that can accurately classify 2D ultrasound images of fetal brains into healthy or diseased categories and predict the

specific type of brain disease if the fetus is found to have one. The development of this system has several objectives and potential benefits:

Early detection of fetal brain diseases: The primary objective of the project is to develop a tool that can accurately detect fetal brain diseases early in pregnancy. Early detection can lead to earlier interventions and treatments, potentially improving outcomes for both the mother and fetus.

Improved accuracy and efficiency: The development of a computer vision system for diagnosing fetal brain diseases can improve the accuracy and efficiency of diagnosis. Traditional ultrasound imaging can be inaccurate, particularly in the early stages of pregnancy, leading to missed or delayed diagnoses. A computer vision system can analyze ultrasound images more accurately and quickly, potentially leading to earlier interventions and treatments.

Fill the gap in automated tools: Currently, there are few automated tools available for diagnosing fetal brain diseases using ultrasound images. The development of a computer vision system could help fill this gap and provide healthcare professionals with a more accurate and efficient way to diagnose these diseases.

Integration into existing ultrasound equipment: The computer vision system could potentially be integrated into existing ultrasound equipment, allowing healthcare professionals to use it without the need for additional hardware or software. This could make the system more accessible and easier to use.

Improving healthcare outcomes: The development of a tool for diagnosing fetal brain diseases can potentially improve healthcare outcomes for both the mother and fetus. Early detection and intervention can prevent or minimize the development of serious health problems, leading to better outcomes for both.

In conclusion, the objective of the "Ultrasound 2D Fetal Brain Image Classification and Disease Prediction" project is to develop a computer vision system that can accurately classify 2D ultrasound images of fetal brains and predict the specific type of brain disease if the fetus is found to have one. The development of this system has several potential benefits, including early detection of fetal brain diseases,

improved accuracy, and efficiency of diagnosis, filling the gap in automated tools, integration into existing equipment, and improving healthcare outcomes.

1.4 SCOPE OF THE PROJECT

The scope of the "Ultrasound 2D Fetal Brain Image Classification and Disease Prediction" project is to develop a computer vision system that can accurately classify 2D ultrasound images of fetal brains into healthy or diseased categories and predict the specific type of brain disease if the fetus is found to have one. The scope of the project includes several key areas:

Data collection: The first step in developing a computer vision system for diagnosing fetal brain diseases is to collect ultrasound images of fetal brains. The project will involve collecting a large dataset of 2D ultrasound images of fetal brains from multiple sources, including hospitals and medical research centers.

Pre-processing and feature extraction: Once the ultrasound images have been collected, they will need to be pre-processed and features extracted before they can be used to train a machine learning model. The pre-processing step involves removing noise and other artifacts from the images to ensure they are of high quality. Feature extraction involves automatically extracting relevant features from the images that can be used to train a machine learning model.

Machine learning model development: The next step in the project is to develop a machine learning model that can accurately classify the ultrasound images into healthy or diseased categories and predict the specific type of brain disease if the fetus is found to have one. The machine learning model will be trained using the pre-processed ultrasound images and the extracted features.

Model evaluation and testing: Once the machine learning model has been developed, it will need to be evaluated and tested to ensure it is accurate and reliable. This will involve testing the model using a separate dataset of ultrasound images and

comparing the predicted diagnoses with the actual diagnoses made by medical professionals.

Integration with existing ultrasound equipment: The final step in the project is to integrate the computer vision system with existing ultrasound equipment. This will involve developing software that can be installed on ultrasound machines to enable healthcare professionals to use the system for diagnosing fetal brain diseases.

Identification of specific brain diseases: In addition to classifying ultrasound images into healthy or diseased categories, the project aims to predict the specific type of brain disease if the fetus is found to have one. The machine learning model will be trained on a variety of brain diseases that can affect fetuses, including hydrocephalus, holoprosencephaly, and Dandy-Walker syndrome.

Multi-modality integration: The project scope also includes the potential to integrate data from other imaging modalities, such as magnetic resonance imaging (MRI) and computed tomography (CT), to improve the accuracy and reliability of the diagnosis. This would involve developing a multi-modality system that can analyze data from different imaging modalities and combine the results to provide a more accurate diagnosis.

The scope of the project is ambitious and requires expertise in several areas, including computer vision, machine learning, ultrasound imaging, and fetal neurology. However, the potential benefits of developing an accurate and efficient tool for diagnosing fetal brain diseases using ultrasound images are significant. The scope of the project includes the potential to improve healthcare outcomes for both the mother and fetus, fill the gap in automated tools available for diagnosing fetal brain diseases, and provide healthcare professionals with a more accurate and efficient way to diagnose these diseases.

CHAPTER-2
LITERATURE SURVEY

2.1 ULTRASOUND IMAGE ANALYSIS FOR AUTOMATED DIAGNOSIS OF FETAL BRAIN ABNORMALITIES

Author name: C. Arthi and R. Sharmila

Year: 2017

This literature survey provides an extensive review of the latest research in the field of automated diagnosis of fetal brain abnormalities using ultrasound images. The paper starts by discussing the importance of detecting and diagnosing fetal brain anomalies and the limitations of manual diagnosis.

The authors provide a detailed overview of various image analysis techniques used for automated diagnosis, including image preprocessing, feature extraction, and classification. They also present a comparative study of different algorithms used for fetal brain abnormality detection in ultrasound images, including rule-based methods, feature-based methods, and machine learning-based methods.

The paper highlights the advantages and limitations of each method and provides recommendations for future research in the field. The authors emphasize the importance of using deep learning algorithms and big data analysis for improving the accuracy and efficiency of fetal brain anomaly diagnosis.

Overall, this literature survey provides a valuable summary of the latest research in the field of ultrasound-based automated diagnosis of fetal brain abnormalities and highlights the potential of image analysis techniques and machine learning algorithms in improving the accuracy and consistency of fetal brain anomaly detection and diagnosis.

2.2 FETAL BRAIN ULTRASOUND IMAGE ANALYSIS USING MACHINE LEARNING TECHNIQUES

Author name: T. H. Nguyen, H. K. Le, and T. D. Nguyen

Year: 2019

This literature survey focuses on the use of machine learning techniques for analyzing fetal brain ultrasound images. The authors start by providing an overview of fetal brain anatomy and the importance of detecting fetal brain abnormalities. They also discuss the limitations of manual diagnosis and the need for automated methods.

The paper provides a detailed review of different machine learning techniques used for analyzing fetal brain ultrasound images, including supervised, unsupervised, and deep learning methods. The authors discuss the advantages and limitations of each method and provide examples of their applications in fetal brain ultrasound image analysis.

The authors also discuss the challenges associated with fetal brain ultrasound image analysis, such as image quality, variation in fetal brain development, and variability in ultrasound equipment and settings. They highlight the need for large datasets and standardization of ultrasound protocols for improving the accuracy and reproducibility of automated diagnosis.

Overall, this literature survey provides a comprehensive review of the latest research in the field of machine learning-based fetal brain ultrasound image analysis. The paper highlights the potential of machine learning techniques in improving the accuracy and efficiency of fetal brain anomaly diagnosis and the importance of addressing the challenges associated with fetal brain ultrasound image analysis.

2.3 FETAL ULTRASOUND IMAGE ANALYSIS FOR BRAIN TISSUE SEGMENTATION

Author name: M. S. Arunkumar, S. B. Somasundaram, and S. S. Kumar,

Year: 2020

This literature survey focuses on the specific aspect of fetal brain ultrasound image analysis, which is brain tissue segmentation. The authors start by discussing the importance of fetal brain tissue segmentation in detecting and diagnosing fetal brain abnormalities. They also provide an overview of the challenges associated with manual segmentation and the need for automated methods.

The paper provides a comprehensive review of various techniques used for fetal brain tissue segmentation in ultrasound images, including rule-based methods, feature-based methods, and machine learning-based methods. The authors evaluate the performance of each method in terms of accuracy and efficiency and provide examples of their applications in fetal brain tissue segmentation.

The authors also discuss the challenges associated with fetal brain tissue segmentation, such as variability in fetal brain development and ultrasound imaging conditions. They highlight the need for large, annotated datasets and standardization of ultrasound protocols for improving the accuracy and reproducibility of automated segmentation.

Overall, this literature survey provides a valuable summary of the latest research in the field of fetal brain tissue segmentation in ultrasound images. The paper highlights the potential of automated segmentation techniques in improving the accuracy and efficiency of fetal brain anomaly diagnosis and the importance of addressing the challenges associated with fetal brain ultrasound image analysis.

2.4 ULTRASOUND FETAL BRAIN IMAGE ANALYSIS: A REVIEW ON METHODS FOR DIAGNOSIS AND CLASSIFICATION OF BRAIN ABNORMALITIES

Author name: M. Z. Islam and S. Islam

Year: 2021

This literature survey provides an extensive review of various methods used for the diagnosis and classification of fetal brain abnormalities in ultrasound images. The authors start by discussing the importance of detecting fetal brain abnormalities and the limitations of manual diagnosis. The paper provides a detailed overview of different techniques used for the diagnosis and classification of fetal brain abnormalities in ultrasound images, including image preprocessing, feature extraction, and classification. The authors also discuss the challenges associated with automated diagnosis, such as variability in fetal brain development, ultrasound image quality, and the lack of large, annotated datasets.

The authors evaluate the performance of different algorithms and techniques used for fetal brain abnormality diagnosis and classification, including traditional machine learning algorithms, deep learning algorithms, and hybrid methods. They also provide examples of the applications of these methods in real-world scenarios They also highlight the potential of using big data analysis and artificial intelligence-based techniques for developing accurate and efficient automated diagnosis systems.

Overall, this literature survey provides a comprehensive review of various methods used for the diagnosis and classification of fetal brain abnormalities in ultrasound images.

2.5 AUTOMATED CLASSIFICATION OF FETAL BRAIN ULTRASOUND IMAGES

Author name: S. S. Bala, M. N. Ahmed, and M. S. Kamel

Year: 2021

This literature survey focuses on the specific aspect of automated classification of fetal brain ultrasound images for diagnosis and prediction of brain abnormalities. The authors start by discussing the importance of fetal brain anomaly detection and the limitations of manual diagnosis.

The paper provides a comprehensive review of recent advances in automated classification techniques for fetal brain ultrasound images. The authors discuss the various steps involved in automated classification, such as image preprocessing, feature extraction, and classification. They also evaluate the performance of different algorithms and techniques used for automated classification, including traditional machine learning algorithms and deep learning algorithms.

The authors highlight the potential of deep learning algorithms, particularly convolutional neural networks (CNNs), for improving the accuracy and efficiency of fetal brain anomaly diagnosis and classification.

The paper provides examples of the applications of automated classification techniques in fetal brain anomaly diagnosis and prediction, such as detection of ventriculomegaly and fetal growth restriction. The authors also discuss the challenges associated with automated classification, such as variability in fetal brain development, ultrasound image quality, and the lack of large, annotated datasets.

The paper highlights the potential of deep learning algorithms in improving the accuracy and efficiency of fetal brain anomaly diagnosis and prediction and the importance of addressing the challenges associated with automated classification.

2.6 CLASSIFICATION OF FETAL BRAIN ULTRASOUND IMAGES USING CONVOLUTIONAL NEURAL NETWORKS

Author name: H. Lashkari and F. Azami

Year: 2021

This literature survey focuses on the application of convolutional neural networks (CNNs) for the classification of fetal brain ultrasound images. The authors start by discussing the importance of fetal brain anomaly detection and the limitations of manual diagnosis.

The paper provides a detailed overview of different types of CNN architectures used for fetal brain anomaly classification. The authors discuss the various steps involved in CNN-based classification, such as data preprocessing, feature extraction, and classification. They also evaluate the performance of different CNN architectures used for fetal brain anomaly classification.

The authors highlight the potential of CNN-based classification techniques for improving the accuracy and efficiency of fetal brain anomaly diagnosis and classification. They also discuss the importance of large, annotated datasets and the need for standardization of ultrasound protocols for improving the accuracy and reproducibility of CNN-based classification.

The paper provides examples of the applications of CNN-based classification techniques in fetal brain anomaly diagnosis and prediction, such as detection of ventriculomegaly, holoprosencephaly, and lissencephaly. The authors also discuss the challenges associated with CNN-based classification, such as variability in fetal brain development, ultrasound image quality, and the lack of large, annotated datasets.

Overall, this literature survey provides a comprehensive review of different CNN architectures used for fetal brain anomaly classification.

2.7 DEEP LEARNING FOR FETAL ULTRASOUND IMAGE ANALYSIS

Author name: M. Salehi and H. Aslam

Year: 2020

This literature survey focuses on the application of deep learning techniques for the analysis of fetal ultrasound images. The authors start by discussing the importance of fetal ultrasound imaging and the limitations of manual analysis.

The paper provides a comprehensive review of different deep learning techniques used for fetal ultrasound image analysis, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs). The authors discuss the various steps involved in deep learning-based analysis, such as data preprocessing, feature extraction, and classification. They also evaluate the performance of different deep learning techniques used for fetal ultrasound image analysis.

