

A PROJECT REPORT ON
A NOVEL METHOD OF CLASSIFICATION OF TUMOR
USING CNN

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA

In Partial fulfillment for the award of the degree Of

BACHELOR OF TECHNOLOGY

In

ELECTRONICS & COMMUNICATION ENGINEERING

By

R. MOHITH VENKATA SAI	188X1A0479
T. SATYA NANDAN	188X1A0498
P. ROHITHA	188X1A0475
B. PAVAN BHARGAV	188X1A0474
CH. CHEERANJIVI	188X1A0461

Under The Esteemed Guidance of

MR.Y. MURALI KRISHNA

M. Tech

ASSOCIATE PROFESSOR



DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
KALLAM HARANADHAREDDY INSTITUTE OF TECHNOLOGY
(APPROVED BY AICTE, AFFILIATED TO JNTU-KAKINADA, ACCREDITED
WITH NAAC 'A' GRADE, ACCREDITED BY NBA)

NH-5, CHOWDAVARAM, GUNTUR-522019

KALLAM HARANADHAREDDY INSTITUTE OF TECHNOLOGY

(APPROVED BY AICTE, AFFILIATED TO JNTU-KAKINADA

ACCREDITED WITH NAAC 'A' GRADE, ACCREDITED BY NBA)

NH-5, CHOWDAVARAM, GUNTUR-522019

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING



CERTIFICATE

This is to certify that the project report entitled “A NOVEL METHOD OF CLASSIFICATION OF TUMOR USING CNN” is submitted by **R. MOHITH VENKATASAI (188X1A0479); T. SATYA.NANDAN (188X1A0498) P. ROHITHA (188X1A0475); B. PAVAN BHARGAV (188X1A0474); CH. CHIRANJEEVI (188X1A0461)** to the Jawaharlal Nehru Technological University Kakinada, In partial fulfillment for the award of degree of Bachelor of Technology in Electronic and Communication Engineering is a bonafide record of the project work carried out by them under my supervision during the year 2021-2022.

PROJECT GUIDE:

Y. MURALI KRISHNA

M.Tech

ASSOCIATE PROFESSOR

HEAD OF THE DEPARTMENT:

Dr. S. SURYA NARAYANA

M.Tech, Ph.D, MISTE, MIETE

PROFESSOR, HOD & Dean

PROJECT CO-ORDINATOR

Mr. A. MANI KANTH

M.Tech, MISTE, MIETE

ASSOCIATE PROFESSOR

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We profoundly express our gratitude and respect towards our honorable chairman **SRI KALLAM HARANADHA REDDY**, Chairman, KHIT for his precious support in the college.

We express our deepest gratitude to Dynamic Director of our institute **Dr. M. UMASHANKAR REDDY**, Director, KHIT for his valuable guidance and blessings.

We would like to thank **Dr. B. S. B. REDDY** Principal, Kallam Haranadhareddy Institute of Technology, chowdavaram, Guntur for providing required resources to complete this project work.

We express our sincere thanks to **Dr. S. SURYANARAYANA**, Head of the Electronics and Communication Engineering Department, KHIT for his encouragement and valuable guidance in bringing shape to this project.

We are indebted to our guide **Mr. Y. MURALI KRISHNA**, Department of Electronics and Communication Engineering, KHIT for his guidance and help throughout the development of this Project work by providing us with required information.

We also express our sincere thanks to faculty of Electronics and Communication Engineering department for providing assistance throughout our project.

Finally, we would like to thank our parents and friends for being supportive all the time, and we are very much obliged to them.

R. MOHITH VENKATA SAI	188X1A0479
T. SATYA NANDAN	188X1A0498
P. ROHITHA	188X1A0475
B. PAVAN BHARGAV	188X1A0474
CH. CHEERANJIVI	188X1A0461

DECLARATION

We hereby declare that the entire project work embodied in this dissertation entitled “**A NOVEL METHOD OF CLASSIFICATION OF TUMOR USING CNN**” been independently carried out by us. As per our knowledge, no part of this work has submitted for any degree in any institution, university and organization previously.

R. MOHITH VENKATA SAI	188X1A0479
T. SATYA NANDAN	188X1A0498
P. ROHITHA	188X1A0475
B. PAVAN BHARGAV	188X1A0474
CH. CHEERANJIVI	188X1A0461

TABLE OF CONTENTS	PAGE NO
ACKNOWLEDGEMENT.....	I
DECLARATION.....	II
TABLE OF CONTENTS.....	III
LIST OF FIGURES.....	VI
ABSTRACT.....	VIII

INDEX

CHAPTER-1 INTRODUCTION.....	1-2
1.1 Back Ground.....	1
1.2 Problem Definition.....	1
1.3 Motivation.....	1
1.4 Existing System.....	1
1.5 Proposed System.....	2
CHAPTER-2 LITERATURE SURVEY.....	3-5
2.1 RGB to Gray Conversion.....	3
2.2 Pre-Processing and Enhancement.....	3
2.3 Segmentation.....	4
2.3.1 Otsu Method.....	4
2.4 Feature Extraction.....	4
2.4.1 Discrete Wavelet Transform.....	4
2.5 Classification.....	5
2.5.1 Support Vector Machine.....	5
2.5.2 Convolution Neural Network.....	5
2.6 Transfer Learning.....	5
CHAPTER-3 EXISTING SYSTEM.....	6-8
3.1 Software Architecture.....	6
3.2 Pre-Processing.....	7
3.3 Otsu Method.....	7
3.4 DWT & PCA.....	7
3.5 Support Vector Machine.....	8

CHAPTER-4 PROPOSED SYSTEM.....	9-41
4.1 Tumor.....	9
4.1.1 Benign vs Malignant Brain Tumor.....	10
4.1.2 Types of Brain Tumors.....	10
4.2 Software Architecture.....	14
4.3 Segmentation.....	14
4.3.1 Thresholding	15
4.4 Discrete Wavelet Transform.....	19
4.4.1 1-D Discrete Wavelet Transform.....	19
4.4.2 2-D Discrete Wavelet Transform.....	20
4.5 Principal Components Analysis on raw data.....	25
4.6 Classification.....	27
4.6.1 Support Vector Machine.....	27
4.6.2 Convolution Neural Networks.....	37
4.7 Transfer Learning.....	38
4.7.1 How Transfer Learning Works.....	39
4.7.2 Why use Transfer Learning.....	40
CHAPTER-5 CONVNET.....	42-51
5.1 Definition	42
5.2 Architecture.....	43
5.2.1 Convolution Layers.....	43
5.2.2 Pooling Layers.....	43
5.2.3 Fully Connected Layers.....	43
5.2.4 Receptive Field.....	44
5.2.5 Weights.....	44
5.3 Distinguishing Features.....	44
5.4 Building Blocks.....	46
5.4.1 Convolution Layer.....	46
5.4.2 Pooling Layer.....	49
5.4.3 ReLU Layer.....	50
5.4.4 Fully Connected Layer.....	50
5.4.5 Loss Layer.....	50
5.4.6 SoftMax.....	50

CHAPTER-6	IMPLEMENTATION	52-59
6.1	Classification between Benign or Malignant	52
6.2	Classification of Tumor Type.....	55
6.2.1	Google Net.....	55
6.3	Tumor Spot.....	58
CHAPTER-7	RESULTS.....	60-61
CHAPTER-8	CONCLUSION & FUTURES SCOPE.....	62
8.1	Conclusion.....	62
8.2	Future Scope.....	62
	REFERENCES.....	63
	GLOSSARY.....	64-65
	APPENDIX.....	66

LIST OF FIGURES

FIG. NO	P.NO
3.1 Existing Software Architecture.....	6
4.1 Tumor.....	9
4.2 Benign vs Malignant.....	10
4.3 Glioma Tumor.....	12
4.4 Pituitary Tumor.....	13
4.5 Meningioma Tumor.....	14
4.6 Software Architecture.....	14
4.7 Thresholding of brain tumor from grayscale MRI image.....	15
4.8 Flow of Otsu's thresholding in Image Processing.....	17
4.9 1-D DWT.....	19
4.10 Block Diagram of DWT	20
4.11 DWT for Lena image	20
4.12 2-D DWT.....	21
4.13 Forward & Inverse Transform.....	23
4.14 Lifting Scheme.....	23
4.15 Training a SVM Model.....	27
4.16 Simple Neural Network, Multilayer Perceptron.....	28
4.17 Illustration of Linear SVM.....	29
4.18 Representation of Hyper Plane.....	30
4.19 Representation of Support Vectors.....	31
4.20 Kernels.....	32
4.21 Feature Space Representation.....	33
4.22 How to control complexity.....	35
4.23 Deep Learning Layers.....	37
4.24 CNN.....	37
4.25 Classification Layer.....	38
4.26 Transfer Learning.....	39
4.27 Feature extraction.....	41
5.1 CNN layers arranged in 3 dimensions.....	45
5.2 CNN Network.....	47
5.3 Max pooling with a 2x2 filter and stride = 2.....	49

5.4	Example of SoftMax.....	51
6.1	Benign or Malignant User Interface.....	52
6.2	SVM Flow Chart.....	53
6.3	Labels in Trained data of SVM.....	54
6.4	Measurements in Trained data of SVM.....	55
6.5	Google Net Architecture.....	56
6.6	Training Progress.....	56
6.7	Tumor Type UI.....	57
6.8	Flow chart of Classification of type.....	58
6.9	Block diagram of Tumor detection.....	58
6.10	UI of Tumor Spot.....	59
7.1	Result of Benign or Malignant.....	60
7.2	Result of Tumor Type.....	60
7.3	Result of Detection of Tumor.....	61

ABSTRACT

Brain tumor detection and classification is that the most troublesome and tedious task within the space of medicative image getting ready. MRI (Magnetic Resonance Imaging) may be a medicative procedure, typically adopted by the medical specialist for illustration of inner structure of the brain with no surgery. Magnetic resonance imaging provides long information concerning the human delicate tissue, which helps within the conclusion of brain tumor. Precise segmentation of magnetic resonance imaging image is basic for the conclusion of brain tumor by laptop supported clinical device. This paper is concentrated towards the look of Associate in Nursing best and additional correct approach for the detection of neoplasm from brain magnetic resonance imaging scans and if it confirms the presence of tumor then it's focused on evaluating its stage, i.e., benign or malignant. We've through an experiment shown that our projected methodology features a larger accuracy than different existent strategies for classifying tumor kind to be either as Malignant or Benign. And also classifying the tumor is of what type based on three categories. MRI scan images of the brain. This method incorporates with, segmentation and morphological operations which are the basic concepts of image processing. Detection and extraction of tumor from MRI scan images of the brain is done by using MATLAB software.

1. INTRODUCTION

This chapter discusses the main concepts that the project is based on. It identifies what the project is actually meant to accomplish.

1.1 Background

The human body consists of myriad number of cells. When cell growth becomes uncontrollable the extra mass of cell transforms into tumor. CT scans and MRI scans are used for identification of the tumor. The goal of our study is to accurately detect tumors in the brain and classify it through the means of several techniques involving medical image processing, pattern analysis, and computer vision for enhancement, segmentation and classification of brain diagnosis. This system can be used by neurosurgeons, radiologists and healthcare specialists.

The system is expected to improve the sensitivity, specificity, and diagnostic efficiency of brain tumor screening using industry standard simulation software tool, MATLAB. These techniques involve pre-processing of MRI scans collected from online cancer imaging archives as well as scans obtained from several pathology labs. Images are resized and then we apply the proposed algorithms for segmentation and classification. The system is expected to improve the brain tumor screening procedure currently at use, and possibly reduce health care costs by decreasing the need for followup procedures. Several processing steps are required for the accurate characterization and analysis of biomedical image data.

1.2 Problem Definition

Our study deals with automated brain tumor detection and classification. Normally the anatomy of the brain is analysed by MRI scans or CT scans. Our system aims to detect if the given MRI scan has a tumor and if found it then classifies the tumor as malignant or benign. And also classifies the tumor is of which type based on three types Pituitary tumor, Meningioma tumor, Glioma tumor.

1.3 Motivation

The motivation of the proposed application is to aid neurosurgeons and radiologists in detecting brain tumors in an inexpensive and non-invasive manner.

1.4 Existing System

The existing system takes the Computed Tomography scan of the brain and detects the

presence of tumor in the given scan and classifies whether the tumor is Benign or malignant, that is benign can be taken as non-cancerous and Malignant can be taken as cancerous. And also the existing provides the morphological operation and displays the tumor in the scan. The required features for the classification is extracted by the discrete wavelet transform (DWT).

The classification of the tumor between benign or malignant is done by the Super vector Machine algorithm.

1.5 Proposed System

The proposed system achieves all the features of the existing system but in addition to that here we are proposing the classification in a next level. Here the tumor is classified between the three types Pituitary tumor, Meningioma tumor, Glioma tumor.

And for this classification we are using the Convolution Neural Network. Instead of the CT scans we are using the Magnetic Resonance image of the brain. Which has a low noise in the image when compared to CT scan of the brain.

2. LITERATURE SURVEY

This chapter discusses the papers referenced in preparation for undertaking this project. These papers serve as a benchmark to enable this project to be undertaken.

In Medical diagnosis, robustness and accuracy of the prediction algorithms are very important, because the result is crucial for treatment of patients. There are many popular classification and clustering algorithms used for prediction. The goal of clustering a medical image is to simplify the representation of an image into a meaningful image and make it easier to analyse. Several Clustering and Classification algorithms are aimed at enhancing the prediction accuracy of diagnosis process in detecting abnormalities.

The first section of this chapter presents different pre-processing techniques on the MRI Images obtained. Types of Segmentation algorithms used are given in the second section. The next section discusses about the different feature extraction methods and finally the classification algorithms have been presented followed by the summary of this chapter.

2.1 RGB to Gray Conversion

An RGB image can be viewed as three images(a red scale image, a green scale image and a blue scale image) stacked on top of each other. In MATLAB, an RGB image is basically a $M \times N \times 3$ array of colour pixel, where each colour pixel is a triplet which corresponds to red, blue and green colour component of RGB image at a specified spatial location. Similarly, A Grayscale image can be viewed as a single layered image. In MATLAB, a grayscale image is basically $M \times N$ array whose values have been scaled to represent intensities. In MATLAB, there is a function called `rgb2gray()` is available to convert RGB image to grayscale image. Here we will convert an RGB image to grayscale image without using `rgb2gray()` function.

2.2 Pre – Processing and Enhancement

This process was not a mandatory step; it is required only when the MRI image contains the X-ray labels and any other details in the scan. Pre-processing and enhancement techniques are used to improve the detection of the suspicious region from Magnetic Resonance Image (MRI). This section presents the gradient - based image enhancement method for brain MR images which is based on the first derivative and local statistics. The pre-processing and enhancement method consists of two steps; first the removal of film artifacts such as labels and X - ray marks are removed from the MRI using tracking algorithm. Second, the removal of

high frequency components using weighted median filtering technique. It gives high resolution MRI compared to median filter, adaptive filter and spatial filter. The performance of the proposed method is also evaluated by means of peak single - to noise - ratio (PSNR), Average Signal - to - Noise Ratio (ASNR).

2.3 Segmentation

Image segmentation is the primary step and the most critical tasks of image analysis. Its purpose is that of extracting from an image by means of image segmentation. The mechanization of medical image segmentation has established wide application in diverse areas such as verdict for patients, treatment management planning, and computer - integrated surgery. The segmentation algorithm that have been implemented:

2.3.1 Otsu Method:

Otsu's thresholding is a non - linear operation that converts a grayscale image into a binary image where the two levels are assigned to pixel those that are below or above the specified threshold value. The two levels in a binary image are assigned to pixels below or above the particular threshold. It is based on threshold range by statistical calculations. Otsu suggested minimizing the weighted sum of within - class variances of the object and background pixels to establish an optimum threshold.

2.4 Feature Extraction

Feature extraction is process of extracting quantitative information from an image such as color features, texture, shape and contrast. Here, we have used discrete wavelet transform (DWT) for extracting wavelet coefficients and gray-level co-occurrence matrix (GLCM) for statistical feature extraction.

2.4.1 Discrete Wavelet Transform:

The wavelet was used to analyse different frequencies of an image using different scales. Here, we are using discrete wavelet transform (DWT) which is a powerful tool for feature extraction. It was used to extract coefficient of wavelets from brain MR images. The wavelet localizes frequency information of signal function which was important for classification. By using 2D discrete wavelet transform, the images were decomposed into spatial frequency components which were extracted from LL (low-low) sub-bands and since HL (high-low) sub-bands have higher performance when compared to LL (low-low), we have

used both LL (low–low) and HL (high–low) for better analysis which describes image text features.

2.5 Classification

In classification tasks, there are often several candidate feature extraction methods available. The most suitable method can be chosen by training neural networks to perform the required classification task using different input features (derived using different methods). The error in the neural network response to test examples provides an indication of the suitability of the corresponding input features (and thus method used to derive them) to the considered classification task. Following are the classification algorithms that have been implemented:

2.5.1 Support Vector Machine:

SVM is one of the classification technique applied on different fields such as face recognition, text categorization, cancer diagnosis, glaucoma diagnosis, microarray gene expression data analysis. SVM utilizes binary classification of brain MR image as normal or tumor affected. SVM divides the given data into decision surface, (i.e. a hyperplane) which divides the data into two classes. The prime objective of SVM is to maximize the margins between two classes of the hyper-plane. Dimensionality reduction and precise feature set given as input to the SVM on the duration of training part as well as during the testing part. SVM is based on binary classifier which employs supervised learning to provide better results.

2.5.2 Convolutional Neural Networks:

Convolutional Neural Networks (CNNs) have proven to be very successful frameworks for image recognition. In the past few years, variants of CNN models achieve increasingly better performance for object classification.

2.6 Transfer Learning

Transfer learning is the reuse of a pre-trained model on a new problem. It's currently very popular in deep learning because it can train deep neural networks with comparatively little data. This is very useful in the data science field since most real-world problems typically do not have millions of labeled data points to train such complex models.

We'll take a look at what transfer learning is, how it works, why and when you it should be used. Additionally, we'll cover the different approaches of transfer learning and provide you with some resources on already pre-trained models.

3. EXISTING SYSTEM

This chapter discusses about the Existing System and Methods used in it

Introduction

The Existing System uses the Support Vector Machines (SVM) to classify the tumor as Benign or Malignant. And for segmentation process uses the otsu binarization and displays cancer in the given MRI scan.

3.1 Software Architecture

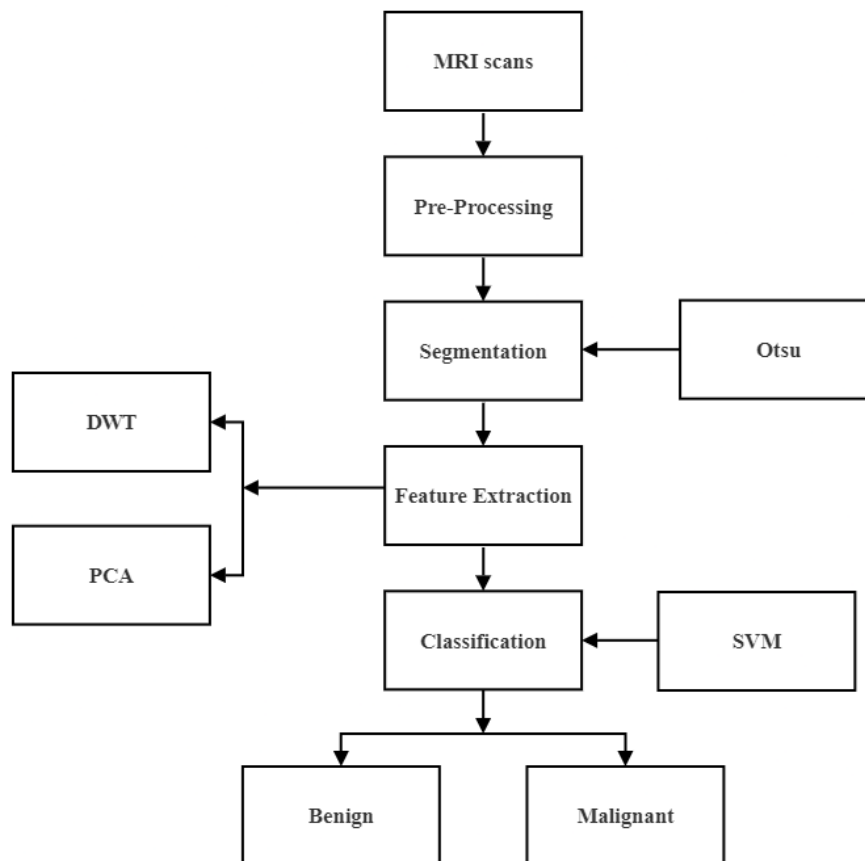


Fig 3.1 Existing Software Architecture

The MRI scans undergo the pre-processing techniques, which include image resizing and image cropping to remove the names or any other details, which may cause accuracy loss for the prediction of the tumor. Median filters are used to remove the noise. Next, the image undergoes Otsu's technique, and the image is displayed to the user, which helps the user to spot the tumor in the MRI scan.

Next, The Features are extracted from the MRI scan using the Discrete Wavelet Transformation and Principle Component Analysis. And the extracted features are passed to the Support Vector Machines (SVM), and here based on the Trained data, the system classifies it as benign or malignant.

3.2 Pre – Processing

Pre-processing and enhancement techniques are used to improve the detection of the suspicious region from Magnetic Resonance Image (MRI). This section presents the gradient - based image enhancement method for brain MR images which is based on the first derivative and local statistics. The pre-processing and enhancement method consists of two steps; first the removal of film artifacts such as labels and X - ray marks are removed from the MRI using tracking algorithm. Second, the removal of high frequency components using weighted median filtering technique [12]. It gives high resolution MRI compared to median filter, adaptive filter and spatial filter. The performance of the proposed method is also evaluated by means of peak single - to noise - ratio (PSNR), Average Signal - to - Noise Ratio (ASNR).

3.3 Otsu Method

Otsu's thresholding is a non - linear operation that converts a grayscale image into a binary image where the two levels are assigned to pixel those that are below or above the specified threshold value. The two levels in a binary image are assigned to pixels below or above the particular threshold. It is based on threshold range by statistical calculations. Otsu suggested minimizing the weighted sum of within - class variances of the object and background pixels to establish an optimum threshold.

3.4 DWT & PCA

Discrete Wavelet Transform: The wavelet was used to analyze different frequencies of an image using different scales. Here, we are using discrete wavelet transform (DWT) which is a powerful tool for feature extraction. It was used to extract coefficient of wavelets from brain MR images. The wavelet localizes frequency information of signal function which was important for classification. By using 2D discrete wavelet transform, the images were decomposed into spatial frequency components which were extracted from LL (low–low) sub-bands and since HL(high–low) sub-bands have higher performance when compared to LL (low–low), Department of Computer Engineering Page 6 of 89 we have used both LL (low–low) and HL (high–low) for better analysis which describes image text features.

Principal Component Analysis: The most successful techniques that have been used in image recognition and compression is the Principal Component Analysis (PCA) and it is used to reduce the large dimensionality of the data. The basic approach is to compute the Eigen vectors of the covariance matrix of the original data and approximate it by a linear combination of the leading eigenvectors. By using PCA procedure, the test image can be identified by first, projecting the image onto the Eigen space to obtain the corresponding set of weights, and then comparing with the set of weights of the faces in the training set.

3.5 Support Vector Machine

SVM is one of the classification technique applied on different fields such as face recognition, text categorization, cancer diagnosis, glaucoma diagnosis, microarray gene expression data analysis [15]. SVM utilizes binary classification of brain MR image as normal or tumor affected. SVM divides the given data into decision surface, (i.e. a hyperplane) which divides the data into two classes. The prime objective of SVM is to maximize the margins between two classes of the hyper-plane [16]. Dimensionality reduction and precise feature set given as input to the SVM on the duration of training part as well as during the testing part. SVM is based on binary classifier which employs supervised learning to provide better results.

4. PROPOSED SYSTEM

This chapter discusses the Tumor and the Algorithms used in our proposed system

Features provided

Our system helps in identifying the tumor presence and classifying the tumor is of which type based on three classes and also classifying between benign or malignant.

4.1 Tumor

A brain tumor is a collection, or mass, of abnormal cells in your brain. Your skull, which encloses your brain, is very rigid. Any growth inside such a restricted space can cause problems. Brain tumors can be cancerous (malignant) or noncancerous (benign). When benign or malignant tumors grow, they can cause the pressure inside your skull to increase. This can cause brain damage, and it can be life-threatening.

Brain tumors are categorized as primary or secondary:

- A primary brain tumor originates in your brain. Many primary brain tumors are benign.
- A secondary brain tumor, also known as a metastatic brain tumor, occurs when
- Cancer cells spreadTrusted Source to your brain from another organ, such as your lung or breast.

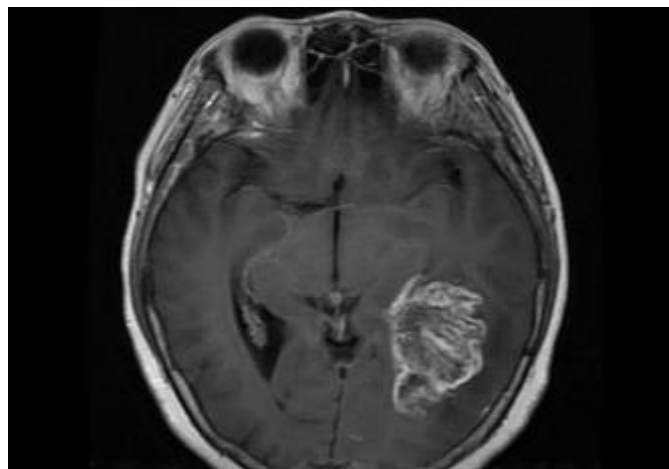


Fig 4.1 Tumor

4.1.1 Benign vs. malignant brain tumors

Though benign brain tumors can cause many serious issues, they are not cancerous, meaning that they grow slowly and don't typically spread to other tissues. They also usually have more clearly defined borders, making them easier to remove surgically, and they don't usually come back after removal. On the other hand, malignant brain tumors are cancerous, grow rapidly, and can spread to other parts of your brain or central nervous system, which can cause life-threatening complications.