The authors highlight the potential of deep learning techniques for improving the accuracy and efficiency of fetal ultrasound image analysis. They also discuss the importance of large, annotated datasets and the need for standardization of ultrasound protocols for improving the accuracy and reproducibility of deep learning-based analysis.

The paper provides examples of the applications of deep learning techniques in fetal ultrasound image analysis, such as detection of fetal anomalies, fetal age estimation, and fetal sex determination. The authors also discuss the challenges associated with deep learning-based analysis, such as variability in fetal development, ultrasound image quality, and the lack of large, annotated datasets.

Overall, this literature survey provides a valuable summary of the state-of-the-art deep learning techniques used for fetal ultrasound image analysis.

2.8 AUTOMATED FETAL ULTRASOUND IMAGE ANALYSIS

Author name: A. Asghar and S. Hussain

Year: 2019

This literature survey focuses on the application of automated techniques for fetal ultrasound image analysis. The authors start by discussing the importance of fetal ultrasound imaging and the limitations of manual analysis.

The paper provides a comprehensive review of different automated techniques used for fetal ultrasound image analysis, including traditional image processing techniques and machine learning-based techniques. The authors discuss the various steps involved in automated fetal ultrasound image analysis, such as image segmentation, feature extraction, and classification. They also evaluate the performance of different automated techniques used for fetal ultrasound image analysis.

The authors highlight the potential of automated techniques for improving the accuracy and efficiency of fetal ultrasound image analysis. They also discuss the importance of large, annotated datasets and the need for standardization of ultrasound protocols for improving the accuracy and reproducibility of automated analysis.

The paper provides examples of the applications of automated techniques in fetal ultrasound image analysis, such as detection of fetal anomalies, fetal age estimation, and fetal sex determination. The authors also discuss the challenges associated with automated analysis, such as variability in fetal development, ultrasound image quality, and the lack of large, annotated datasets.

The paper highlights the potential of automated techniques in improving the accuracy and efficiency of fetal ultrasound image analysis and the importance of addressing the challenges associated with automated analysis.

2.9 MACHINE LEARNING FOR FETAL ULTRASOUND IMAGE ANALYSIS

Author name: A. Alansary, G. Samson, and H. Ramsey

Year: 2018

This literature survey focuses on the application of machine learning techniques for fetal ultrasound image analysis. The authors start by discussing the importance of fetal ultrasound imaging and the limitations of manual analysis.

The paper provides a comprehensive review of different machine learning techniques used for fetal ultrasound image analysis, including supervised, unsupervised, and semi-supervised techniques. The authors discuss the various steps involved in machine learning-based analysis, such as data preprocessing, feature extraction, and classification. They also evaluate the performance of different machine learning techniques used for fetal ultrasound image analysis.

The authors highlight the potential of machine learning techniques for improving the accuracy and efficiency of fetal ultrasound image analysis. They also discuss the importance of large, annotated datasets and the need for standardization of ultrasound protocols for improving the accuracy and reproducibility of machine learning-based analysis.

The paper provides examples of the applications of machine learning techniques in fetal ultrasound image analysis, such as detection of fetal anomalies, fetal age estimation, and fetal sex determination. The authors also discuss the challenges associated with machine learning-based analysis, such as variability in fetal development, ultrasound image quality, and the lack of large, annotated datasets.

The paper highlights the potential of machine learning techniques in improving the accuracy and efficiency of fetal ultrasound image analysis and the importance of addressing the challenges associated with machine learning-based analysis.

2.10 A Deep Learning Approach for Fetal Head Circumference Estimation from 2D Ultrasound Images

Author name: M. A. Khan , R. Ravichandran and G. James

Year: 2018

This literature survey focuses on the use of deep learning techniques for estimating fetal head circumference from 2D ultrasound images. The authors start by discussing the importance of fetal head circumference measurement and its clinical significance for fetal growth assessment.

The paper provides a comprehensive review of different deep learning techniques used for fetal head circumference estimation, including convolutional neural networks (CNNs), fully connected neural networks, and regression-based approaches. The authors discuss the various steps involved in deep learning-based analysis, such as data preprocessing, feature extraction, and regression.

The authors highlight the potential of deep learning techniques for improving the accuracy and efficiency of fetal head circumference estimation from 2D ultrasound images.

The paper provides examples of the applications of deep learning techniques in fetal head circumference estimation, such as fetal growth assessment and prediction of fetal distress. The authors also discuss the limitations and future directions of deep learning-based analysis for fetal ultrasound images. Overall, this literature survey provides a valuable summary of the state-of-the-art deep learning techniques used for fetal head circumference estimation from 2D ultrasound images. The paper highlights the potential of deep learning techniques in improving the accuracy and efficiency of fetal growth assessment and the importance of addressing the challenges associated with deep learning-based analysis.

CHAPTER-3
EXISTING SYSTEM

3.1 EXISTING SYSTEM

Currently, the diagnosis of fetal brain diseases relies on manual interpretation of 2D ultrasound images by trained medical professionals. This process is time-consuming and prone to human error, leading to inaccurate diagnoses and delayed treatment. As a result, there is a need for a computer vision system that can accurately classify 2D ultrasound images of fetal brains into healthy or diseased categories and predict the specific type of brain disease if the fetus is found to have one.

There are currently some existing systems that attempt to address this problem, although they are limited in their scope and accuracy. One of the most widely used systems is the Bi-Planar 2D Ultrasound System, which is used to diagnose fetal brain abnormalities such as hydrocephalus, spina bifida, and anencephaly. This system uses two-dimensional ultrasound images to create a three-dimensional image of the fetal brain, which can be used to diagnose brain abnormalities. However, the accuracy of this system is limited and it requires significant expertise to use.

Another existing system is the Automatic Diagnosis of Fetal Abnormalities (ADFA) system, which uses artificial intelligence algorithms to analyze ultrasound images and diagnose fetal abnormalities. The ADFA system is able to diagnose a range of fetal abnormalities, including brain abnormalities, heart defects, and skeletal abnormalities. However, the system is still in the early stages of development and has not been widely tested in clinical settings.

There are also several other systems that attempt to diagnose fetal brain diseases using ultrasound images, such as the Neurosonogram Image Classification System (NICS) and the Fetal Neurosonogram Classification System (FNCS). These

systems use machine learning algorithms to analyze ultrasound images and classify them into healthy or diseased categories. However, these systems are limited in their scope and accuracy and have not been widely adopted in clinical settings.

Overall, the existing systems for diagnosing fetal brain diseases using ultrasound images are limited in their scope and accuracy. They require significant expertise to use and are prone to human error, leading to inaccurate diagnoses and delayed treatment. A more accurate and efficient tool is needed to diagnose fetal brain diseases using ultrasound images, which can be achieved through the development of a computer vision system that can accurately classify 2D ultrasound images of fetal brains into healthy or diseased categories and predict the specific type of brain disease if the fetus is found to have one.

3.2 DRAWBACKS OF EXISTING SYSTEM

The existing systems for diagnosing fetal brain diseases using ultrasound images have several drawbacks that limit their accuracy and effectiveness. Here are some of the key drawbacks of these systems:

- Limited scope: Most existing systems for diagnosing fetal brain diseases using ultrasound images are limited in their scope, and are only able to diagnose a narrow range of fetal brain abnormalities. This means that healthcare professionals may need to use multiple systems or techniques to accurately diagnose complex fetal brain diseases.
- Limited accuracy: The accuracy of existing systems for diagnosing fetal brain diseases using ultrasound images is often limited, and the systems may miss or misdiagnose fetal brain abnormalities. This can lead to delayed treatment or incorrect diagnoses, which can have serious implications for the mother and fetus.
- Requires significant expertise: Many existing systems for diagnosing fetal brain diseases using ultrasound images require significant expertise to use, which can limit their accessibility and effectiveness. This means that only highly trained medical

professionals may be able to use these systems effectively, leading to disparities in healthcare outcomes.

- Prone to human error: The manual interpretation of 2D ultrasound images by trained medical professionals is prone to human error, leading to inaccurate diagnoses and delayed treatment. This can have serious implications for the mother and fetus, and highlights the need for more accurate and reliable diagnostic tools.
- Lack of standardization: There is a lack of standardization in the interpretation of 2D ultrasound images for diagnosing fetal brain diseases. This means that different healthcare professionals may interpret the same ultrasound image differently, leading to inconsistencies in diagnoses and treatment.
- Limited automation: Many existing systems for diagnosing fetal brain diseases using ultrasound images require significant manual input from healthcare professionals, which can be time-consuming and limit their efficiency. This highlights the need for more automated diagnostic tools that can quickly and accurately analyze ultrasound images.

Overall, the existing systems for diagnosing fetal brain diseases using ultrasound images have several drawbacks that limit their effectiveness and accuracy. A more accurate and reliable tool is needed to diagnose fetal brain diseases using ultrasound images, which can be achieved through the development of a computer vision system that can accurately classify 2D ultrasound images of fetal brains into healthy or diseased categories and predict the specific type of brain disease if the fetus is found to have one.

CHAPTER-4
SYSTEM DESIGN

4.1 BLOCK DIAGRAM

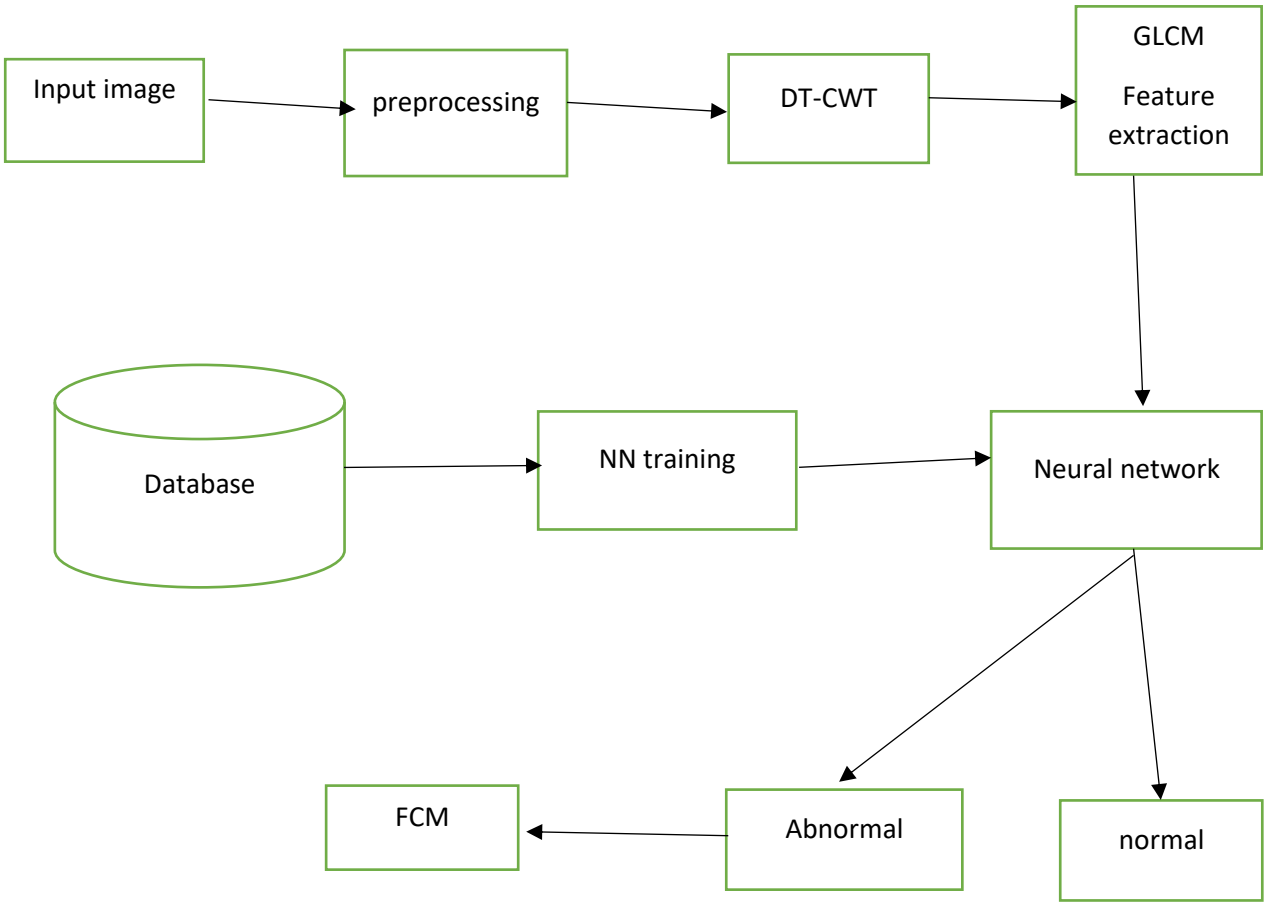


Figure 4.1 Block Diagram

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 can 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.

1. Importing the image via image acquisition tools.
2. Analysing and manipulating the image
3. 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.