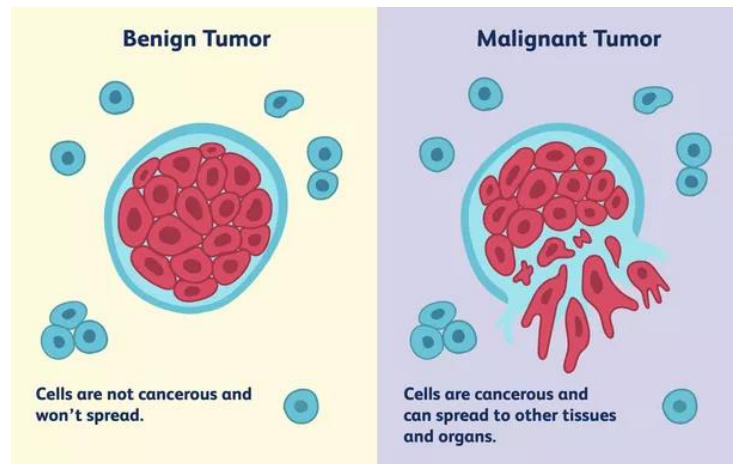


Fig 4.2 Benign vs Malignant

4.1.2 Types of brain tumors

Primary brain tumors

Primary brain tumors originate in your brain. They can develop Trusted Source from

- brain cells
- the membranes that surround your brain, which are called meninges
- nerve cells
- glands, such as the pituitary or pineal

Primary tumors can be benign or cancerous. In adults, the most common types of brain tumors are gliomas and meningiomas.

Gliomas

Gliomas are tumors that develop from glial cells. These cells normally:

- support the structure of your central nervous system

- provide nutrition to your central nervous system
- clean cellular waste
- break down dead neurons

Gliomas can develop from different types of glial cells.

The types of tumors that begin in glial cells include Trusted Source:

- astrocytic tumors, such as astrocytomas, which originate in the cerebrum
- oligodendroglial tumors, which are often found in the frontal temporal lobes
- glioblastomas, which originate in the supportive brain tissue and are the most aggressive type

Other primary brain tumors

Other primary brain tumors include Trusted Source:

- pituitary tumors, which are usually benign
- pineal gland tumors, which can be benign or malignant
- ependymomas, which are usually benign
- craniopharyngiomas, which occur mostly in children and are benign but can have clinical symptoms like changes in vision and premature puberty
- primary central nervous system (CNS) lymphomas, which are malignant
- primary germ cell tumors of the brain, which can be benign or malignant
- meningiomas, which originate in the meninges
- schwannomas, which originate in cells that produce the protective cover of your nerves (myelin sheath) called Schwann cells

Meningiomas are diagnosed Trusted Source more often in women than men, according to a study that grouped participants into men and women. Schwannomas occur Trusted Source equally in both men and women. These tumors are usually benign, but they can cause complications because of their size and location. Cancerous meningiomas and schwannomas are rare but can be very aggressive.

Secondary brain tumors

Secondary brain tumors make up the majority of brain cancers. They start in one part of the body and spread, or metastasize, to the brain.

The following can metastasize Trusted Source to the brain:

- lung cancer
- breast cancer
- kidney cancer
- skin cancer

Secondary brain tumors are always malignant. Benign tumors don't spread from one part of your body to another. In our system we are classifying the tumor in three classes

- Glioma Tumor
- Pituitary Tumor
- Meningioma Tumor

Glioma Tumor

A glioma is a tumor that forms in the brain or spinal cord. There are several types, including astrocytomas, ependymomas and oligodendrogliomas. Gliomas can affect children or adults. Some grow very quickly. Most people with gliomas need a combination of treatments such as surgery, radiation therapy and chemotherapy. Gliomas are malignant (cancerous), but some can be very slow growing. They're primary brain tumors, meaning they originate in the brain tissue. Gliomas don't usually spread outside of the brain or spine, but are life-threatening because they can:

- Be hard to reach and treat with surgery.
- Grow into other areas of the brain.

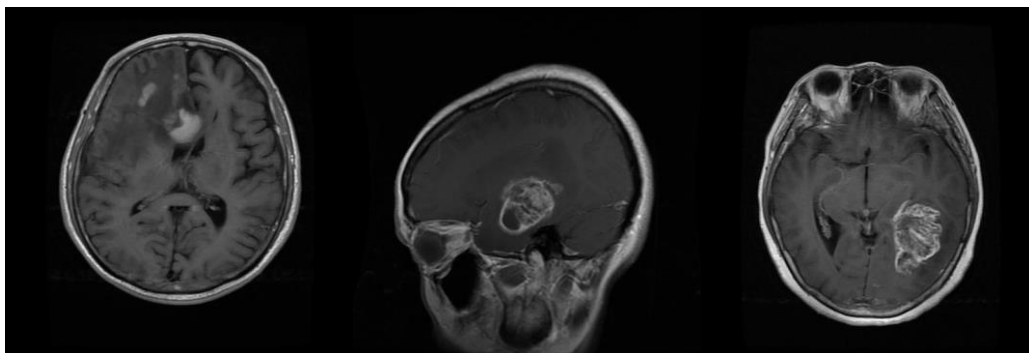


Fig 4.3 Glioma Tumor

The Glioma Tumor is basically in most of the cases it was an Benign type that is it was non-cancerous to our body but it was still cause the damage to the living conditions.

Pituitary Tumor

A pituitary tumor is an abnormal growth in the pituitary gland. The pituitary is a small gland in the brain. It is located behind the back of the nose. It makes hormones that affect many other glands and many functions in your body. Most pituitary tumors are not cancerous (benign). They don't spread to other parts of your body. But they can cause the pituitary to make too few or too many hormones, causing problems in the body. Pituitary tumors that make too many hormones will cause other glands to make more hormones. That will cause symptoms related to each of the specific hormones. Many pituitary tumors will also press against the nearby optic nerves. This can cause vision problems.

Most pituitary tumors don't cause symptoms. As a result, they are not diagnosed. Or they are found only during a routine brain imaging test. About 25% of people may have small pituitary tumors without knowing it.

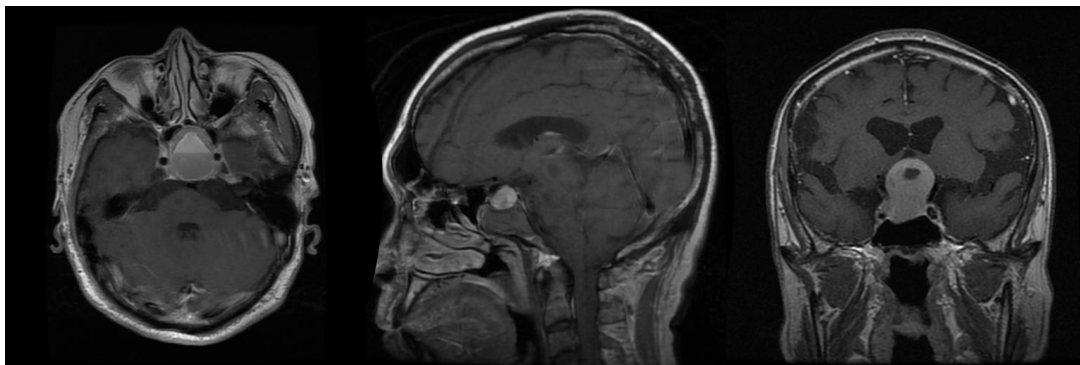


Fig 4.4 Pituitary Tumor

Meningioma Tumor

A meningioma is a tumor that forms on membranes that cover the brain and spinal cord just inside the skull. Specifically, the tumor forms on the three layers of membranes that are called meninges. These tumors are often slow-growing. As many as 90% are benign (not cancerous). Most meningiomas occur in the brain. But they can also grow on parts of the spinal cord. Often, meningiomas cause no symptoms and require no immediate treatment. But the growth of benign meningiomas can cause serious problems. In some cases, such growth can be fatal. Meningiomas are the most common type of tumor that originates in the central nervous system. They occur more often in women than in men. Some meningiomas are classified as atypical. These are not considered either benign or malignant (cancerous). But they may

become malignant. A small number of meningiomas are cancerous. They tend to grow quickly. They also can spread to other parts of the brain and beyond, often to the lungs.

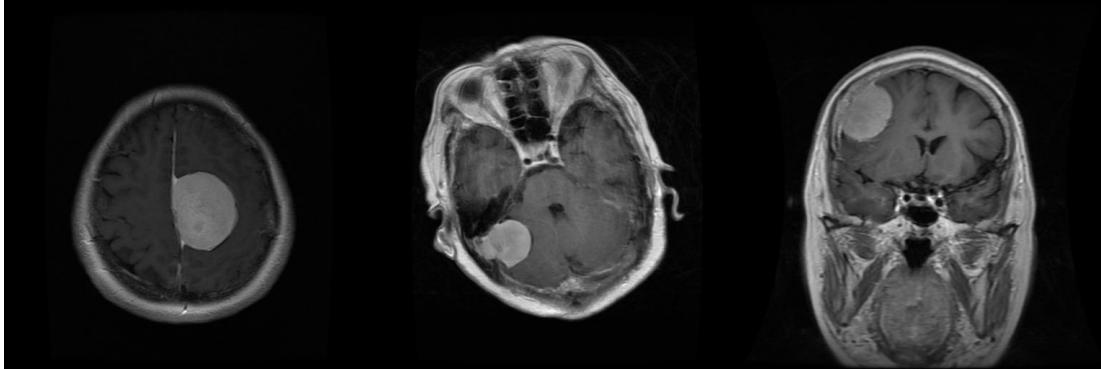


Fig 4.5 Meningioma Tumor

4.2 Software Architecture

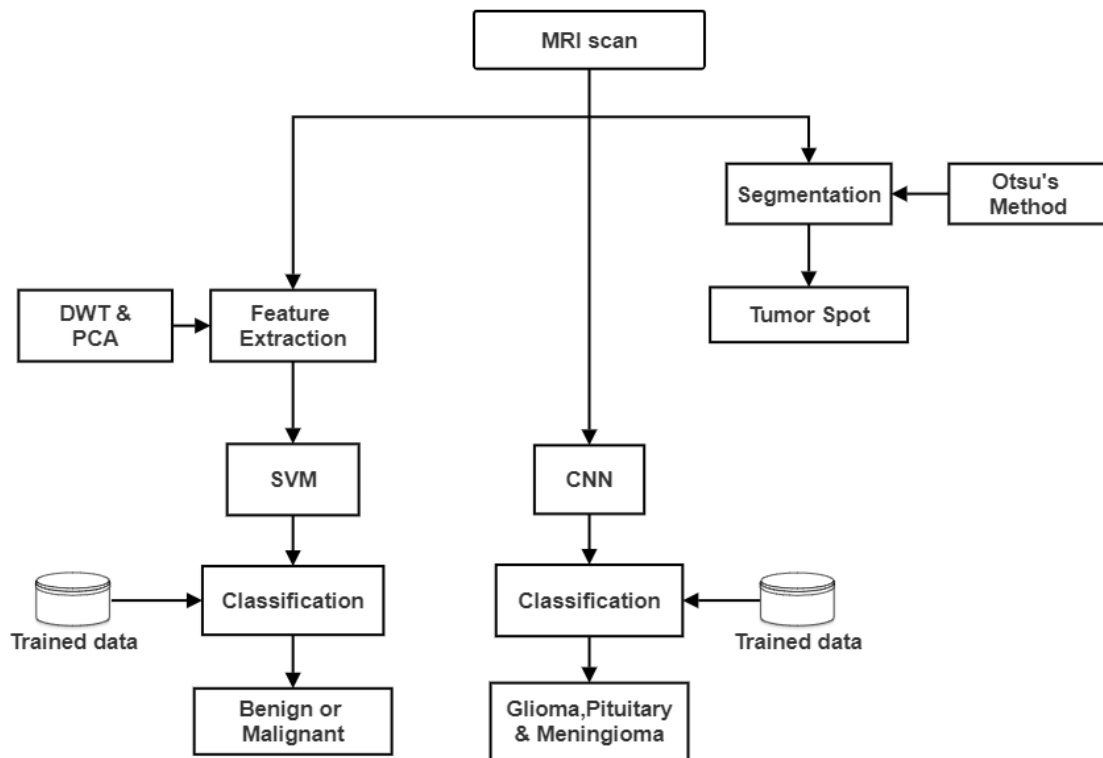


Fig 4.6 Software Architecture

4.3 Segmentation

Image segmentation is the primary step and the most critical tasks of image analysis. Its purpose is that of extracting from an image by means of image segmentation. The mechanization of medical image segmentation has established wide application in diverse areas

such as verdict for patients, treatment management planning, and computer - integrated surgery. The segmentation algorithm that have been implemented:

4.3.1 Thresholding

Thresholding is method used to remove an intended object or target object from its background image by allocating an value of intensity T (threshold) for every pixel such as each pixel is either categorized as an background point or an object point. In various applications of image processing, the gray levels of pixels belonging to the object materially vary from the gray levels of the pixels belonging to the background. Thresholding then becomes a simple but effective tool to separate objects from the background. Examples of thresholding applications are document image analysis, where the goal is to extract printed characters, logos, graphical content, or musical scores: map processing, where lines, legends, and characters are to be found: scene processing, where a target is to be detected: and quality inspection of materials, where defective parts must be delineated. Other applications can be listed as follows: cell images and knowledge representation [2], Otsu's thresholding method implicates iterating through all the credible threshold values and enumerating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that are in foreground or background. The intent is to decide the threshold value where the summation of foreground and background escalates is at its minimum.

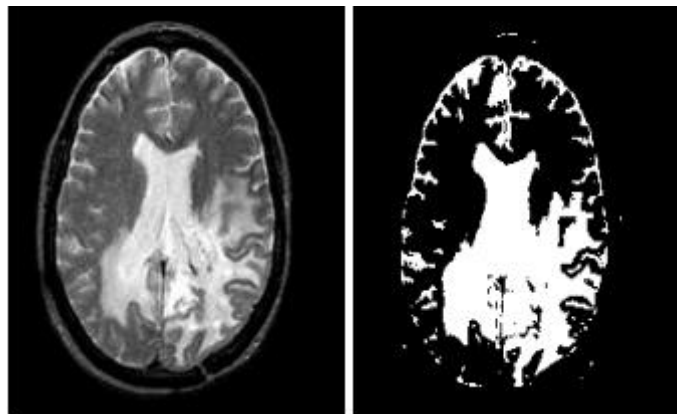


Fig 4.7 Thresholding of brain tumour from grayscale MRI image

Generally intensity is the easiest property which is shared by „pixels“ in particular region. So thresholding is normally used for the segmenting light and dark regions. Here grey level image is converted in to binary image by changing all of the pixels below certain threshold value to ZERO and all above threshold value to ONE. The simplest such abstraction is the process of image thresholding, which can be thought of as an extreme form of gray-level

quantization. Suppose that a gray-level image f can take K possible gray levels $0, 1, 2, \dots, K - 1$. Define an integer threshold, T , that lies in the gray-scale range of T lies between $(0, 1, 2, \dots, K - 1)$. The process of thresholding is a process of simple comparison: each pixel value in f is compared to threshold, T . Based on this comparison; a binary decision is made that defines the value of the corresponding pixel in an output binary image g .

Reviewing Otsu's Method For Image Thresholding

If $g(x, y)$ is a thresholded version of $f(x, y)$ at some global threshold T ,

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) \geq T \\ 0 & \text{otherwise} \end{cases}$$

Algorithm:

Step 1: Compute histogram for a 2D image.

Step 2: Calculate foreground and background variances (measure of spread) for a single threshold.

- i) Calculate weight of background pixels and foreground pixels.
- ii) Calculate mean of background pixels and foreground pixels.
- iii) Calculate variance of background pixels and foreground pixels.

Step 3: Calculate “within class variance”

Logic

Assumptions:

- To simplify the explanation Consider only 6 grayscale levels are used.
- Consider threshold value 3 for 6 grayscale levels.

1) Calculate weight for background and foreground pixels

$$W \text{ (Weight)} = \frac{w_1 + w_2 + \dots + w_n}{\text{Total number of pixels}}$$

2) Calculate mean

Histogram value = h

$$\mu \text{ (Mean)} = (h_1 * w_1) + (h_2 * w_2) + \dots + (h_n * w_n) + (h_n * w_n)$$

Sum of weights (W)

3) Calculate variance

$$V = ((h_1 - \mu)^2 * w_1) + ((h_2 - \mu)^2 * w_2) \dots ((h_n - \mu)^2 * w_n) + ((h_n - \mu)^2 * w_n)$$

Sum of weights (W)

4) Calculate within class variance

I: e sum of two variances multiplied by their associated weights.

$$\text{Within class variance} = W_b * V_b + W_f * V_f$$

Drawbacks of Thresholding

The main problem with this method is the dependency on „intensity“ as no other relationship of pixels is considered. Also pixels at the boundary region and sometimes the shadows of the object create problem. Also noise will affect the thresholding. Illumination in the scene may have some lower or higher brighter parts so global thresholding will not be useful and local thresholding is preferred. Automated thresholding methods can be used for finding thresholds.

Otsu's Thresholding Method

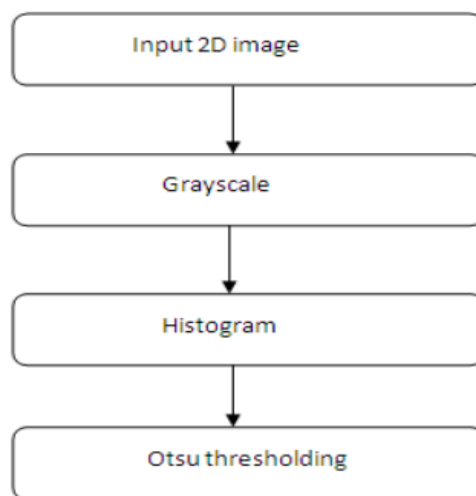


Fig 4.8 Flow of Otsu's thresholding in Image Processing

In optimal thresholding, a “criterion function” is formulated that yields some measure of dissociation between regions. A criterion function is computed for intensity and that which maximizes this function is selected as the threshold. Otsu's thresholding picks the threshold value to minimize the intra-class variance of the thresholded black and white pixels. A specific criterion function is used as a measure of statistical separation for descriptive analysis. First, a criterion for maximizing a modified between-class variance that is equivalent to the criterion

of maximizing the usual between-class variance is proposed for image segmentation. Next, in accordance with the new criterion, a recursive algorithm is designed to efficiently find the optimal threshold. Converting a grayscale image to monochrome is a common image processing task.

Otsu's method instinctively performs clustering-based image thresholding, or, the diminishing of a grayscale image to a binary or threshold image. The algorithm considers that the image contains two classes of pixels backing bi-modal histogram (foreground pixels and background pixels); it then enumerates the optimum threshold disconnecting the two classes so their combined spread (intra-class variance) is minimal. The augmentation of the original method to multi-level thresholding is referred to as the Multi Otsu method. Otsu's method is entitles after its author Nobuyuki Otsu.

An image can be described as a two-dimensional function i:e $f(x, y)$, where x and y are spatial (plane) coordinates, and the amplitude of „ f “ at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point. Each element of the array is called pixel or pel. The Pixels are stored as Integers. The integers can be of 8- bit, 24-bit or 32-bit depending on the type of image. The grayscale images are 8 bit images whereas 32-bit images have an additional transparency channel. The Grayscale image consists of average of R-G-B at each pixel. It is a myth amongst people that grayscale converts colored image to image composed exclusively of shades of grey i.e. it removes color property of image. Operations are performed in some cases on each pixel of an image such as edge detection, thresholding where if there are different values of color at each pixel then it becomes more complicated. So we perform grayscale operation. Histogram equalization is a technique for adjusting image intensities to increase contrast. Through this adjustment, the intensities can be better dispensed on the histogram. This lets areas of lower local contrast to yield a higher contrast. Histogram equalization achieves this by productively expanding the most persistent intensity values. The method is constructive in images with backgrounds and foregrounds that are both bright or both dark. Particularly, the method may lead to superior views of bone structure in x-ray images. Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum. In image processing,

OTSU's thresholding method is used for automatic binarization level decision, based on the shape of the histogram. The algorithm assumes that the image is composed of two basic classes: Foreground and Background. It then computes an optimal threshold value that minimizes the weighted within class variances of these two classes. It is mathematically proven that minimizing the within class variance is same as maximizing the between class variance.

Feature Extraction

Feature extraction is process of extracting quantitative information from an image such as color features, texture, shape and contrast. Here, we have used discrete wavelet transform (DWT) for extracting wavelet coefficients and gray-level co-occurrence matrix (GLCM) for statistical feature extraction.

4.4 Discrete Wavelet Transform

4.4.1 1-D Discrete Wavelet Transform

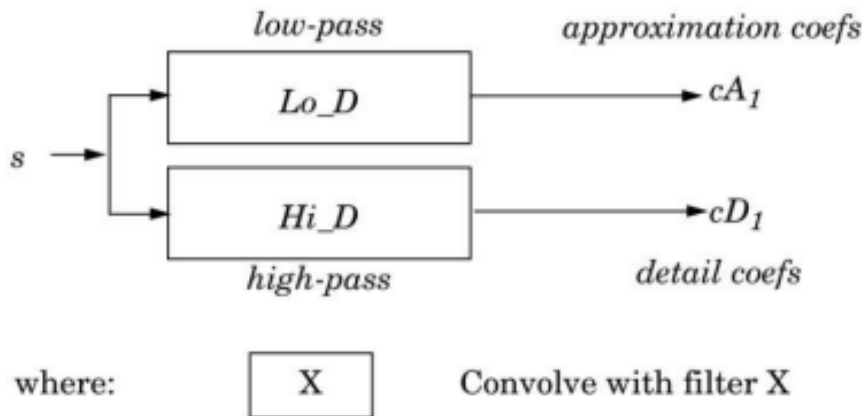


Fig 4.9 1-D DWT

The discrete wavelets transform (DWT), which transforms a discrete time signal to a discrete wavelet representation. The first step is to discretize the wavelet parameters, which reduce the previously continuous basis set of wavelets to a discrete and orthogonal / orthonormal set of basis wavelets.

$$\psi_{m,n}(t) = 2^{-m/2} \psi(2^{-m}t - n); m, n \in \mathbb{Z} \text{ such that } -\infty < m, n < \infty$$

The 1-D DWT is given as the inner product of the signal $x(t)$ being transformed with each of the discrete basis functions.

$$W_{m,n} = \langle x(t), \psi_{m,n}(t) \rangle; m, n \in \mathbb{Z}$$

$$\text{The 1-D inverse DWT is given as: } x(t) = \sum_m \sum_n W_{m,n} \psi_{m,n}(t); m, n \in \mathbb{Z} \dots (2.16)$$

4.4.2 2-D Discrete Wavelet Transform

The 1-D DWT can be extended to 2-D transform using separable wavelet filters. With separable filters, applying a 1-D transform to all the rows of the input and then repeating on all of the columns can compute the 2-D transform. When one-level 2-D DWT is applied to an image, four transform coefficient sets are created. As depicted in figure (c), the four sets are LL, HL, LH, and HH, where the first letter corresponds to applying either a low pass or high pass filter to the rows, and the second letter refers to the filter applied to the columns.

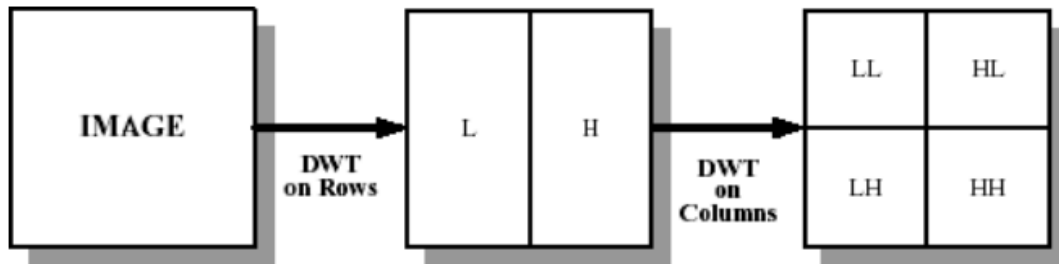


Fig 4.10 Block Diagram of DWT a) Original Image b) Output image after the 1-D applied on Row input (c) Output image after the second 1-D applied on row input



Fig 4.11 DWT for Lena image (a) Original Image (b) Output image after the 1-D applied on column input (c) Output image after the second 1-D applied on row input

The Two-Dimensional DWT (2D-DWT) converts images from spatial domain to frequency domain. At each level of the wavelet decomposition, each column of an image is first transformed using a 1D vertical analysis filter-bank. The same filter-bank is then applied horizontally to each row of the filtered and sub-sampled data. One-level of wavelet decomposition produces four filtered and sub-sampled images, referred to as sub-bands.

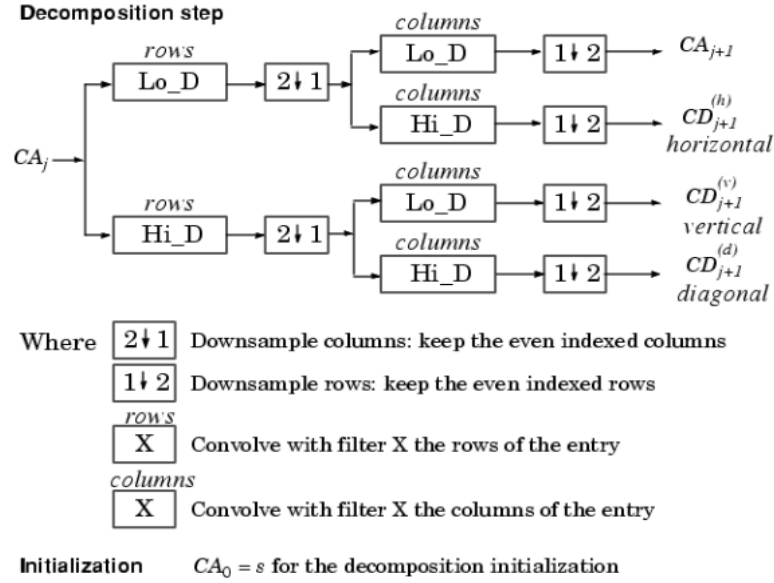


Fig 4.12 2-D DWT

The upper and lower areas of figure (b), respectively, represent the low pass and high pass coefficients after vertical 1D-DWT and sub-sampling. The result of the horizontal 1D-DWT and sub-sampling to form a 2D-DWT output image is shown in figure (c). We can use multiple levels of wavelet transforms to concentrate data energy in the lowest sampled bands. Specifically, the LL sub band in figure 2.11 (c) can be transformed again to form LL2, HL2, LH2, and HH2 sub-bands, producing a two-level wavelet transform. An (R-1) level wavelet decomposition is associated with R resolution levels numbered from 0 to (R-1), with 0 and (R-1) corresponding to the coarsest and finest resolutions. The straight forward convolution implementation of 1D-DWT requires a large amount of memory and large computation complexity. An alternative implementation of the 1DDWT, known as the lifting scheme, provides significant reduction in the memory and the computation complexity. Lifting also allows in-place computation of the wavelet coefficients. Nevertheless, the lifting approach computes the same coefficients as the direct filter-bank convolution.