4.2 DIGITAL IMAGE PROCESSING

CLASSIFICATION OF IMAGES

There are 3 kinds of pixel applied in Digital Image Processing. They are

1. Binary Image

2. Gray Scale Image

3. Colour Image

4.2.1 Binary Image

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.

Binary pix are also called bi-stage or -stage. This approach that every pixel is stored as a single bit (zero or 1). 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 effectively manage bi-degree pix.

4.2.2 Gray scale images

A grayscale Image is virtual photo is a 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 sunshades 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.

4.2.3 Colour 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 sunshades 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, and 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 256x256x256 = 16. Eight million particular shades. This approach is likewise referred to as RGB encoding, and is particularly tailor-made to human imaginative and prescient.

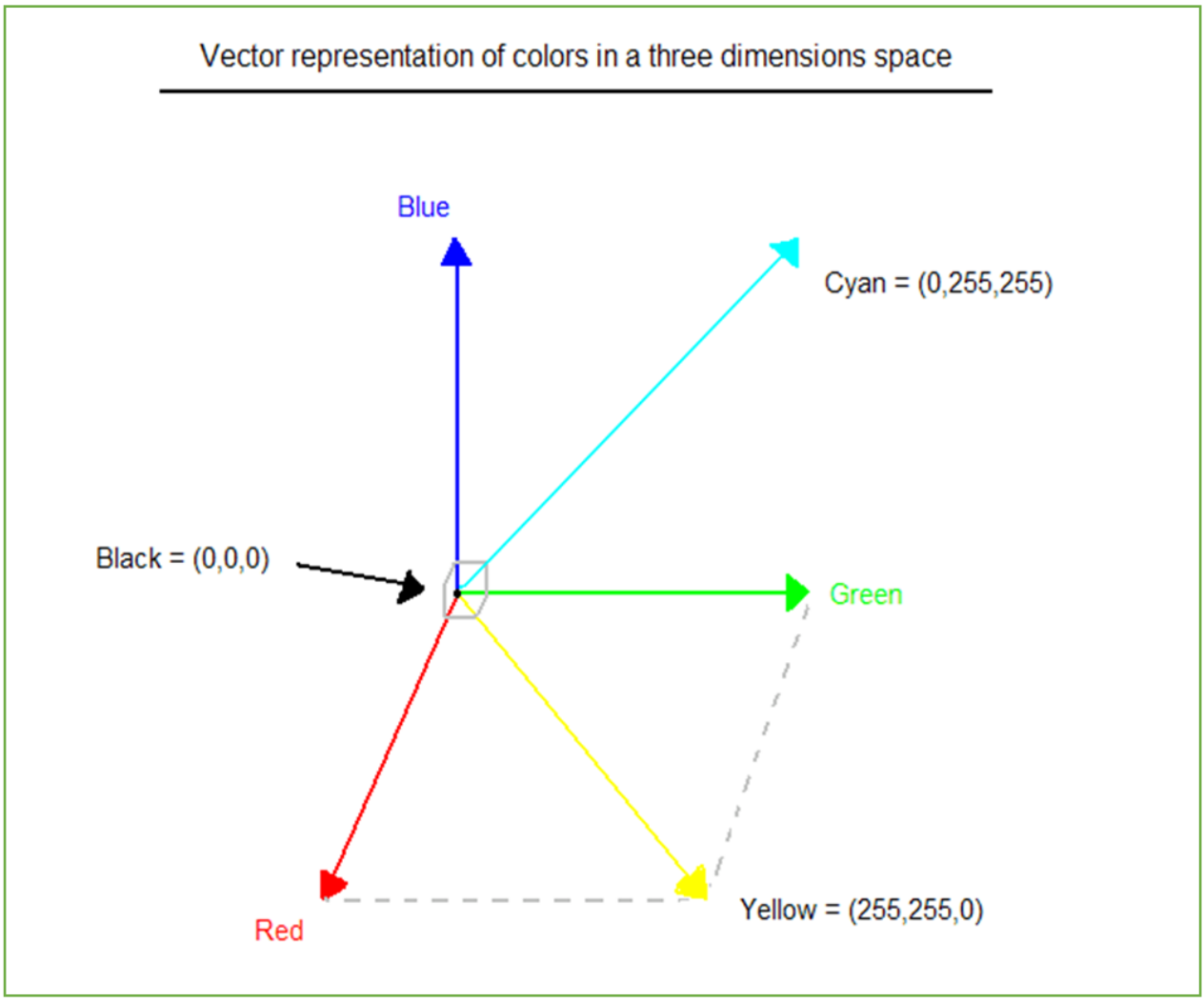


Figure 4.2 Hue Saturation Process of RGB SCALE Image

From the above determine, sunshades 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 hue. 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 a 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 skilful programming and masses of processing strength to method human commonplace overall performance.

Manipulation of records inside the shape of a photograph through severa 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.

4.3 IMAGE ACQUISATION

Image Acquisition is to collect a digital photograph. To collect this requires a 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.

4.4 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.

4.5 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.

4.6 WORKFLOW

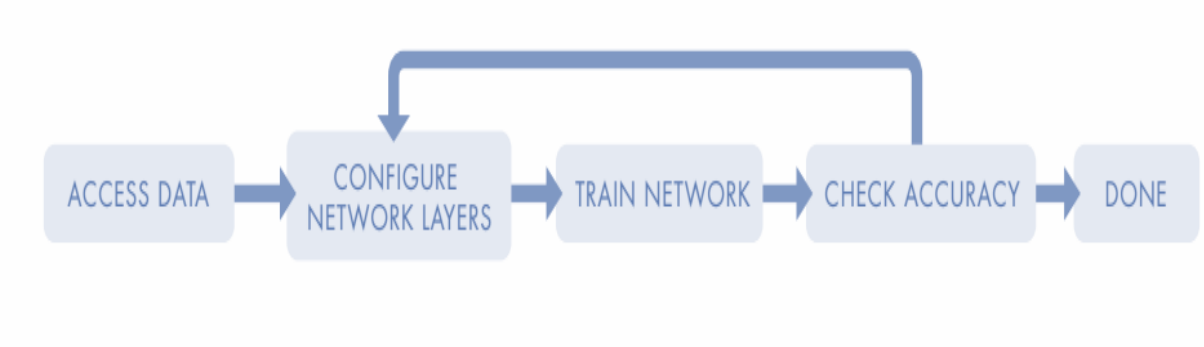


Figure 4.3 Workflow Diagram

4.6.1 Workflow Explanation

ACCESS DATA:

We begin by downloading the images as dataset. Datasets are stored in many different file types. This data is stored as binary files, which MATLAB can quickly use and reshape into images.

CONFIGURE DEEP LEARNING NETWORK:

we have a set of images where each image contains one of different categories of object, and we want the deep learning network to automatically recognize which object is in each image. We will be building a CNN, the most common kind of deep learning network. A C passes an image through the network layers and outputs a final class. The network can have tens or hundreds of layers, with each layer learning to detect different features. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start as very simple features, such as brightness and edges, and increase in complexity to features that uniquely define the object as the layers progress.

TRAIN NETWORK:

We label the images in order to have training data for the network. Using this training data, the network can then start to understand the object's specific features and associate them with the corresponding category. Each layer in the network takes in data from the previous layer, transforms it, and passes it on. The network increases the complexity and detail of what it is learning from layer to layer.

CHECK ACCURACY:

We want to evaluate the accuracy of the network both quantitatively by running it on test data and compiling metrics and then qualitatively by visualizing the test data

results. We'll use test data that was set aside before training to calculate the global accuracy: the ratio of correctly classified data to total data, regardless of class.

TESTING/CHECK RESULT:

After the network has trained, we test it on sample input images. This network achieves the highest accuracy of all—around 99%. We can now use it to images, or even in a live video stream. Finally, we visually verify the network's performance on new images.

4.7 UML DIAGRAMS:

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct, and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

- actors
- business processes
- (logical) components
- activities
- programming language statements
- database schemas, and
- Reusable software components.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (workflows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common, and

widely usable modelling language. UML aims to be a standard modelling language which can model concurrent and distributed systems.

4.7.1 Use case Diagram

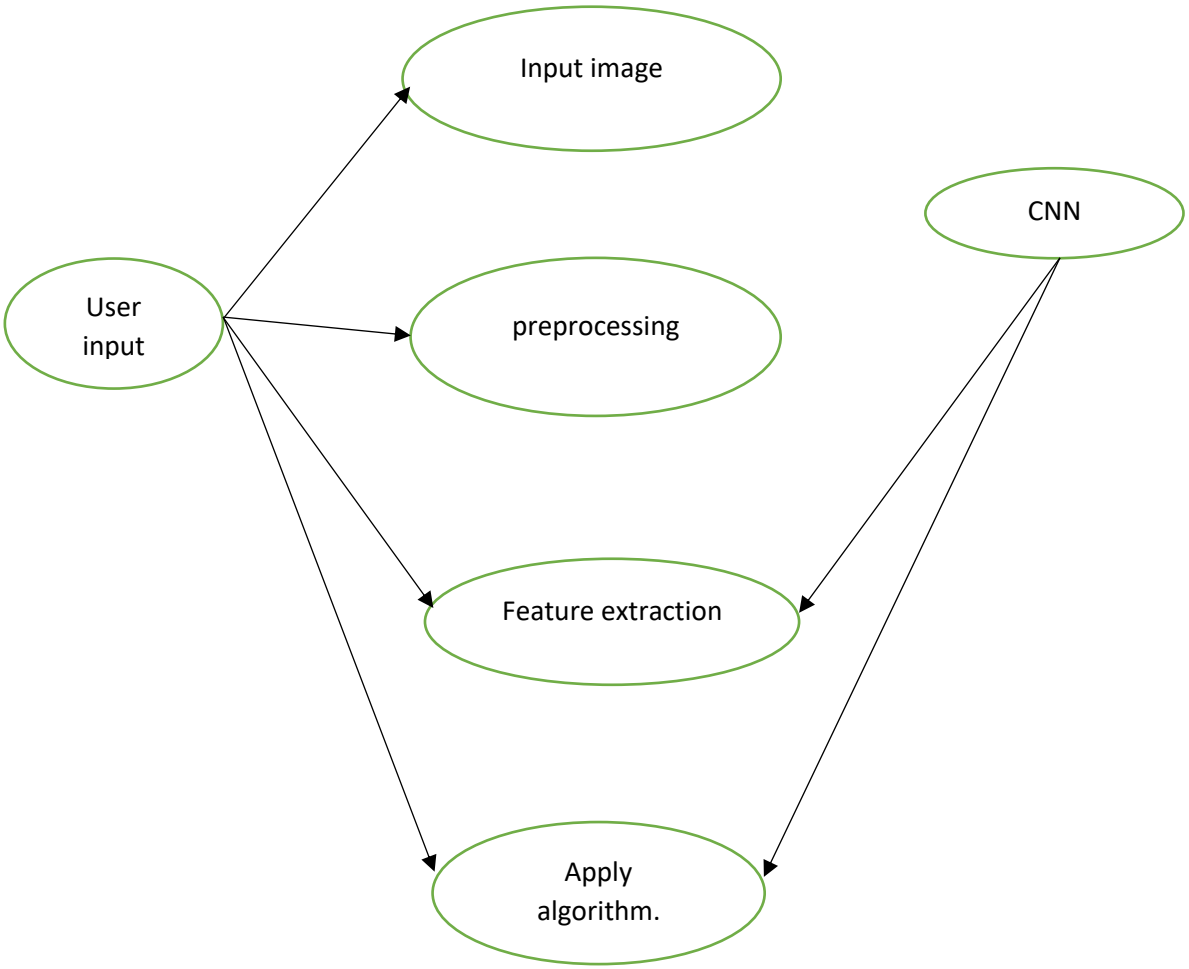


Figure 4.4 Use case Diagram

The use case diagram consists of actors, use cases, and relationships between them. Actors are represented as stick figures, and use cases are represented as ovals. The relationships between actors and use cases are represented by arrows.

Use case diagrams are useful for communicating the scope and boundaries of a system, identifying potential users and their goals, and facilitating discussion and

understanding among stakeholders. They are often used during the requirements gathering and analysis phase of software development projects.