Lifting Scheme

Wavelet decompositions are widely used in signal and image processing applications. Classical linear wavelet transforms perform homogeneous smoothing of signal contents. In a number of cases, in particular in applications in image and video processing, such homogeneous smoothing is undesirable. This has led to a growing interest in nonlinear wavelet representations called as adaptive wavelet decomposition that can preserve discontinuities such as transitions in signals and edges in images. Adaptive wavelet decomposition is very useful in

various applications, such as image analysis, compression, and feature extraction and denoising. The adaptive multi resolution representations take into account the characteristics of the underlying signal and do leave intact important signal characteristics, such as sharp transitions, edges, singularities, and other region of interests.

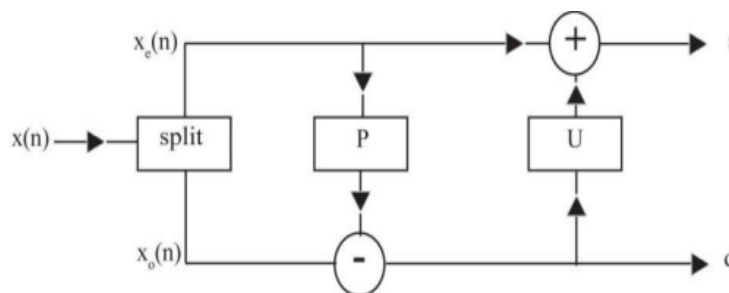
In the adaptive update lifting framework, the update lifting step, which yields a low pass filter or a moving average process, is performed first on the input polyphase components (the output from the splitting process, according to the lifting terminology), followed by a fixed prediction step yielding the wavelet coefficients. In this research, the adaptivity is done.

Combining the norms technique, in this way only homogeneous regions are smoothed while discontinuities are, more or less, preserved. Since the update lifting step represents an averaging process, this framework deviates from the unit diagonal elementary matrix representation associated with the classical lifting framework.

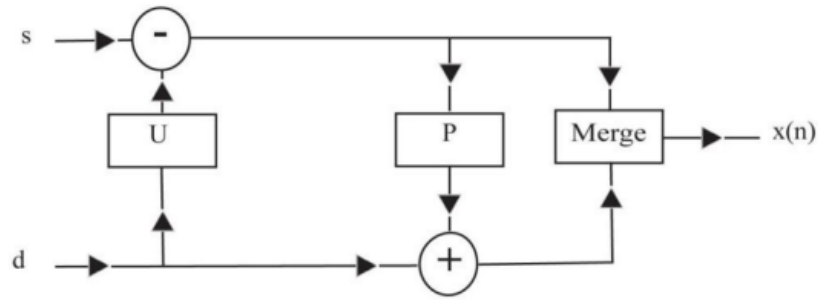
The wavelet transform of image is implemented using the lifting scheme. The lifting operation consists of three steps. First, the input signal $x[n]$ is down sampled into the even position signal $x_e(n)$ and the odd position signal $x_o(n)$, then modifying these values using alternating prediction and updating steps.

$$x_e(n) = x[2n] \text{ and } x_o(n) = x[2n+1] \dots \dots \dots (2.17)$$

A prediction step consists of predicting each odd sample as a linear combination of the even samples and subtracting it from the odd sample to form the prediction error. An update step consists of updating the even samples by adding them to a linear combination of the prediction error to form the updated sequence. The prediction and update may be evaluated in several steps until the forward transform is completed.



A) Forward Transform



B) Inverse Transform

Fig 4.13 Forward & Inverse Transform

The inverse transform is similar to forward. It is based on the three operations undo update, undo prediction, and merge. The averages (even elements) become the input for the next recursive step of the forward transform. This is shown in figure below.

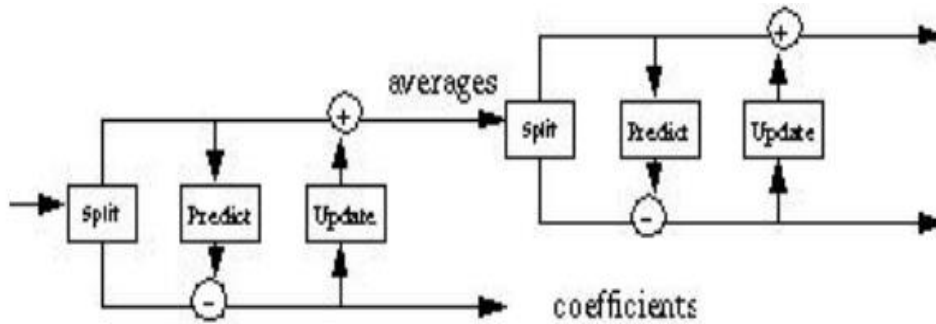


Fig 4.14 Lifting Scheme

Lifting Using Haar

The lifting scheme is a useful way of looking at discrete wavelet transform. It is easy to understand, since it performs all operations in the time domain, rather than in the frequency domain, and has other advantages as well. This section illustrates the lifting approach using the Haar Transform.

The Haar transform is based on the calculations of the averages (approximation coefficient) and differences (detail co-efficient). Given two adjacent pixels a and b , the principle is to calculate the average $s = (a + b) / 2$ and the difference is $d = a - b$. If a and b are similar, s will be similar to both and d will be small, i.e., require few bits to represent. This transform is reversible, since $a = s - d / 2$ and $b = s + d / 2$ it can be written using matrix

$$(s, d) = (a, b) \begin{pmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{pmatrix} = (a, b) A \dots\dots\dots (2.18)$$

$$(a,b)=(s,d)\begin{pmatrix} 1 & 1 \\ -1/2 & 1/2 \end{pmatrix}=(a,b)A^{-1}.....(2.19)$$

Consider a row of 2^n pixels values $S_{n,l}$ for $0 \leq l < 2^n$. There are 2^{n-1} pairs of pixels $s_{n,2l}, s_{n,2l+1}$ for $l = 0, 2, 4, \dots, 2^{n-2}$. Each pair is transformed into an average $s_{n,2l} = (s_{n,2l} + s_{n,2l+1})/2$ and the difference $d_{n-1,l} = s_{n,2l+1} - s_{n,2l}$. The result is a set S of 2^{n-1} averages and a set d of 2^{n-1} differences.

Generally wavelet domain allows us to hide data in regions that the human visual system (HVS) is less sensitive to, such as the high resolution detail bands (HL, LH and HH), Hiding data in these regions allow us to increase the robustness while maintaining good visual quality. Integer wavelet transform maps an integer data set into another integer data set. In discrete wavelet transform, the used wavelet filters have floating point coefficients so that when we hide data in their coefficients any truncations of the floating point values of the pixels that should be integers may cause the loss of the hidden information which may lead to the failure of the data hiding system. To avoid problems of floating point precision of the wavelet filters when the input data is integer as in digital images, the output data will no longer be integer which doesn't allow perfect reconstruction of the input image and in this case there will be no loss of information through forward and inverse transform.

Due to the mentioned difference between integer wavelet transform (IWT) and discrete wavelet transform (DWT) the LL sub band in the case of IWT appears to be a close copy with smaller scale of the original image while in the case of DWT the resulting LL sub band is distorted as shown in figure. Lifting schemes is one of many techniques that can be used to perform integer wavelet transform it is also the scheme used in this research. The Haar wavelet transform can be written as simple pair wise averages and differences:

$$S_{1,n} = \frac{(S_{n,2l+1} + S_{n,2n+1})}{2} \dots\dots\dots(2.20)$$

$$d_{1,n} = S_{n,2n+1} - S_{n,2n} \dots\dots\dots(2.21)$$

Consider a row of 2^n pixels values $S_{n,l}$ for $0 \leq l < 2^n$. There are 2^{n-1} pairs of pixels $s_{n,2l}, s_{n,2l+1}$ for $l = 0, 2, 4, \dots, 2^{n-2}$. Each pair is transformed into an average $s_{n,2l} = (s_{n,2l} + s_{n,2l+1})/2$ and the difference $d_{n-1,l} = s_{n,2l+1} - s_{n,2l}$. The result is a set S of 2^{n-1} averages and a set d of 2^{n-1} differences.

4.5 Principal Component Analysis (pca) on raw data.

$\text{COEFF} = \text{pca}(X)$ returns the principal component coefficients for the N by P data matrix X. Rows of X correspond to observations and columns to variables. Each column of COEFF contains coefficients for one principal component. The columns are in descending order in terms of component variance (LATENT). pca , by default, centers the data and uses the singular value decomposition algorithm. For the non-default options, use the name/value pair arguments. The extracted features are:

1. Correlation

It measures the linear dependency of grey levels of neighbour pixels. It is defined as

$$\text{Correlation} = \sum_{i,j=0}^{N-1} P_{ij} \frac{(i - \mu)(j - \mu)}{\sigma^2}$$

2. Contrast

Also called the sum of Square Variance. It defers the calculation of the intensity contrast linking pixel and its neighbour over the whole image. It is defined as

$$\text{Contrast} = \sum_{i,j=0}^{N-1} P_{ij} (i - j)^2$$

3. Energy

It makes use for the texture that calculates orders in an image. It gives the sum of square elements in GLCM. It is defined as

$$\text{Energy} = \sum_{i,j=0}^{N-1} (P_{ij})^2$$

4. Homogeneity

It passes the value that calculates the tightness of distribution of the elements in the GLCM to the GLCM diagonal. It is defined as

$$\text{Homogeneity} = \sum_{i,j=0}^{N-1} \frac{P_{ij}}{1 + (i - j)^2}$$

5. Mean

Defined as the mean of the pixel values of the input image. It is defined as

$$Mean = \sum_{i,j=0}^{N-1} i(P_{i,j})$$

6. Standard Deviation

It is defined as the dispersion of the pixel in consideration from the mean of the pixels of the input image. It is defined as

$$Standard\ Deviation = \sqrt{\sigma_i^2}$$

7. Entropy

It shows the amount of information of the image that is needed for the image compression. Entropy measures the loss of information or message in a transmitted signal and also measures the image information. It is defined as

$$Entropy = \sum_{i,j=0}^{N-1} -\ln(P_{ij})P_{ij}$$

8. Root Mean Square (RMS)

RMS is calculated on a set of pixels by taking the square of each pixel, calculating the sum of those squares, and taking the square root. The result is scaled by the number of pixels. RMS gives an accurate measurement of the amount of noise present. It is defined as

$$RMS\ noise = \sqrt{\frac{\sum_{i=1}^n (X_i - \frac{\sum_{i=1}^n X_i}{n})^2}{n}}$$

9. Variance

It is the expectation of the square deviation of a pixel from its mean. It is defined as

$$Variance = \sum_{i,j=0}^{N-1} P_{i,j} (i - \mu_i)^2$$

10. Inverse Difference Movement (IDM)

Inverse Difference Moment (IDM) is the local homogeneity. It is high when local gray level is uniform and inverse GLCM is high. It is defined as

$$IDM = \frac{\sum_{i=0}^{Ng-1} \sum_{j=0}^{Ng-1} P_{ij}}{1+(i-j)^2}$$

4.6 Classification

4.6.1 Support Vector Machine (SVM)

Support vector machines (SVMs) are a type of supervised learning models along with associated learning algorithms that analyze data and recognize various patterns, used for classification analysis. The basic SVM takes a set of input data and predicts, for each given input, which of two possible classes, malignant and benign forms the output, making it a non-probabilistic binary linear classifier. Now that there are set of training examples at hand, each marked as belonging to one of two categories, an SVM training algorithm constructs a model that assigns new examples into one category or the other. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. Newer examples are then plotted into it and then predicted to belong to a category based on which side of the gap they fall on. More formally, a support vector machine constructs a hyper plane or set of hyper planes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyper plane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. A SVM takes a set of feature vectors as input, generates a training model after scaling, selecting and validating, and generates a training model as the output. The following figure represents the training process of a SVM:

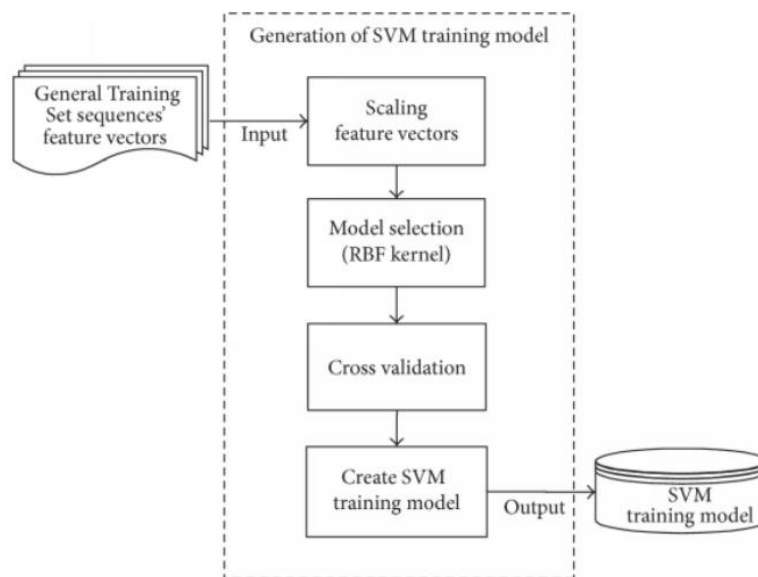


Fig 4.15 Training a SVM Model

Firstly working with neural networks for supervised and unsupervised learning showed good results while used for such learning applications. MLP's uses feed forward and recurrent networks. Multilayer perceptron (MLP) properties include universal approximation of continuous nonlinear functions and include learning with input-output patterns and also involve advanced network architectures with multiple inputs and outputs.