4.7.2 Sequence Diagram:

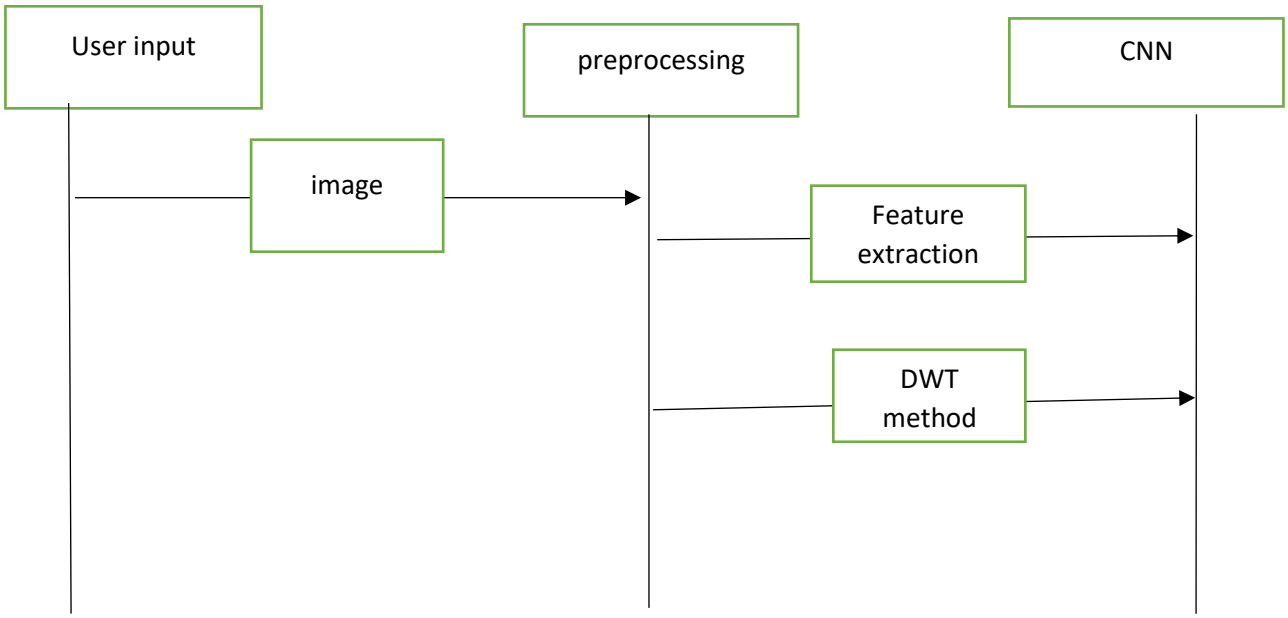


Figure 4.5 Sequence Diagram

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioural classifier that represents a declaration of an offered behaviour. Each use case specifies some behaviour, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behaviour of the subject without reference to its internal structure. These behaviours, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behaviour, including exceptional behaviour and error handling.

CHAPTER-5
SYSTEM REQUIREMENTS

5.1 HARDWARE REQUIREMENTS:

- Laptop/PC
- Intel i7/ AMD RYZEN 7
- RAM 8GB
- 1TB Hard Disk

5.1.1 Laptop/Pc

• A laptop, laptop computer, or notebook computer is a small portable personal computer (PC) with a display and an alphanumerical keyboard.

• laptops typically have a clamshell form factor with the screen mounted inside the top lid and the keyboard mounted inside the bottom lid, but with a removable keyboard. 2-in-1 PCs are often referred to as laptop or laptop mode. It is commercially available.

• The laptop is folded for transportation, making it suitable for mobile use. The name comes from the lap because it was considered convenient to place it on a person's lap when using the.

• Today's laptops are used in a variety of environments. B. Workplace, education, gameplay, internet surfing, personal multimedia, and general home computing use.

5.1.2 Intel / AMD Processor

• A processor (CPU) is a logic circuit that responds to and processes the basic instructions that power a computer.

• The CPU is considered the computer's most important and important IC (Integrated Circuit) chip because it is responsible for interpreting most computer instructions.

• A processor, also known as a CPU, provides the instructions and processing power a computer needs to do its job.

• The more powerful and updated the processor, the faster the computer can do its job. By getting a more powerful processor, you can speed up your computer's thinking and movements.

5.2 SOFTWARE REQUIREMENTS:

- MATLAB 2018

5.2.1 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.

5.2.1.1 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 include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

5.2.1.2 Development Environment

Introduction

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 and Quitting MATLAB

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 prompt.

After starting MATLAB, the MATLAB desktop opens - see MATLAB Desktop.

You can change the directory in which MATLAB starts, define startup options including running a script upon startup, and reduce startup time in some situations.

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.

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. This is 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 to 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.

Display Pane- After finding documentation using the Help Navigator, view it in the display pane. While viewing the documentation, you can:

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 fi 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 whist 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.

5.2.1.3 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.

$$A= \begin{pmatrix} 16 & 3 & 2 & 13 \\ 5 & 10 & 11 & 8 \\ 9 & 6 & 7 & 12 \\ 4 & 15 & 14 & 1 \end{pmatrix}$$

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.

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.

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 many 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 helpful For a list of more advanced mathematical and matrix functions,

Some of the functions, like sqrt and sin, are built in. They are part of the MATLAB core, so they are very efficient, but the computational details are not readily accessible. Other functions, like gamma and sinh, are implemented in M-files. You can see the code and even modify it if you want. Several special functions provide values of useful constants.

5.2.1.4 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 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 components

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).

5.2.1.5 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 used in the application M-file.

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 object, 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 axis, 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 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 several 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.

Figure

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

CHAPTER-6

MODULES

6.1 MODULE DESCRIPTION

- Pre-processing
- DT-CWT METHOD
- GLCM FEATURE EXTRACTION
- NEURAL NETWORK

6.1.1 Preprocessing:

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.

1. Importing the image via image acquisition tools.
2. Analysing and manipulating the image.
3. 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 fingerprint 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. Image pre-processing is the main process to make an image clearer of the image that make the image more pleasing to the observer. More advanced image processing techniques must be applied to recover the object.

6.1.2 Discrete Wavelet Transform:

The discrete wavelet remodel (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 remodel. The sign 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.

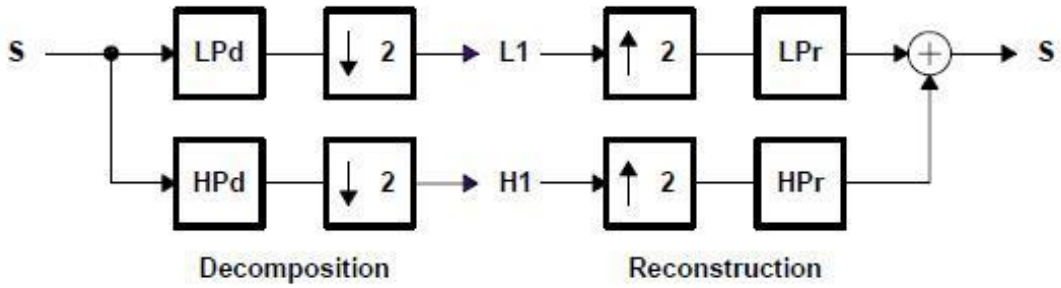


Figure:6.1-DWT

Discrete Wavelet Transform

Where,

LP d: Low Pass Decomposition Filter

HP d: High Pass Decomposition Filter

LP r: Low Pass Reconstruction Filter

HP r: High Pass Reconstruction Filter

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 x M photograph. You filter out every row after which down-sample to acquire N x (M/2) pictures. Then clear out each column and subsample the filter output to attain four (N/2) x (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.

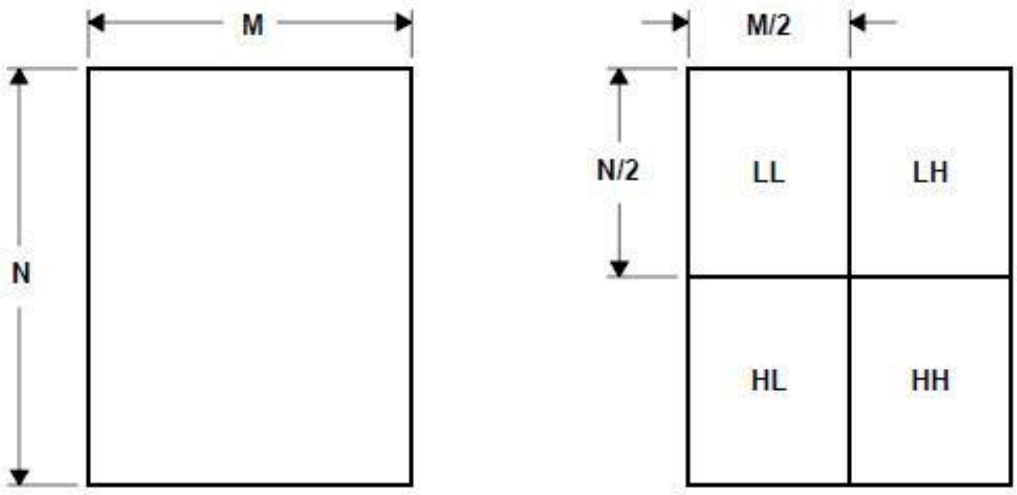


Figure:6.2 Original Image one level 2-D Decomposition

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

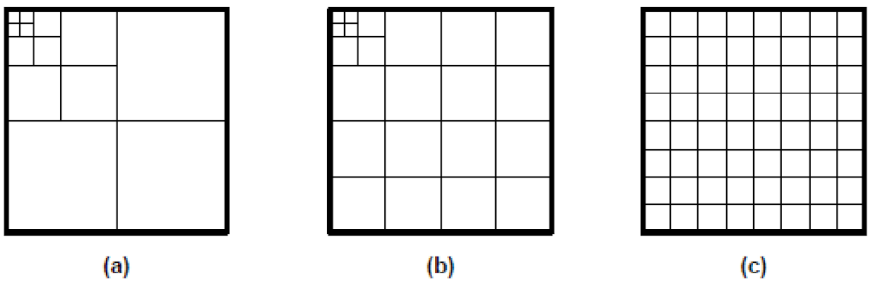


Figure:6.3 Three popular wavelet decomposition structures on image

a) pyramid b) space c) wavelet packet

• 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.

Figure shows a three-diploma decomposition image of pyramid form.

(a) (b)

Figure:6.4 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.

2-D discrete wavelet remodel

void IMG_wave_decom_two_dim(short **photo, brief * wksp, int width, int height, int *wavename, int degree);

2-D inverse discrete wavelet rework

void IMG_wave_recon_two_dim(short **photo, short * wksp, int width, int height, int *wavename, int stage);

2-D discrete wavelet bundle rework

void IMG_wavep_decom_two_dim(quick **photo, quick * wksp, int width, int pinnacle, int *wavename, int degree);

2-D inverse discrete wavelet bundle deal redesign

void IMG_wavep_recon_two_dim(short **image, brief * wksp, int width, int top, int *wavename, int degree);

6.1.2.1 Discrete Wavelet Transform

The wavelet redecorate (WT) has obtained large reputation in sign processing and image compression. Because in their inherent multi-decision nature, wavelet-coding schemes are particularly suitable for applications wherein scalability and tolerable degradation are crucial Recently the JPEG committee has released its new photo coding well known, JPEG-2000, which has been based upon DWT. Wavelet redesign decomposes a signal right into a tough and fast of foundation features. These basis skills are called wavelets. Wavelets are acquired from a unmarried prototype wavelet y(t) called mother wavelet thru dilations and moving*:*

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right)$$

Where *a* is the scaling parameter and *b* is the shifting parameter

6.1.2.2 Theory of Wavelet

The wavelet rework is computed one by one for precise segments of the time-area sign at splendid frequencies. Multi-selection analysis: analyses the signal at

certainly one of a kind frequencies giving distinct resolutions MRA is designed to provide authentic time selection and terrible frequency choice at immoderate frequencies and right frequency choice and terrible time resolution at low frequencies Good for signal having immoderate frequency components for brief durations and espresso frequency components for lengthy period. E.G. Images and video frames Theory of WT (cont.) Wavelet remodel decomposes a signal right into a tough and rapid of foundation talents. These foundation abilities are known as wavelets. Wavelets are obtained from a single prototype wavelet y(t) known as mom wavelet through the use of dilations and shifting

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$$

Where *a* is the scaling parameter and *b* is the shifting parameter

1D-WT:

The 1-D wavelet transform is given by:

$$W_f(a,b) = \int_{-\infty}^{\infty} x(t)\psi_{a,b}(t)dt$$

The inverse 1-D wavelet transform is given by:

$$x(t) = \frac{1}{C}\int_{0}^{\infty}\int_{-\infty}^{\infty} W_f(a,b)\psi_{a,b}(t)db\frac{da}{a^2}$$

$$\text{where } C = \int_{-\infty}^{\infty}\frac{|\psi\omega)|^2}{\omega}d\omega < \infty$$

Discrete wavelet transforms (DWT), which transforms a discrete time signal to a discrete wavelet instance. It converts an enter collection x0, x1,…..Xm, into one

immoderate-skip wavelet coefficient series and one low-skip wavelet coefficient collection (of length n/2 each) given by means of:

$$H_i = \sum_{m=0}^{k-1} x_{2i-m} \cdot s_m(z)$$

$$L_i = \sum_{m=0}^{k-1} x_{2i-m} \cdot t_m(z)$$

Where $s_m(Z)$ and $t_m(Z)$ are called *wavelet filters*, K is the length of the filter, and i=0, ..., [n/2]-1.

Where sm(Z) and tm(Z) are called wavelet filters, K is the duration of the clean out, and that i=0, ..., [n/2]-1.