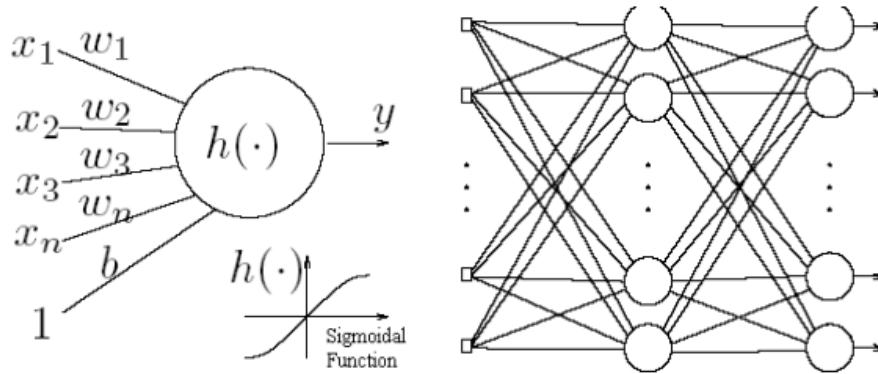


Fig 4.16 Simple Neural Network, Multilayer Perceptron.

There can be some issues noticed. Some of them are having many local minima and also finding how many neurons might be needed for a task is another issue which determines whether optimality of that NN is reached. Another thing to note is that even if the neural network solutions used tends to converge, this may not result in a unique solution. Now let us look at another example where we plot the data and try to classify it and we see that there are many hyper planes which can classify it. But which one is better?

Here we see that there are many hyper planes which can be fit in to classify the data but which one is the best is the right or correct solution. From above illustration, there are many linear classifiers (hyper planes) that separate the data. However only one of these achieves maximum separation. The reason we need it is because if we use a hyper plane to classify, it might end up closer to one set of datasets compared to others and we do not want this to happen and thus we see that the concept of maximum margin classifier or hyper plane as an apparent solution. The next illustration gives the maximum margin classifier example which provides a solution to the above mentioned problem.

The below illustration is the maximum linear classifier with the maximum range. In this context it is an example of a simple linear SVM classifier. Another interesting question is why maximum margin? There are some good explanations which include better empirical performance. Another reason is that even if we've made a small error in the location of the boundary this gives us least chance of causing a misclassification. The other advantage would

be avoiding local minima and better classification. Now we try to express the SVM mathematically and for this tutorial we try to present a linear SVM. The goals of SVM are separating the data with hyper plane and extend this to non-linear boundaries using kernel trick.

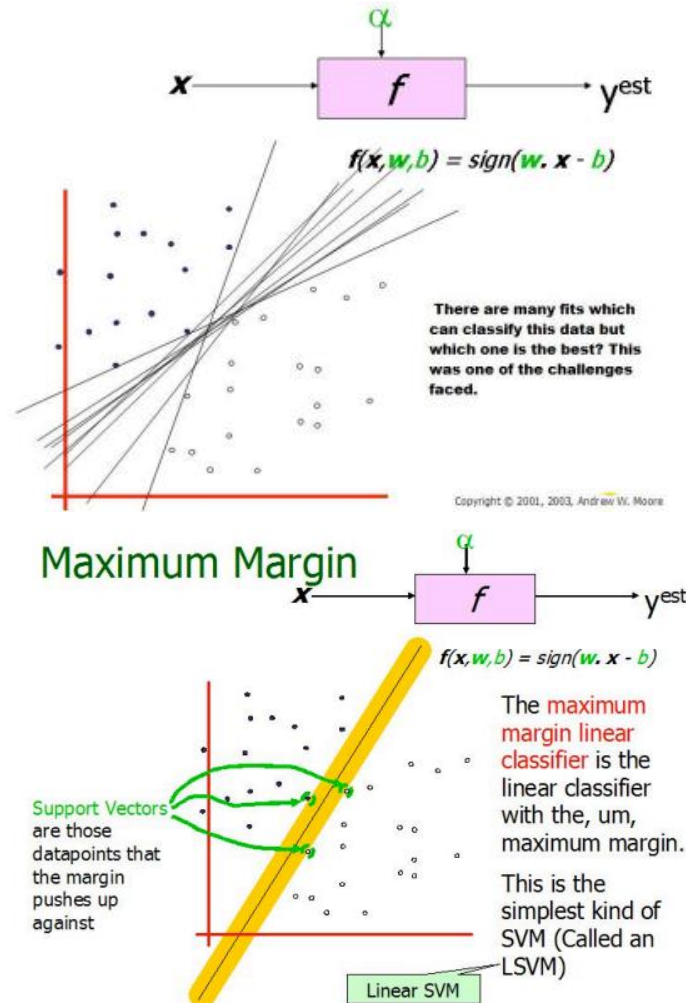


Fig 4.17 Illustration of Linear SVM

Expression for Maximum margin is given as

$$\text{margin} = \arg \min_{x \in D} d(x) = \arg \min_{x \in D} \frac{|x \cdot w + b|}{\sqrt{\sum_{i=1}^d w_i^2}}$$

For calculating the SVM we see that the goal is to correctly classify all the data. For mathematical calculations we have,

- [a] If $Y_i = +1$; $w x_i + b \geq 1$
- [b] If $Y_i = -1$; $w x_i + b \leq -1$
- [c] For all i ; $y_i (w x_i + b) \geq 1$

In this equation x is a vector point and w is weight and is also a vector. So to separate the data [a] should always be greater than zero. Among all possible hyper planes, SVM selects the one where the distance of hyper plane is as large as possible. If the training data is good and every test vector is located in radius r from training vector. Now if the chosen hyper plane is located at the farthest possible from the data [12]. This desired hyper plane which maximizes the margin also bisects the lines between closest points on convex hull of the two datasets. Thus we have [a], [b] & [c].

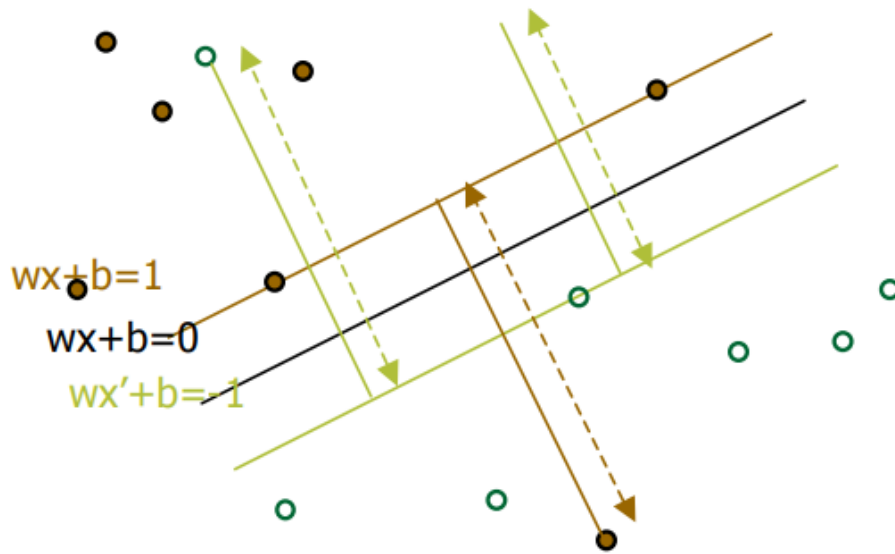


Fig 4.18 Representation of Hyper planes.

Distance of closest point on hyperplane to origin can be found by maximizing the x as x is on the hyper plane. Similarly for the other side points we have a similar scenario. Thus solving and subtracting the two distances we get the summed distance from the separating hyperplane to nearest points. Maximum Margin = $M = 2 / \|w\|$

Now maximizing the margin is same as minimum [8]. Now we have a quadratic optimization problem and we need to solve for w and b . To solve this we need to optimize the quadratic function with linear constraints. The solution involves constructing a dual problem and where a Lagrange's multiplier α_i is associated. We need to find w and b such that $\Phi(w) = \frac{1}{2} \|w'\|w$ is minimized;

And for all $\{(x_i, y_i)\}$: $y_i (w * x_i + b) \geq 1$.

Now solving: we get that $w = \sum \alpha_i * x_i$; $b = y_k - w * x_k$ for any x_k such that $\alpha_k \neq 0$

Now the classifying function will have the following form: $f(x) = \sum \alpha_i y_i x_i * x + b$

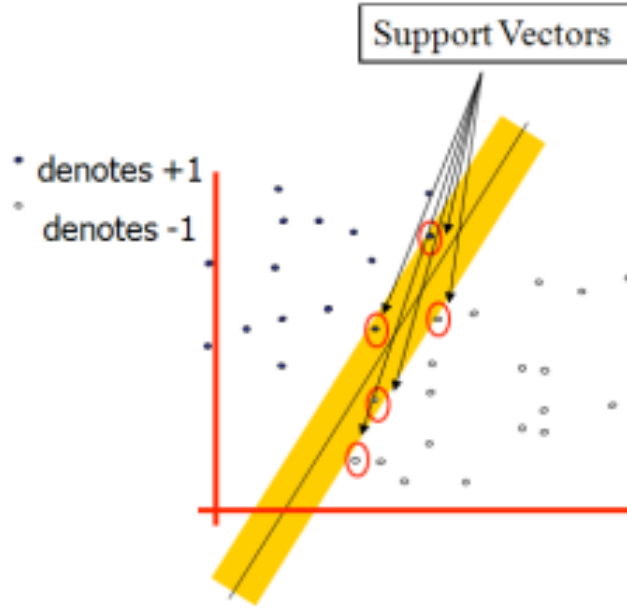


Fig 4.19 Representation of Support Vectors

SVM Representation

In this we present the QP formulation for SVM classification [4][8][12][13]. This is a simple representation only.

SV classification:

$$\min_{f, \xi_i} \|f\|_K^2 + C \sum_{i=1}^l \xi_i \quad y_i f(\mathbf{x}_i) \geq 1 - \xi_i, \text{ for all } i \quad \xi_i \geq 0$$

SVM classification, Dual formulation:

$$\min_{\alpha_i} \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad 0 \leq \alpha_i \leq C, \text{ for all } i; \quad \sum_{i=1}^l \alpha_i y_i = 0$$

Variables ξ_i are called slack variables and they measure the error made at point (x_i, y_i) . Training SVM becomes quite challenging when the number of training points is large. A number of methods for fast SVM training have been proposed.

Soft Margin Classifier

In real world problem it is not likely to get an exactly separate line dividing the data within the space. And we might have a curved decision boundary. We might have a hyperplane which might exactly separate the data but this may not be desirable if the data has noise in it. It is better for the smooth boundary to ignore few data points than be curved or go in loops, around the outliers. This is handled in a different way; here we hear the term slack variables

being introduced. Now we have, $y_i(w'x + b) \geq 1 - S_k$ [4] [12]. This allows a point to be a small distance S_k on the wrong side of the hyper plane without violating the constraint. Now we might end up having huge slack variables which allow any line to separate the data, thus in such scenarios we have the Lagrangian variable introduced which penalizes the large slacks.

$$\min L = \frac{1}{2} w'w - \sum \lambda_k (y_k (w'x_k + b) + s_k - 1) + \alpha \sum s_k$$

Where reducing α allows more data to lie on the wrong side of hyper plane and would be treated as outliers which give smoother decision boundary.

Kernel Trick

Let's first look at what is a kernel and what does feature space mean?

Kernel:

If data is linear, a separating hyper plane may be used to divide the data. However it is often the case that the data is far from linear and the datasets are inseparable. To allow for this kernels are used to non-linearly map the input data to a high-dimensional space. The new mapping is then linearly separable [1]. A very simple illustration of this is shown below.

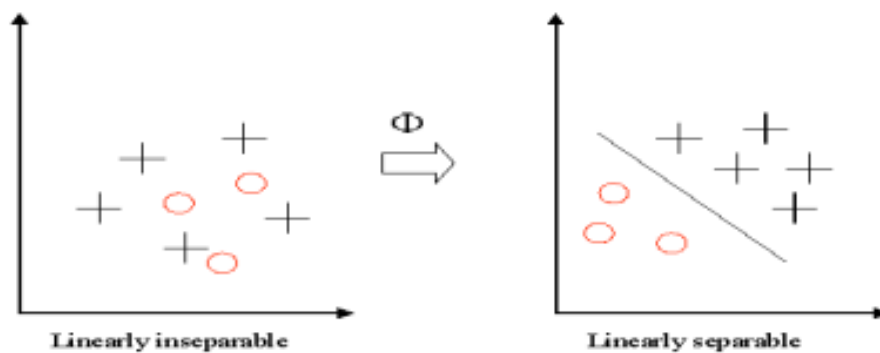


Fig 4.20 Kernels

This mapping is defined by the Kernel: $K(x,y) = \Phi(x) \cdot \Phi(y)$

Feature Space: Transforming the data into feature space makes it possible to define a similarity measure on the basis of the dot product.