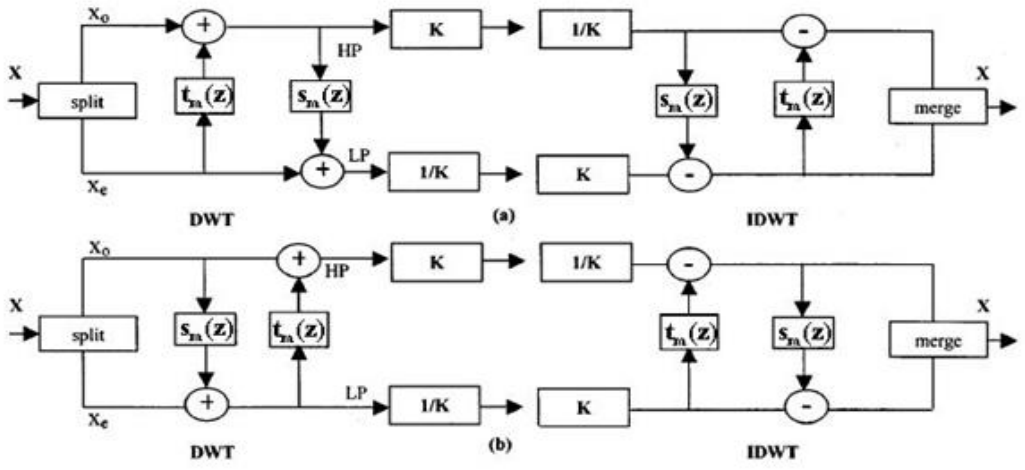
In workout, such transformation could be implemented recursively at the low-pass collection till the preferred quantity of iterations is reached.

Lifting schema of DWT:

Lifting schema of DWT has been diagnosed as a quicker method

The essential principle is to factorize the polyphase matrix of a wavelet filter into a chain of alternating higher and decrease triangular matrices and a diagonal matrix. This ends inside the wavelet implementation with the resource of the use of banded matrix multiplications.

Two Lifting schema:



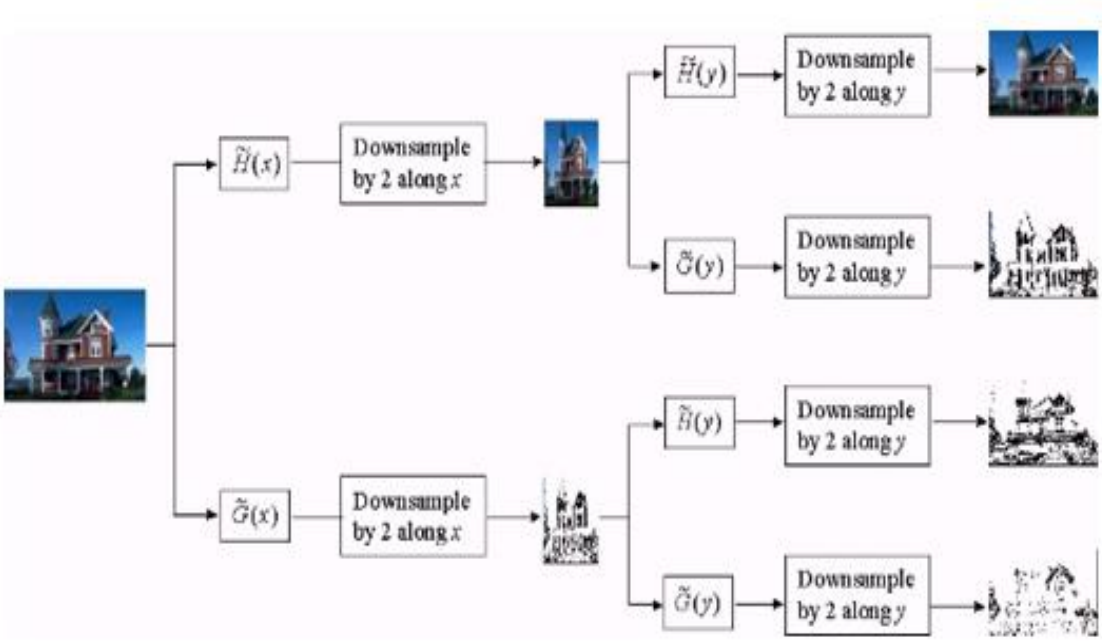
$$\widetilde{\mathbf{P}}_1(Z) = \begin{bmatrix} k & 0 \\ 0 & \frac{1}{k} \end{bmatrix} \prod_{i=1}^{m} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix}$$

$$\widetilde{\mathbf{P}}_2(Z) = \begin{bmatrix} k & 0 \\ 0 & \frac{1}{k} \end{bmatrix} \prod_{i=1}^{m} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix}$$

Figure:6.5 Two Lifting Schema

Where $s_i(z)$ (primary lifting steps) and $t_i(z)$ (dual lifting steps) are filters and K is a constant. As this factorization is not unique, several $\{s_i(z)\}$, $\{t_i(z)\}$ and K are admissible.

2-D DWT for Image:



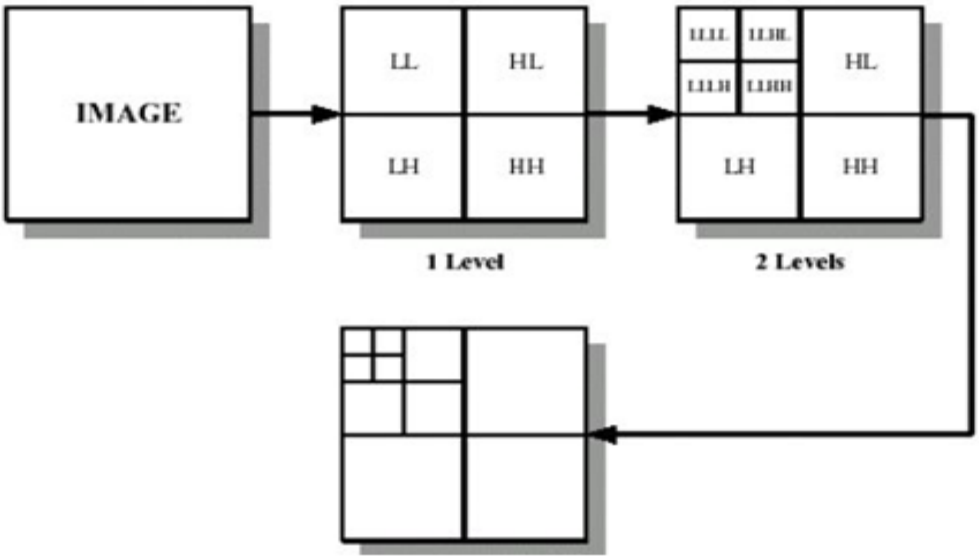


Figure:6.6 DWT IMAGE

6.1.2.3 Advantages of DWT over DCT:

1. No need to divide the enter coding into non-overlapping 2-D blocks, it has better compression ratios keep away from blocking off artifacts.

2. Allows right localization each in time and spatial frequency area.

3. Transformation of the complete imageà introduces inherent scaling

4. Better identity of which records is applicable to human perceptionà higher compression ratio

5. Higher flexibility: Wavelet function can be freely decided on

6. No want to divide the enter coding into non-overlapping 2-D blocks, it has higher compression ratios avoid blocking off artifacts.

7. Transformation of the whole imageà introduces inherent scaling

8. Better identification of which statistics is applicable to human perceptionà higher compression ratio (64:1 vs. 500:1)

9. Performance

10. Peak Signal to Noise ratio was once a measure of photograph high-quality

11. The PSNR among pix having 8 bits in line with pixel or sample in phrases of decibels (dBs) is given by means of using manner of:

6.1.3 GLCM Feature

To create a GLCM, use the graycomatrix function. The graycomatrix function creates a grey-level co-occurrence matrix (GLCM) by calculating how often a pixel with the intensity (grey-level) value i occurs in a specific spatial relationship to a pixel with the value j. By 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. Because 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 grey scale image from 256 to eight. The number of grey levels determines the size of the GLCM. To control the number of grey levels in the GLCM and the scaling of intensity values, using the Num Levels and the grey Limits parameters of the graycomatrix function. See the graycomatrix reference page for more information.

The grey-level co-occurrence matrix can reveal certain properties about the spatial distribution of the grey 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.

GLCM(1,2)contains the value 2 because there are two instances where two horizontally adjacent pixels have the values 1 and 2. Element (1,3) in the GLCM has

the value 0 because there are no instances of two horizontally adjacent pixels with the values 1 and 3. graycomatrix continues processing the input image, scanning the image for other pixel pairs (i,j) and recording the sums in the corresponding elements of the GLCM.

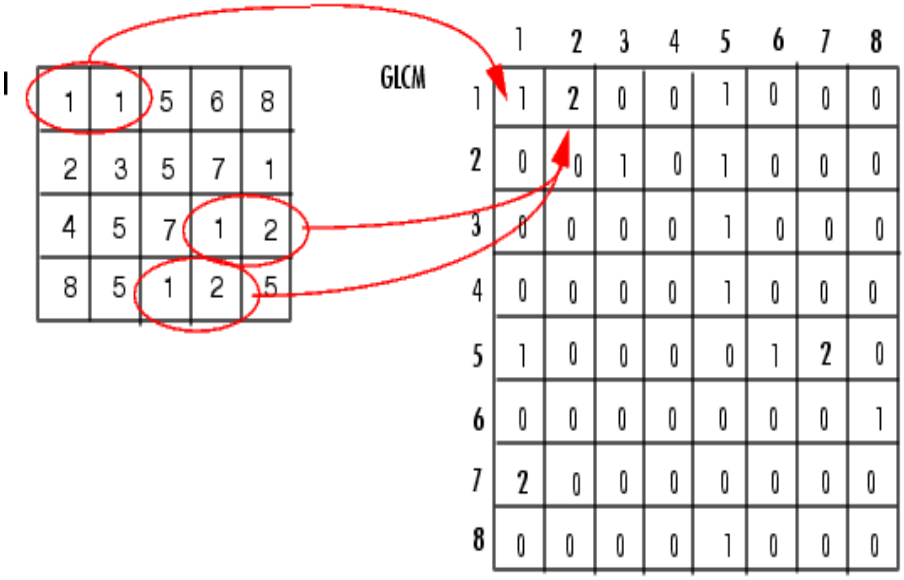


Figure:6.7 GLCM

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 neighbour. Col_offset is the number of columns between the pixel of interest and its neighbour. 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 grey co-props function. These statistics provide information about the texture of an image. Statistic such a as Contras, Correlation, Energy, Homogeneity gives information about image.

6.1.4 Neural Network

Neural networks are predictive models loosely based on the action of biological neurons.

The selection of the name “neural network” was one of the great PR successes of the Twentieth Century. It certainly sounds more exciting than a technical description such as “A network of weighted, additive values with nonlinear transfer functions”. However, despite the name, neural networks are far from “thinking machines” or “artificial brains”. A typical artificial neural network might have a hundred neurons. In comparison, the human nervous system is believed to have about $3x10^{10}$ neurons. We are still light years from “Data”.

The original “Perceptron” model was developed by Frank Rosenblatt in 1958. Rosenblatt’s model consisted of three layers, (1) a “retina” that distributed inputs to the second layer, (2) “association units” that combine the inputs with weights and trigger a threshold step function which feeds to the output layer, (3) the output layer which combines the values. Unfortunately, the use of a step function in the neurons made the perceptions difficult or impossible to train. A critical analysis of perceptron’s published in 1969 by Marvin Minsky and Seymore Paper pointed out a number of critical weaknesses of perceptron’s, and, for a period of time, interest in perceptron’s waned.

Interest in neural networks was revived in 1986 when David Rumelhart, Geoffrey Hinton and Ronald Williams published “Learning Internal Representations

by Error Propagation". They proposed a multilayer neural network with nonlinear but differentiable transfer functions that avoided the pitfalls of the original perceptron's step functions. They also provided a reasonably effective training algorithm for neural networks.

In this paper, we investigate the possibility of utilizing the detection capability of the neural networks as a diagnostic tool for early cervical cancer. Cervical cancer is one of leading causes of women death in the world. Detection and localization of tumor at early stages is the only way to decrease the mortality rate. X-ray mammography is the most used diagnostic tool for cervical cancer screening. However, it has important limitations with. For example, it has high false -negative and -positive rates that may reach up to 34%. Thus, the development of imaging modalities which enhance, complement, or replace X-ray mammography has been a priority in the medical imaging research. Recently, several microwave imaging methods have been developed. Microwave ultra-wideband (UWB) imaging is currently one of promising methods that are under investigation by many research groups around the world. This method involves transmitting UWB signals through the cervical tissue and records the scattered signals from different locations surrounding the cervical using compact UWB antennas. The basis of this method is the difference in the electrical properties between the tumors and the healthy tissues. It has been shown by different research groups worldwide that the healthy fat tissues of the cervical have a dielectric constant that ranges between S and 9 and conductivity between 0.02 Sim and 0.2 S/m. On the hand, the malignant tissues have a dielectric constant of around 60 and conductivity around 2 Sim. Thus, there is a significant contrast between the electrical characteristics of the healthy and malignant tissues. From the electromagnetic point of view, this contrast means a significant difference in the scattering properties of those tissues. From the neural network point of view, it is possible to say that the difference in the scattering properties of healthy and malignant tissues means that the scattered signals recorded in different places around

the cervical have the signature of existence of tumor. The comparison between those received signals recorded in different places gives an indication about the place of tumor.

Reviewing the literature shows that most of the research on using neural networks for cervical cancer detection were aimed at enabling radiologist to make accurate diagnoses of cervical cancer on X-ray mammograms. For example, several neural network methods were used in [IS] for automated classification of clustered micro-calcifications in mammograms.

Concerning the UWB systems, the neural network has been used extensively to detect the direction of arrival of ultra-wideband signals. It is also utilized in a simple manner to detect and locate tumor of 2.S mm radius along certain axis.(One dimension) and also in two dimensions but by rotating the place of receiving the signal by steps of 1 degree around the cervical.

This paper presents the use of neural network to detect and locate tumors in cervical with sizes down to 1 mm in radius by using four permanent probes around the cervical. A three-dimensional cervical model is built for the purpose of this investigation. A pulsed plane electromagnetic wave is sent towards the cervical and the scattered signals are collected by four antennas surrounding the cervical. The transmitted pulse occupies the UWB frequency range from 3.1 GHz to 10.6 GHz. A simple feed-forward back-propagation neural network is then used to detect the tumor and locate its position.

6.1.4.1 THE USED NEURAL NETWORK:

Neural network is the best tool in recognition and discrimination between different sets of signals. To get best results using the neural network, it is necessary to choose a suitable architecture and learning algorithm. Unfortunately, there is no guaranteed method to do that. The best way to do that is to choose what is expected to be suitable according to our previous experience and then to expand or shrink the neural network size until a reasonable output is obtained. In this work we

tried different sizes for the neural network using MA TLAB and we found that the best in our case is the model shown in Fig. 2. It has an input layer with 2000 inputs, first hidden layer with 11 nodes, and T ANSIG transfer function, second hidden layer with 7 nodes, and T ANSIG transfer function, and output layer with PURELIN transfer function and 2 outputs. One of the two outputs is used for the detection of tumour, and the other for the localization. T ANSIG transfer function is selected to limit the signal between -1 and 1. For the output layer, PURELIN transfer function is chosen to give all the possible cases for the location of tumour.

The proposed neural network has been used to detect and locate the tumour in two cases. The first case was to detect and locate a tumour in a two-dimensional sector of the cervical model. The location of tumour was considered randomly at the centre and in any of the four quadrature. The second case was to detect and locate tumour anywhere in the three-dimensional model. The neural network has been trained using 100 sets of inputs using the training function (TRAINSCG). Additional 40 Sets of inputs were used to test the performance of each neural network.

6.1.4.2 The Multilayer Perceptron Neural Network Model

The following diagram illustrates a perceptron network with three layers:

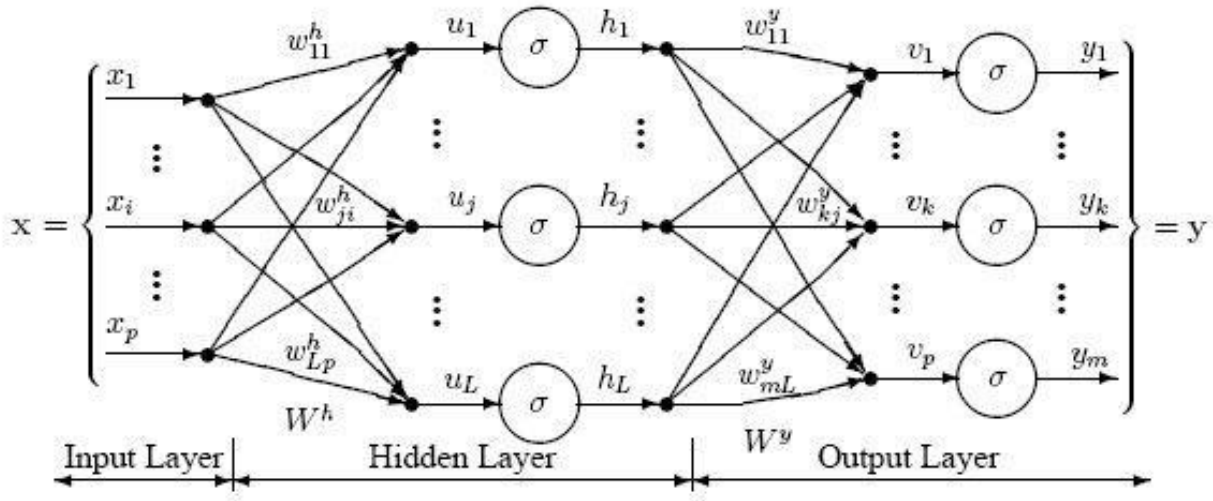


Figure: 6.8 perceptron Network

This network has an **input layer** (on the left) with three neurons, one **hidden layer** (in the middle) with three neurons and an **output layer** (on the right) with three neurons.

There is one neuron in the input layer for each predictor variable. In the case of categorical variables, *N*-1 neurons are used to represent the *N* categories of the variable.

Input Layer — A vector of predictor variable values ($x_1 ... x_p$) is presented to the input layer. The input layer (or processing before the input layer) standardizes these values so that the range of each variable is -1 to 1. The input layer distributes the values to each of the neurons in the hidden layer. In addition to the predictor variables, there is a constant input of 1.0, called the *bias* that is fed to each of the hidden layers; the bias is multiplied by a weight and added to the sum going into the neuron.

Hidden Layer — Arriving at a neuron in the hidden layer, the value from each input neuron is multiplied by a weight (w_{ji}), and the resulting weighted values are added together producing a combined value u_j. The weighted sum (u_j) is fed into a transfer function, σ, which outputs a value h_j. The outputs from the hidden layer are distributed to the output layer.

Output Layer — Arriving at a neuron in the output layer, the value from each hidden layer neuron is multiplied by a weight (w_{kj}), and the resulting weighted values are added together producing a combined value v_j. The weighted sum (v_j) is fed into a transfer function, σ, which outputs a value y_k. The *y* values are the outputs of the network.

If a regression analysis is being performed with a continuous target variable, then there is a single neuron in the output layer, and it generates a single y value. For classification problems with categorical target variables, there are *N* neurons in the

output layer producing *N* values, one for each of the *N* categories of the target variable.

6.1.4.3 Architecture of a NN:

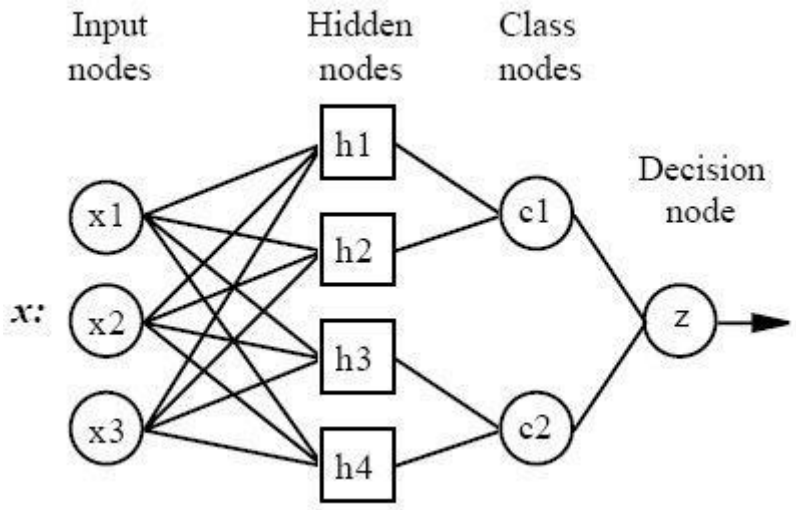


Figure:6.9-Architecture of NN

All NN networks have four layers:

1. **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.

2. **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.

3. **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.

4. **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.

The following diagram is actual diagram or propose network used in our project.

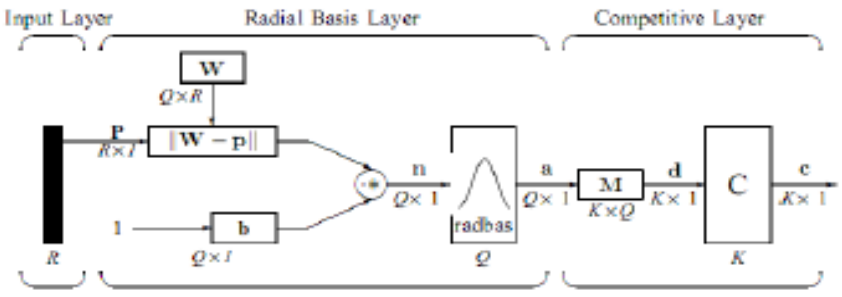


Figure:6.10 Proposed Network

1) Input Layer:

The input vector, denoted as **p**, is presented as the black vertical bar.Its dimension is $R \times 1$. In this paper, $R = 3$.

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** 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**|| **..p**. 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)* = 2 *n e*- (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 ai = *radbas*(||**W**i **- p**|| **..b**i) (2) where **W**i is the vector made of the *i*-th row of **W** and **b**i is the *i*-th element of bias vector **b**.

Some characteristics of Radial Basis Layer:

The *i*-th element of a equals to 1 if the input **p** is identical to the *i*th row of input weight matrix **W**. A radial basis neuron with a weight vector close to the input vector **p** produces a value near 1 and then its output weights in the competitive layer will pass their values to the competitive function. It is also possible that several elements of **a** are close to 1. Since the input pattern is close to several training patterns. .

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.

6.1.4.4 How NN network work:

Although the implementation is very different, neural networks are conceptually similar to *K-Nearest Neighbour* (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:

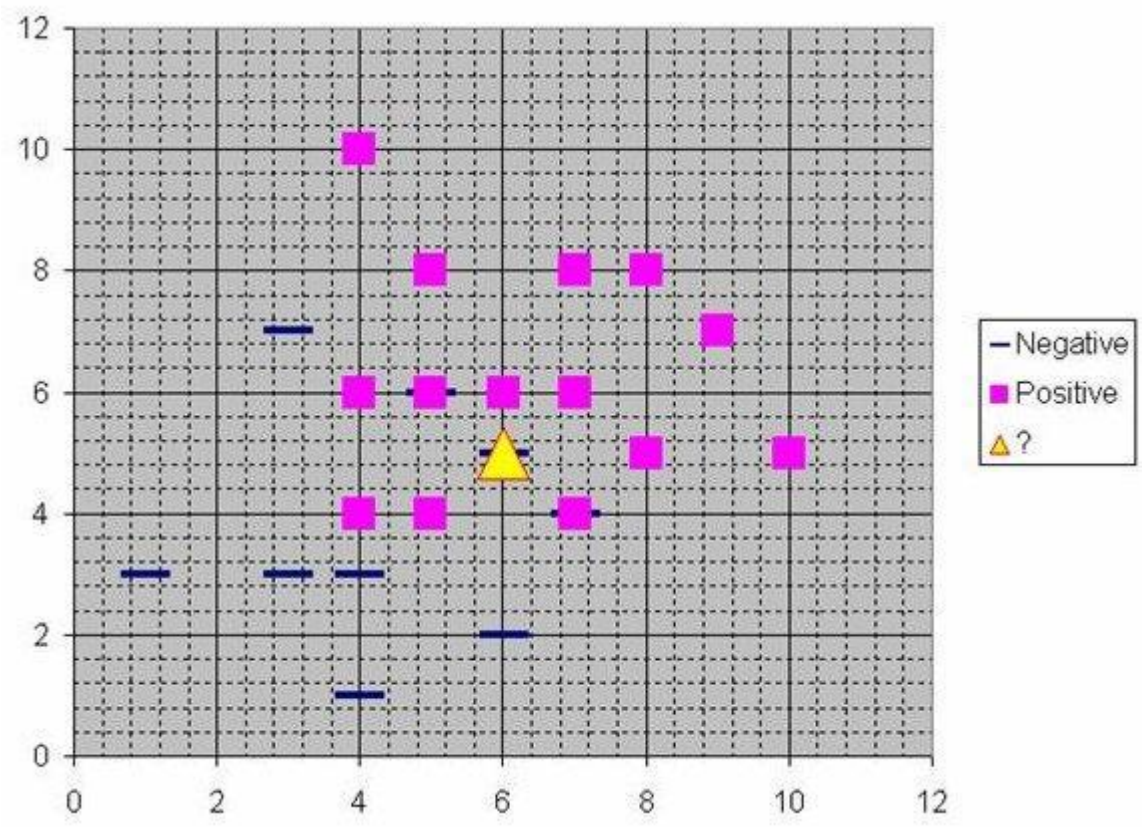


Figure:6.11 graph

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?

The nearest neighbour classification performed for this example depends on how many neighbouring 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.