If the feature space is chosen suitably, pattern recognition can be easy.

$$(x1 \cdot x2) \leftarrow K(x1, x2) = (\Phi(x1) \cdot \Phi(x2))$$

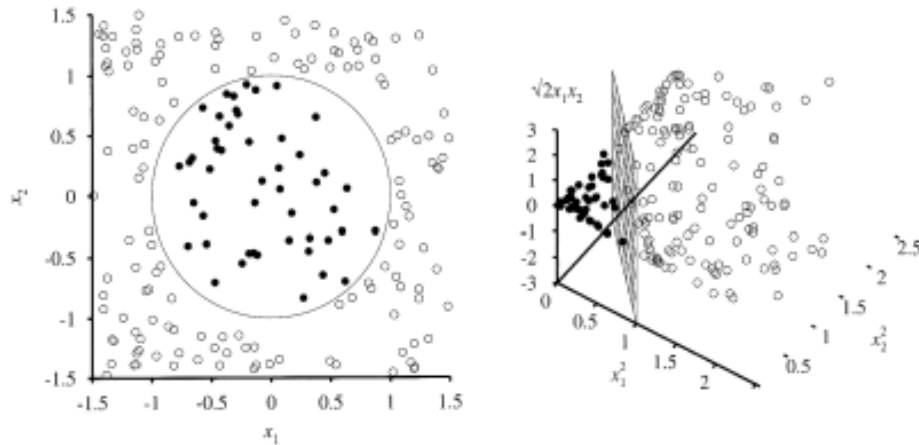


Fig 4.21 Feature Space Representation

Now getting back to the kernel trick, we see that when w, b is obtained the problem is solved for a simple linear scenario in which data is separated by a hyper plane. The Kernel trick allows SVM's to form nonlinear boundaries. Steps involved in kernel trick are given below.

[a] The algorithm is expressed using only the inner products of data sets. This is also called as dual problem.

[b] Original data are passed through non-linear maps to form new data with respect to new dimensions by adding a pair wise product of some of the original data dimension to each data vector.

[c] Rather than an inner product on these new, larger vectors, and store in tables and later do a table lookup, we can represent a dot product of the data after doing non-linear mapping on them. This function is the kernel function. More on kernel functions is given below

Kernel Trick: Dual Problem

First we convert the problem with optimization to the dual form in which we try to eliminate w , and a Lagrangian now is only a function of λ_i . There is a mathematical solution for it but this can be avoided here as this tutorial has instructions to minimize the mathematical equations, I would describe it instead. To solve the problem we should maximize the LD with respect to λ_i . The dual form simplifies the optimization and we see that the major achievement is the dot product obtained

Kernel Trick: Inner Product summarization

Here we see that we need to represent the dot product of the data vectors used. The dot

product of nonlinearly mapped data can be expensive. The kernel trick just picks a suitable function that corresponds to dot product of some nonlinear mapping instead. Some of the most commonly chosen kernel functions are given below in later part of this tutorial. A particular kernel is only chosen by trial and error on the test set, choosing the right kernel based on the problem or application would enhance SVM's performance.

Kernel Functions

The idea of the kernel function is to enable operations to be performed in the input space rather than the potentially high dimensional feature space. Hence the inner product does not need to be evaluated in the feature space. We want the function to perform mapping of the attributes of the input space to the feature space. The kernel function plays a critical role in SVM and its performance. It is based upon reproducing Kernel Hilbert Spaces.

$$K(x, x') = (\Phi(x), \Phi(x')),$$

If K is a symmetric positive definite function, which satisfies Mercer's Conditions,

$$K(x, x') = \sum_m^\infty a_m \phi_m(x) \phi_m(x'), a_m \geq 0,$$

$$\iint k(x, x') g(x) g(x') dx dx' > 0, g \in L_2$$

Then the kernel represents a legitimate inner product in feature space. The training set is not linearly separable in an input space. The training set is linearly separable in the feature space. This is called the "Kernel trick". The different kernel functions are listed below [8]: More explanation on kernel functions can be found in the book [8]. The below mentioned ones are extracted from there and just for mentioning purposes are listed below.

1] Polynomial: A polynomial mapping is a popular method for non-linear modeling. The second kernel is usually preferable as it avoids problems with the hessian becoming Zero.

$$k(x, x') = (x, x')^d.$$

$$k(x, x') = (x, x' + 1)^d.$$

2] Gaussian Radial Basis Function: Radial basis functions most commonly with a Gaussian form

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

3] Exponential Radial Basis Function: A radial basis function produces a piecewise linear solution which can be attractive when discontinuities are acceptable.

$$k(x, x^1) = \exp\left(-\frac{\|x - x'\|}{2\sigma^2}\right)$$

4] Multi-Layer Perceptron: The long established MLP, with a single hidden layer, also has a valid kernel representation

$$k(x, x') = \tanh(\rho(x, x^1) + e)$$

There are many more including Fourier, splines, B-splines, additive kernels and tensor products.

Controlling Complexity in SVM: Trade-offs

SVM is powerful to approximate any training data and generalizes better on given datasets. The complexity in terms of kernel affects the performance on new datasets [8]. SVM supports parameters for controlling the complexity and above all SVM does not tell us how to set these parameters and we should be able to determine these Parameters by Cross-Validation on the given datasets. The diagram given below gives a better illustration.

SVM for Classification SVM is a useful technique for data classification. Even though it's considered that Neural Networks are easier to use than this, however, sometimes unsatisfactory results are obtained. A classification task usually involves with training and testing data which consist of some data instances. Each instance in the training set contains one target values and several attributes. The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes.

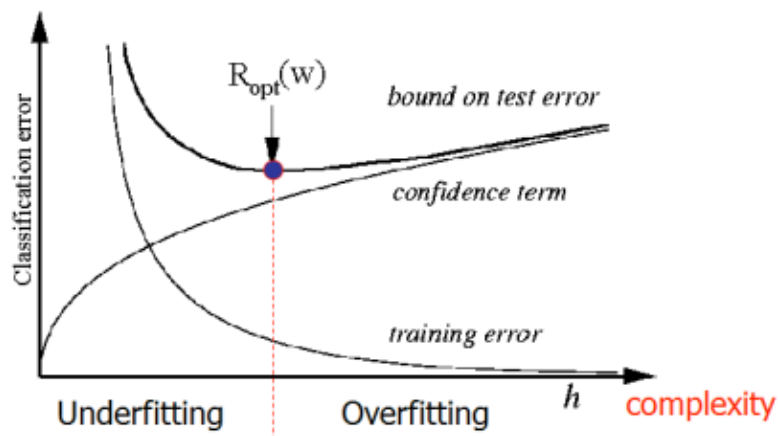


Fig 4.22 How to control complexity

Classification in SVM is an example of Supervised Learning. Known labels help indicate whether the system is performing in a right way or not. This information points to a

desired response, validating the accuracy of the system, or be used to help the system learn to act correctly. A step in SVM classification involves identification as which are intimately connected to the known classes. This is called feature selection or feature extraction. Feature selection and SVM classification together have a use even when prediction of unknown samples is not necessary. They can be used to identify key sets which are involved in whatever processes distinguish the classes.

SVM for Regression

SVMs can also be applied to regression problems by the introduction of an alternative loss function . The loss function must be modified to include a distance measure. The regression can be linear and non linear. Linear models mainly consist of the following loss functions, e-intensive loss functions, quadratic and Huber loss function. Similarly to classification problems, a non-linear model is usually required to adequately model data. In the same manner as the non-linear SVC approach, a non-linear mapping can be used to map the data into a high dimensional feature space where linear regression is performed. The kernel approach is again employed to address the curse of dimensionality. In the regression method there are considerations based on prior knowledge of the problem and the distribution of the noise. In the absence of such information Huber's robust loss function, has been shown to be a good alternative.

Deep Learning

Deep learning is a type of machine learning in which a model learns to perform classification tasks directly from images, text, or sound. Deep learning is usually implemented using a neural network architecture. The term “deep” refers to the number of layers in the network—the more layers, the deeper the network. Traditional neural networks contain only 2 or 3 layers, while deep networks can have hundreds.

How A Deep Neural Network Learns

Let's say we have a set of images where each image contains one of four different categories of object, and we want the deep learning network to automatically recognize which object is in each image. 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. Notice that the network learns directly from the data—we have no influence over what features are being learned.

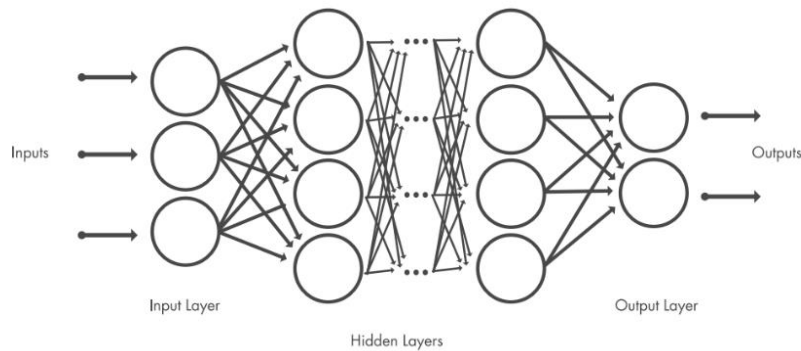


Fig 4.23 Deep Learning Layers

4.6.2 Convolutional Neural Networks

A convolutional neural network (CNN, or ConvNet) is one of the most popular algorithms for deep learning with images and video. Like other neural networks, a CNN is composed of an input layer, an output layer, and many hidden layers in between.

Feature Detection Layers

These layers perform one of three types of operations on the data: convolution, pooling, or rectified linear unit (ReLU).

Convolution puts the input images through a set of convolutional filters, each of which activates certain features from the images.

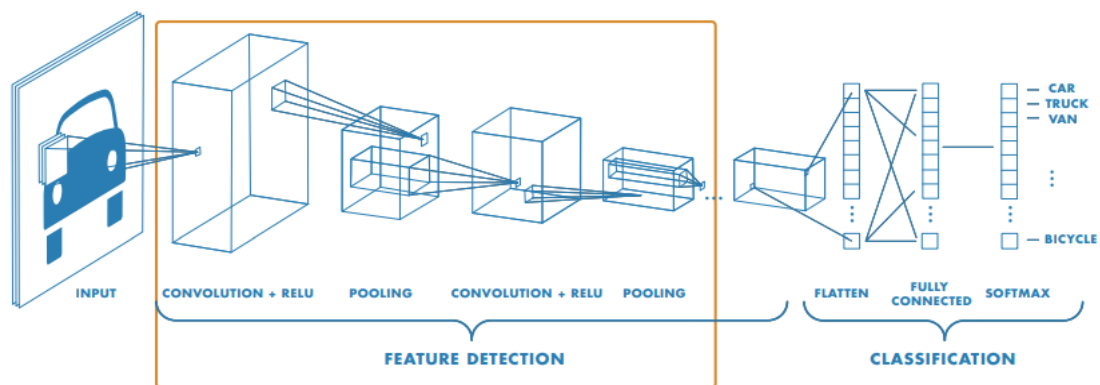


Fig 4.24 CNN

Pooling simplifies the output by performing nonlinear down sampling, reducing the number of parameters that the network needs to learn about.

Rectified linear unit (ReLU) allows for faster and more effective training by mapping negative values to zero and maintaining positive values. These three operations are repeated over tens or hundreds of layers, with each layer learning to detect different features.

Classification Layers

After feature detection, the architecture of a CNN shifts to classification.

The next-to-last layer is a fully connected layer (FC) that outputs a vector of K dimensions where K is the number of classes that the network will be able to predict. This vector contains the probabilities for each class of any image being classified. The final layer of the CNN architecture uses a softmax function to provide the classification output.

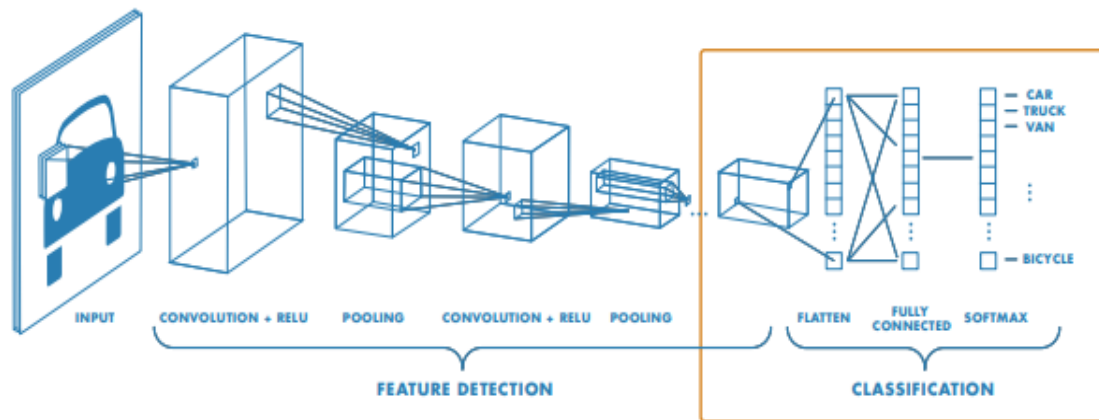


Fig 4.25 Classification Layer

4.7 Transfer Learning

Transfer learning is the reuse of a pre-trained model on a new problem. It's currently very popular in deep learning because it can train deep neural networks with comparatively little data. This is very useful in the data science field since most real-world problems typically do not have millions of labeled data points to train such complex models. We'll take a look at what transfer learning is, how it works, why and when you it should be used. Additionally, we'll cover the different approaches of transfer learning and provide you with some resources on already pre-trained models. Transfer learning, used in machine learning, is the reuse of a pre-trained model on a new problem. In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another. For example, in training a classifier to predict whether an image contains food, you could use the knowledge it gained during training to recognize drinks.

In transfer learning, the knowledge of an already trained machine learning model is applied to a different but related problem. For example, if you trained a simple classifier to predict whether an image contains a backpack, you could use the knowledge that the model gained during its training to recognize other objects like sunglasses. With transfer learning, we basically try to exploit what has been learned in one task to improve generalization in another. We transfer the weights that a network has learned at "task A" to a new "task B."