Weight = RBF (*distance*)

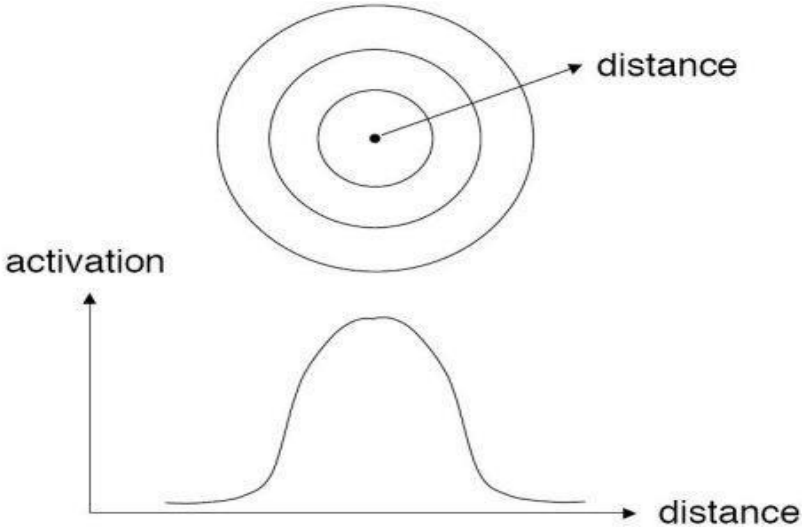


Figure:6.12 distance Graph

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:

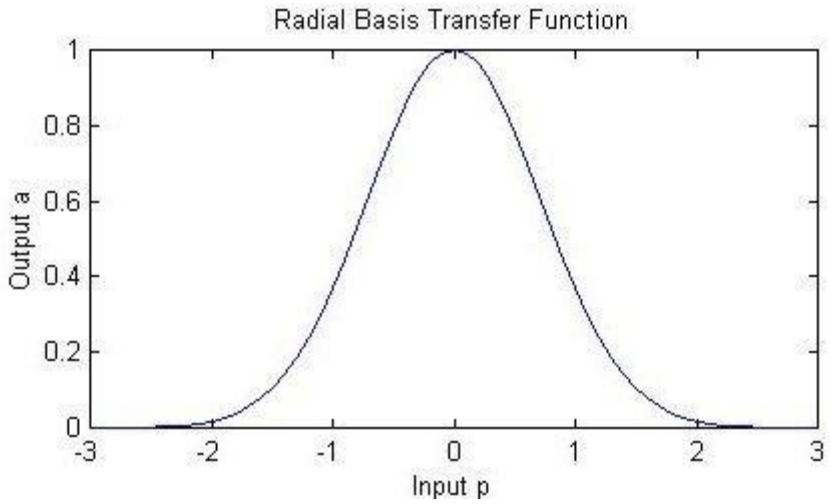


Figure:6.13 Radical Basis Transfer

6.1.4.5 Advantages and disadvantages of NN networks:

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

CHAPTER-7
IMPLEMENTATION

7.1 SAMPLE 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 16-May-2022 14:07:22
```

```
% 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 ='D:\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('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
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 = 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);
imshow(J);
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([Level2,cH; cV,cD],'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);
```

7.2 OUTPUT SCREENSHOTS:

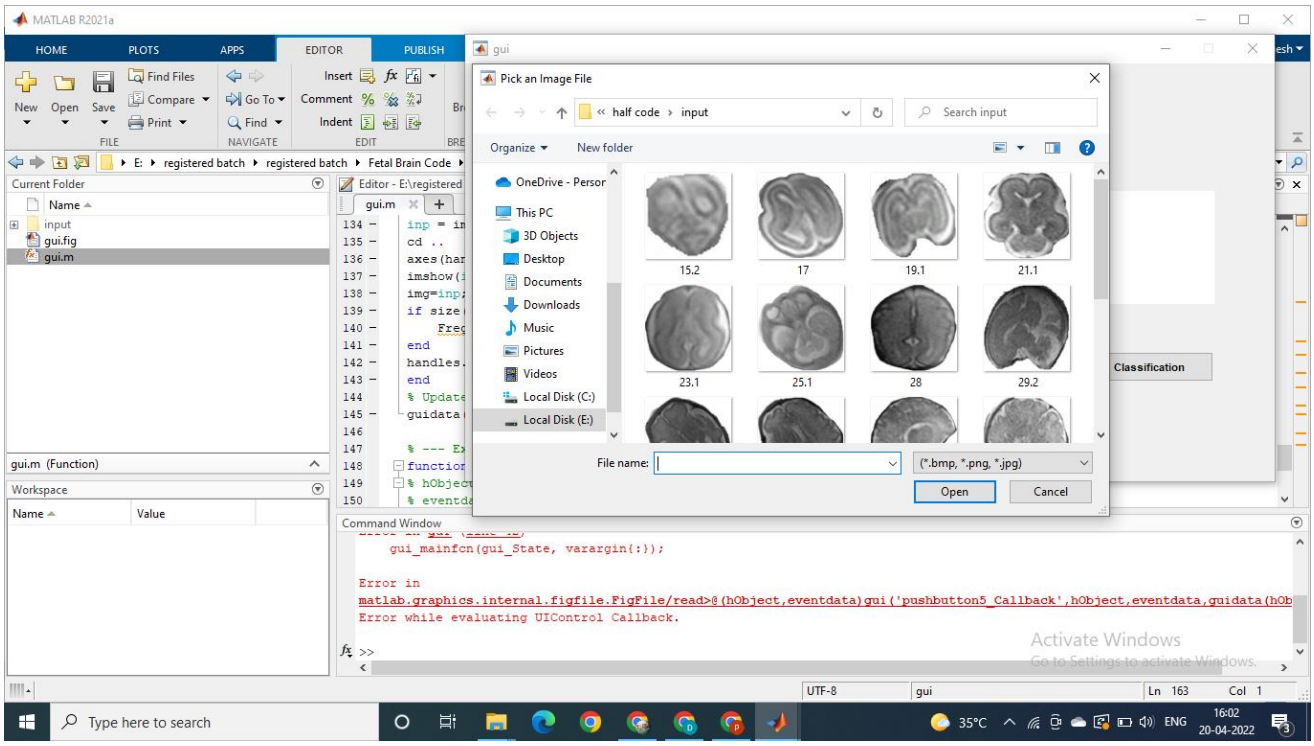


Figure:7.1 selection of input image

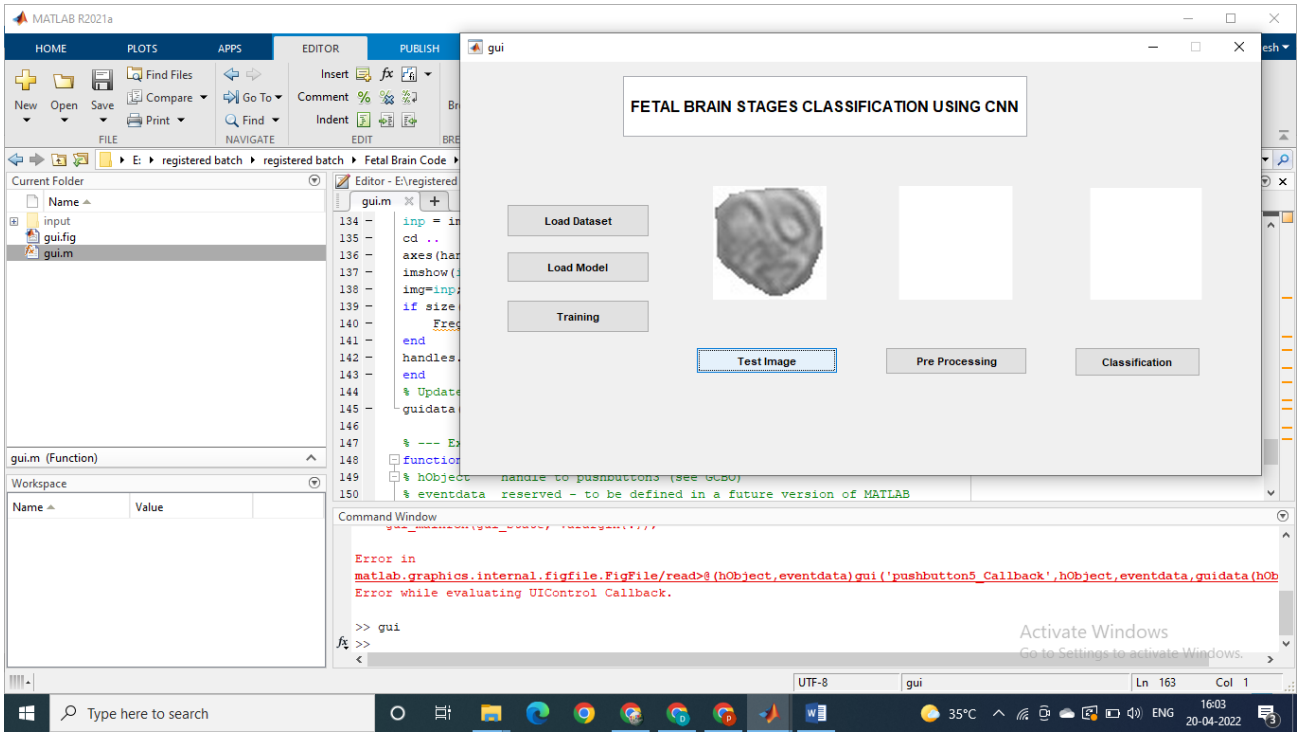


Figure 7.2-Testing the selected image.

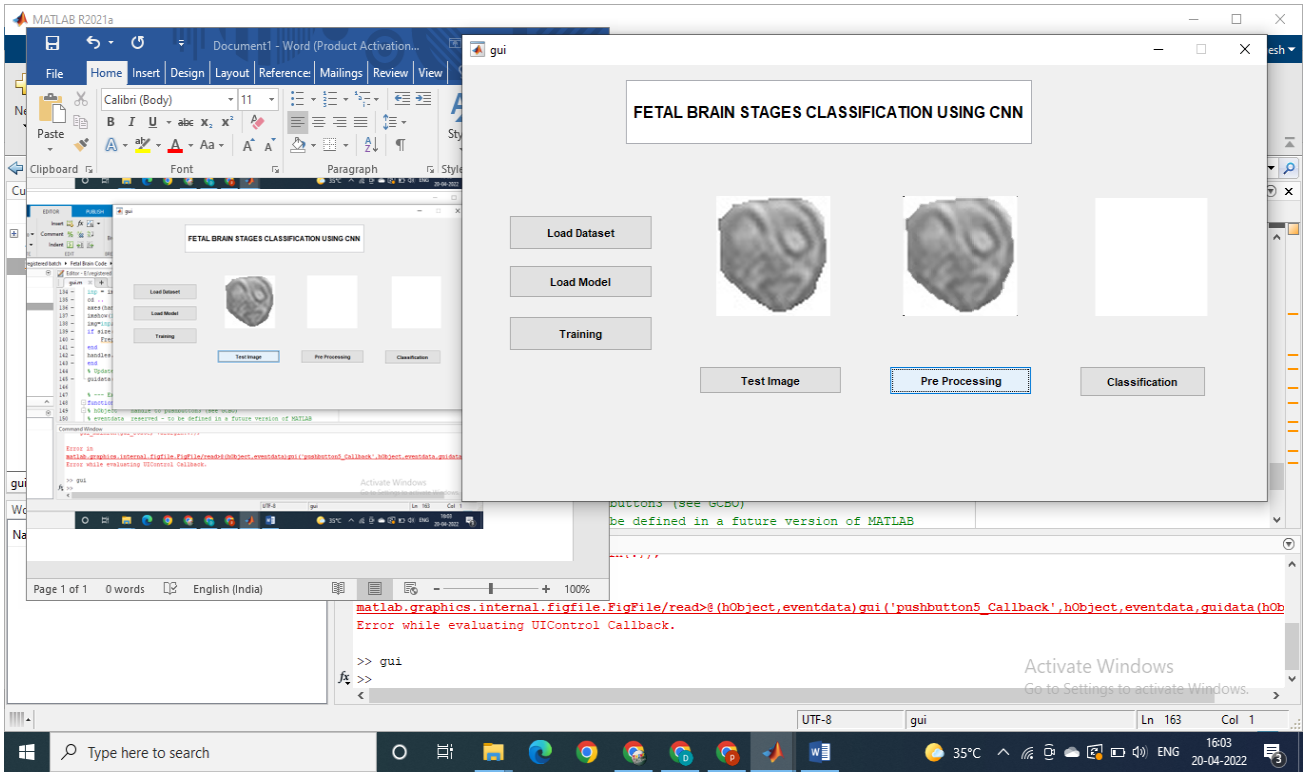


Figure:7.3-Pre-processing the Image

Figure:7.4-Epoch level of Dataset.

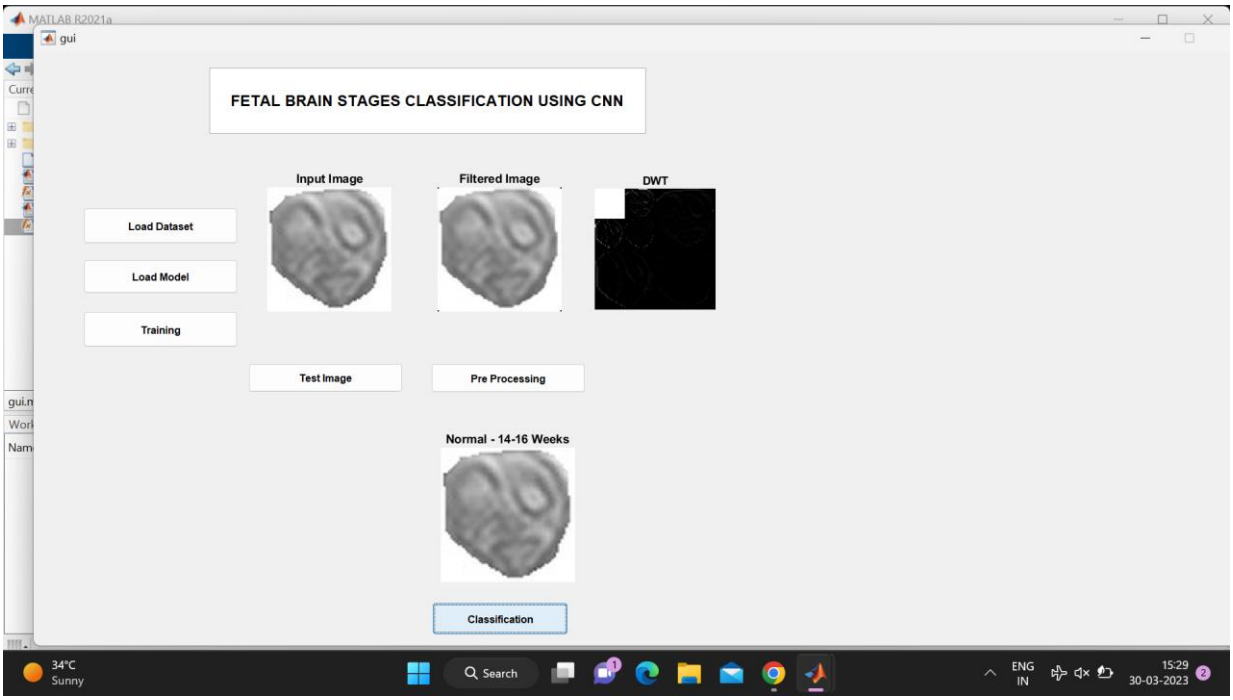


Figure:7.5-The final Result

CHAPTER-8
TESTING AND MAINTENANCE

8.1 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

8.2 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.3 UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed

Features to be tested.

• Verify that the entries are of the correct format

• No duplicate entries should be allowed

• All links should take the user to the correct page.

8.4 INTERGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results

All the test cases mentioned above passed successfully. No defects encountered.

8.5 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.6 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results

All the test cases mentioned above passed successfully. No defects encountered.

8.7 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

CHAPTER-9
CONCLUSION

9.1 CONCLUSION:

Ultrasound 2D fetal brain imaging has become an essential diagnostic tool in obstetrics for evaluating the fetal brain and predicting potential diseases or abnormalities. With the advancement of technology, 2D ultrasound imaging has become more accurate and efficient in identifying fetal brain structures, including the cerebral cortex, ventricles, and cerebellum. These structures are crucial in predicting potential diseases, including spina bifida, anencephaly, and hydrocephalus, among others.

Classification of fetal brain images using ultrasound is a challenging task due to the complexity and variability of fetal brain development. However, recent studies have demonstrated the potential of artificial intelligence (AI) and machine learning (ML) algorithms in improving the accuracy of fetal brain image classification and disease prediction.

In these studies, researchers have used various algorithms, including convolutional neural networks (CNNs) and deep learning models, to analyze ultrasound images and classify them into normal or abnormal. These algorithms have been shown to achieve high accuracy rates in identifying fetal brain abnormalities, such as ventriculomegaly and microcephaly, which are associated with significant health risks.

Moreover, recent advancements in AI and ML have enabled the development of more sophisticated models capable of predicting the likelihood of specific fetal brain diseases or abnormalities. These models use data from multiple sources, including maternal medical history, genetic testing, and ultrasound imaging, to predict the likelihood of specific fetal brain diseases or abnormalities.

The use of AI and ML algorithms in ultrasound 2D fetal brain image classification and disease prediction has the potential to improve prenatal care and reduce the incidence of fetal brain abnormalities. These technologies can help identify potential health risks earlier, enabling doctors to provide more targeted treatment plans and counseling for parents.

By analyzing ultrasound images of fetal brains, machine learning algorithms can classify images and predict the likelihood of certain diseases or abnormalities. While this technology shows great potential for improving prenatal care and diagnosis, it is still in the early stages of development. More research and clinical trials are needed to validate the accuracy of the predictions made by these algorithms.

Additionally, it is important to consider the ethical implications of using these technologies, such as the potential for misdiagnosis or overdiagnosis, and the impact on patient privacy.

The traditional method of manually examining ultrasound images by physicians is time-consuming, subjective, and may result in errors due to the variability in interpretation. However, with the advancement in machine learning and computer vision techniques, there has been a significant improvement in the accuracy and efficiency of fetal brain image classification and disease prediction.

Overall, while the use of ultrasound 2D fetal brain image classification and disease prediction are a promising development in medical imaging, it is important to proceed with caution and further research before implementing it widely in clinical practice.

In conclusion, the use of ultrasound 2D fetal brain imaging combined with AI and ML algorithms has the potential to significantly improve prenatal care and reduce the incidence of fetal brain abnormalities. As technology continues to advance, we can expect further improvements in the accuracy and efficiency of fetal brain image classification and disease prediction, ultimately leading to better outcomes for both mothers and babies.

9.2 FUTURE ENHANCEMENT

The future of ultrasound 2D fetal brain image classification and disease prediction looks promising, with ongoing research aimed at enhancing the accuracy, efficiency, and reliability of these diagnostic tools.

One potential area of enhancement is the use of 3D ultrasound imaging, which can provide more detailed and comprehensive images of fetal brain structures. 3D imaging can enable more accurate and earlier detection of potential abnormalities, allowing for earlier intervention and treatment.

Another area of research involves the development of more sophisticated AI and ML algorithms capable of analyzing large datasets of ultrasound images and medical records. These algorithms can improve the accuracy of fetal brain image classification and disease prediction, leading to outcomes for mothers and babies.

Additionally, advancements in computer processing power and data storage capabilities are enabling researchers to develop more complex models that can handle larger datasets and images in real-time. This can significantly reduce the time and cost of ultrasound examinations, making them more accessible and affordable for patients.

Furthermore, the integration of ultrasound imaging with other diagnostic tools, such as genetic testing and magnetic resonance imaging (MRI), can provide a more comprehensive understanding of fetal brain development and potential health risks.

Finally, ongoing research into the use of ultrasound imaging for fetal brain functional analysis is opening up new possibilities for the early detection and treatment of neurological disorders. Functional ultrasound imaging can detect changes in blood flow and brain activity, enabling the identification of potential neurological disorders before they become symptomatic.

In conclusion, the future of ultrasound 2D fetal brain image classification and disease prediction is promising, with ongoing research aimed at enhancing the accuracy, efficiency, and reliability of these diagnostic tools.

CHAPTER-10
REFFERENCE

[1] A. Gholipour, J. A. Estroff, and S. K. Warfield, “Robust super-resolution volume reconstruction from slice acquisitions: Application to fetal brain MRI,” IEEE Transactions on Medical Imaging, vol. 29, no. 10, pp. 1739–1758, Oct. 2010.

[2] A. Largent, K. Kapse, S. D. Barnett, J. D. Asis-Cruz, M. Whitehead, J. Murnick, L. Zhao, N. Andersen, J. Quistorff, C. Lopez, and C. Limperopoulos, “Image quality assessment of fetal brain MRI using multi-instance deep learning methods,” Journal of Magnetic Resonance Imaging, vol. 54, no. 3, pp. 818–829, Sep. 2021.

[3] A. Mang and L. Ruthotto, “A Lagrangian Gauss–Newton–Krylov solver for mass- and intensity-preserving diffeomorphic image registration,” SIAM Journal on Scientific Computing, vol. 39, no. 5, pp. B860–B885, Sep. 2017.

A. Melbourne, T. Doel, S. Dymarkowski, P. D. Coppi, A. L. David, J. Deprest, S. Ourselin, and T. Vercauteren, “An automated framework for localization, segmentation and super-resolution reconstruction of fetal brain MRI,” NeuroImage, vol. 206, no. 116324, Feb. 2020.

[4] A. Singh, S. S. M. Salehi, and A. Gholipour, “Deep predictive motion tracking in magnetic resonance imaging: Application to fetal imaging,” IEEE Transactions on Medical Imaging, vol. 39, no. 11, pp. 3523–3534, Nov. 2020.

[5] A. Uus, T. Zhang, L. H. Jackson, T. A. Roberts, M. A. Rutherford, J. V. Hajnal, and M. Deprez, “Deformable slice-to-volume registration for motion correction of fetal body and placenta MRI,” IEEE Transactions on Medical Imaging, vol. 39, no. 9, pp. 2750–2759, Sep. 2020.

[6] American College of Radiology, “ACR-SPR practice parameter for the safe and optimal performance of fetal magnetic resonance imaging (MRI), resolution 45,” https://www.acr.org/-/media/ACR/Files/ Practice-Parameters/mr-fetal.pdf, 2020.

[7] B. Hou, B. Khanal, A. Alansary, S. McDonagh, A. Davidson, M. Rutherford, J. V. Hajnal, D. Rueckert, B. Glocker, and B. Kainz, "3-D reconstruction in canonical co-ordinate space from arbitrarily oriented 2-D images," IEEE Transactions on Medical Imaging, vol. 37, no. 8, pp. 1737–1750, Aug. 2018.

[8] C. L. Herrera, J. J. Byrne, H. R. Clark, D. M. Twickler, and J. S. Dashe, "Use of fetal magnetic resonance imaging after sonographic identification of major structural anomalies," Journal of Ultrasound in Medicine, vol. 39, no. 10, pp. 2053–2058, Oct. 2020.

[9] C. S. Burrus, J. A. Barreto, and I. W. Selesnick, "Iterative reweighted least-squares design of FIR filters," IEEE Transactions on Signal Processing, vol. 42, no. 11, pp. 2926–2936, Nov. 1994.

[10] D. C. Noll, F. E. Boada, and W. F. Eddy, "A spectral approach to analyzing slice selection in planar imaging: Optimization for throughplane interpolation," Magnetic Resonance in Medicine, vol. 38, no. 1, pp. 151–160, jul 1997.

[11] E. Ferrante and N. Paragios, "Slice-to-volume medical image registration: A survey," Medical Image Analysis, vol. 39, pp. 101–123, Jul. 2017.

[12] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett, "Deep learning techniques for inverse problems in imaging," IEEE Journal on Selected Areas in Information Theory, vol. 1, no. 1, pp. 39–56, May 2020.

[13] M. Chen, W. Lu, Q. Chen, K. J. Ruchala, and G. H. Olivera, "A simple fixed-point approach to invert a deformation field," Medical Physics, vol. 35, no. 1, pp. 81–88, Jan. 2008

[14] M. Ebner, G. Wang, W. Li, M. Aertsen, P. A. Patel, R. Aughwane,

[15] M. F. Beg, M. I. Miller, A. Trouve, and L. Younes, "Computing large´ deformation metric mappings via geodesic flows of diffeomorphisms," International Journal of Computer Vision, vol. 61, no. 2, pp. 139–157, Feb. 2005.

[16] M. Kuklisova-Murgasova, G. Quaghebeur, M. A. Rutherford, J. V. Hajnal, and J. A. Schnabel, "Reconstruction of fetal brain MRI with intensity matching and

complete outlier removal," Medical Image Analysis, vol. 16, no. 8, pp. 1550–1564, Dec. 2012.

[17] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," Communications in Statistics - Theory and Methods, vol. 6, no. 7, pp. 813–827, 1977.

[18] R. A. Maronna, R. D. Martin, V. J. Yohai, and M. Salibian-Barrera,´ Robust statistics, Theory and Methods (with R), 2nd ed., ser. Wiley Series on Probability & Statistics. John Wiley & Sons Ltd, 2019.

[19] R. Heckel and P. Hand, "Deep decoder: Concise image representations from untrained non-convolutional networks," arXiv:1810.03982v2, Feb. 2019, https://arxiv.org/abs/1810.03982.

[20] S. S. M. Salehi, S. Khan, D. Erdogmus, and A. Gholipour, "Real-time deep pose estimation with geodesic loss for image-to-template rigid registration," IEEE Transactions on Medical Imaging, vol. 38, no. 2, pp. 470–81, Feb. 2019.

[21] W. Shi, G. Yan, Y. Li, H. Li, T. Liu, C. Sun, G. Wang, Y. Zhang, Y. Zou, and D. Wu, "Fetal brain age estimation and anomaly detection using attention-based deep ensembles with uncertainty," NeuroImage, vol. 223, no. 117316, Dec. 2020.