4.7.1 How Transfer Learning Works

In computer vision, for example, neural networks usually try to detect edges in the earlier layers, shapes in the middle layer and some task-specific features in the later layers. In transfer learning, the early and middle layers are used and we only retrain the latter layers. It helps leverage the labeled data of the task it was initially trained on.

Let's go back to the example of a model trained for recognizing a backpack on an image, which will be used to identify sunglasses. In the earlier layers, the model has learned to recognize objects, because of that we will only retrain the latter layers so it will learn what separates sunglasses from other objects. This approach is mostly used in computer vision because it can reduce the size of your dataset, which decreases computation time and makes it more suitable for traditional algorithms, as well.

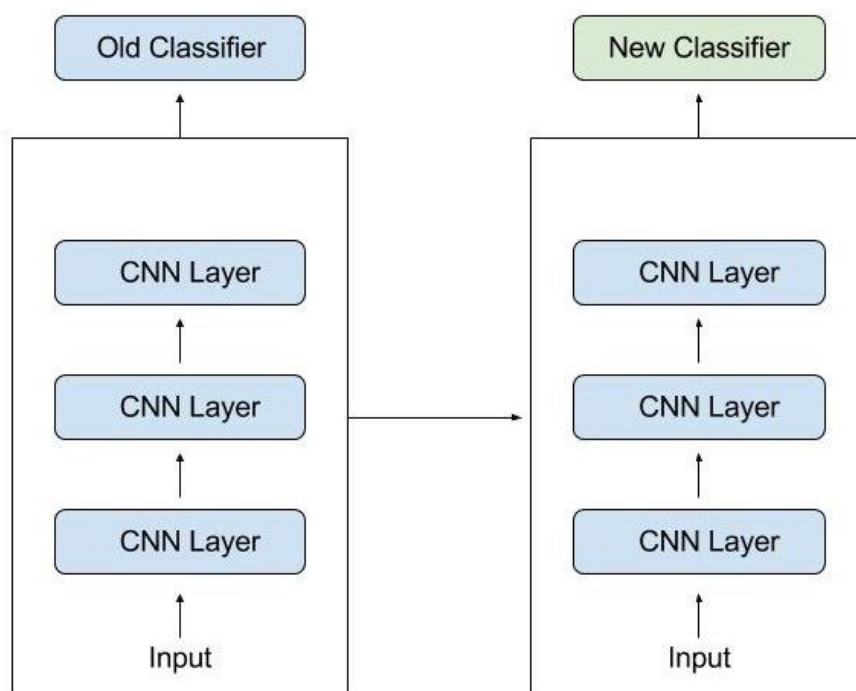


Fig 4.26 Transfer Learning

4.7.2 Why use Transfer Learning

Transfer learning has several benefits, but the main advantages are saving training time, better performance of neural networks (in most cases), and not needing a lot of data.

Usually, a lot of data is needed to train a neural network from scratch but access to that data isn't always available — this is where transfer learning comes in handy. With transfer learning a solid machine learning model can be built with comparatively little training data because the model is already pre-trained. This is especially valuable in natural language processing because mostly expert knowledge is required to create large labeled datasets. Additionally, training time is reduced because it can sometimes take days or even weeks to train a deep neural network from scratch on a complex task.

1. TRAINING A MODEL TO REUSE IT

Imagine you want to solve task A but don't have enough data to train a deep neural network. One way around this is to find a related task B with an abundance of data. Train the deep neural network on task B and use the model as a starting point for solving task A. Whether you'll need to use the whole model or only a few layers depends heavily on the problem you're trying to solve.

If you have the same input in both tasks, possibly reusing the model and making predictions for your new input is an option. Alternatively, changing and retraining different task-specific layers and the output layer is a method to explore.

2. USING A PRE-TRAINED MODEL

The second approach is to use an already pre-trained model. There are a lot of these models out there, so make sure to do a little research. How many layers to reuse and how many to retrain depends on the problem.

Keras, for example, provides nine pre-trained models that can be used for transfer learning, prediction, feature extraction and fine-tuning. You can find these models, and also some brief tutorials on how to use them, [here](#). There are also many research institutions that release trained models.

This type of transfer learning is most commonly used throughout deep learning.

3. FEATURE EXTRACTION

Another approach is to use deep learning to discover the best representation of your problem, which means finding the most important features.

This approach is also known as representation learning, and can often result in a much better performance than can be obtained with hand-designed representation.

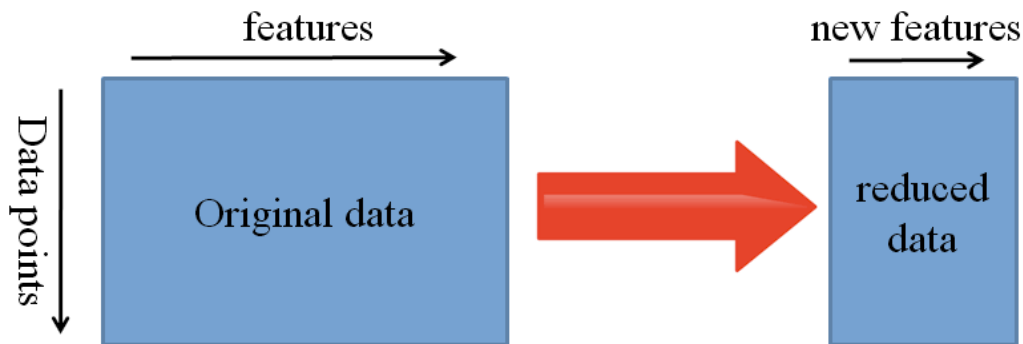


Fig 4.27 Feature extraction

In machine learning, features are usually manually hand-crafted by researchers and domain experts. Fortunately, deep learning can extract features automatically. Of course, this doesn't mean feature engineering and domain knowledge isn't important anymore — you still have to decide which features you put into your network. That said, neural networks have the ability to learn which features are really important and which ones aren't. A representation learning algorithm can discover a good combination of features within a very short timeframe, even for complex tasks which would otherwise require a lot of human effort.

The learned representation can then be used for other problems as well. Simply use the first layers to spot the right representation of features, but don't use the output of the network because it is too task-specific. Instead, feed data into your network and use one of the intermediate layers as the output layer. This layer can then be interpreted as a representation of the raw data.

This approach is mostly used in computer vision because it can reduce the size of your dataset, which decreases computation time and makes it more suitable for traditional algorithms, as well.

5. CONVNET

This chapter discusses the CNN and working of the Convolution Neural Networks

This chapter introduces convolutional neural networks (CNNs), a powerful family of neural networks that are designed for precisely this purpose. CNN-based architectures are now ubiquitous in the field of computer vision, and have become so dominant that hardly anyone today would develop a commercial application or enter a competition related to image recognition, object detection, or semantic segmentation, without building off of this approach. Modern CNNs, as they are called colloquially owe their design to inspirations from biology, group theory, and a healthy dose of experimental tinkering. In addition to their sample efficiency in achieving accurate models, CNNs tend to be computationally efficient, both because they require fewer parameters than fully-connected architectures and because convolutions are easy to parallelize across GPU cores. Consequently, practitioners often apply CNNs whenever possible, and increasingly they have emerged as credible competitors even on tasks with a one-dimensional sequence structure, such as audio, text, and time series analysis, where recurrent neural networks are conventionally used. Some clever adaptations of CNNs have also brought them to bear on graph-structured data and in recommender systems. First, we will walk through the basic operations that comprise the backbone of all convolutional networks. These include the convolutional layers themselves, nitty-gritty details including padding and stride, the pooling layers used to aggregate information across adjacent spatial regions, the use of multiple channels at each layer, and a careful discussion of the structure of modern architectures. We will conclude the chapter with a full working example of LeNet, the first convolutional network successfully deployed, long before the rise of modern deep learning. In the next chapter, we will dive into full implementations of some popular and comparatively recent CNN architectures whose designs represent most of the techniques commonly used by modern practitioners.

5.1 Definition

Convolutional neural networks are a specialized type of artificial neural networks that use a mathematical operation called convolution in place of general matrix multiplication in at least one of their layers. They are specifically designed to process pixel data and are used in image recognition and processing.

5.2 Architecture

5.2.1 Convolutional Layers

In a CNN, the input is a tensor with a shape: (number of inputs) x (input height) x (input width) x (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape: (number of inputs) x (feature map height) x (feature map width) x (feature map channels).

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus.[14] Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features and classify data, this architecture is generally impractical for larger inputs such as high-resolution images. It would require a very high number of neurons, even in a shallow architecture, due to the large input size of images, where each pixel is a relevant input feature. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for each neuron in the second layer. Instead, convolution reduces the number of free parameters, allowing the network to be deeper.

For example, regardless of image size, using a 5 x 5 tiling region, each with the same shared weights, requires only 25 learnable parameters. Using regularized weights over fewer parameters avoids the vanishing gradients and exploding gradients problems seen during backpropagation in traditional neural networks.[16][17] Furthermore, convolutional neural networks are ideal for data with a grid-like topology (such as images) as spatial relations between separate features are taken into account during convolution and/or pooling.

5.2.2 Pooling layers

Convolutional networks may include local and/or global pooling layers along with traditional convolutional layers. Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, tiling sizes such as 2 x 2 are commonly used. Global pooling acts on all the neurons of the feature map.[18][19] There are two common types of pooling in popular use: max and average. Max pooling uses the maximum value of each local cluster of neurons in the feature map,[20][21] while average pooling takes the average value.

5.2.3 Fully connected Layers

Fully connected layers connect every neuron in one layer to every neuron in another

layer. It is the same as a traditional multilayer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

5.2.4 Receptive field

In neural networks, each neuron receives input from some number of locations in the previous layer. In a convolutional layer, each neuron receives input from only a restricted area of the previous layer called the neuron's *receptive field*. Typically the area is a square (e.g. 5 by 5 neurons). Whereas, in a fully connected layer, the receptive field is the *entire previous layer*. Thus, in each convolutional layer, each neuron takes input from a larger area in the input than previous layers. This is due to applying the convolution over and over, which takes into account the value of a pixel, as well as its surrounding pixels. When using dilated layers, the number of pixels in the receptive field remains constant, but the field is more sparsely populated as its dimensions grow when combining the effect of several layers.

5.2.5 Weights

Each neuron in a neural network computes an output value by applying a specific function to the input values received from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning consists of iteratively adjusting these biases and weights.

The vector of weights and the bias are called *filters* and represent particular features of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces the memory footprint because a single bias and a single vector of weights are used across all receptive fields that share that filter, as opposed to each receptive field having its own bias and vector weighting.

5.3 Distinguishing Features

In the past, traditional multilayer perceptron (MLP) models were used for image recognition. However, the full connectivity between nodes caused the curse of dimensionality, and was computationally intractable with higher-resolution images. A 1000×1000-pixel image with RGB color channels has 3 million weights per fully-connected neuron, which is too high to feasibly process efficiently at scale. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning consists of iteratively adjusting these biases and weights.

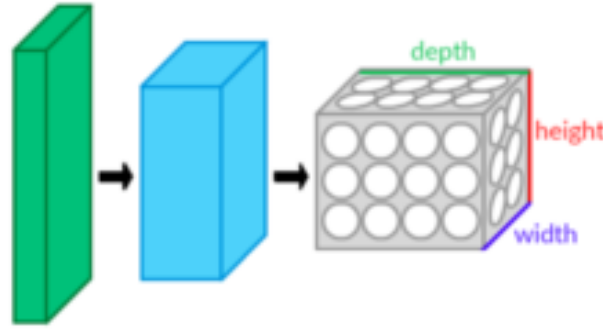


Fig 5.1 CNN layers arranged in 3 dimensions

For example, in CIFAR-10, images are only of size $32 \times 32 \times 3$ (32 wide, 32 high, 3 color channels), so a single fully connected neuron in the first hidden layer of a regular neural network would have $32 \times 32 \times 3 = 3,072$ weights. A 200×200 image, however, would lead to neurons that have $200 \times 200 \times 3 = 120,000$ weights.

Also, such network architecture does not take into account the spatial structure of data, treating input pixels which are far apart in the same way as pixels that are close together. This ignores locality of reference in data with a grid-topology (such as images), both computationally and semantically. Thus, full connectivity of neurons is wasteful for purposes such as image recognition that are dominated by spatially local input patterns.

Convolutional neural networks are variants of multilayer perceptrons, designed to emulate the behavior of a visual cortex. These models mitigate the challenges posed by the MLP architecture by exploiting the strong spatially local correlation present in natural images. As opposed to MLPs, CNNs have the following distinguishing features:

- 3D volumes of neurons. The layers of a CNN have neurons arranged in 3 dimensions: width, height and depth.^[60] Where each neuron inside a convolutional layer is connected to only a small region of the layer before it, called a receptive field. Distinct types of layers, both locally and completely connected, are stacked to form a CNN architecture.
- Local connectivity: following the concept of receptive fields, CNNs exploit spatial locality by enforcing a local connectivity pattern between neurons of adjacent layers. The architecture thus ensures that the learned "filters" produce the strongest response to a spatially local input pattern. Stacking many such layers leads to nonlinear filters that become increasingly global (i.e. responsive

to a larger region of pixel space) so that the network first creates representations of small parts of the input, then from them assembles representations of larger areas.

- **Shared weights:** In CNNs, each filter is replicated across the entire visual field. These replicated units share the same parameterization (weight vector and bias) and form a feature map. This means that all the neurons in a given convolutional layer respond to the same feature within their specific response field. Replicating units in this way allows for the resulting activation map to be equivariant under shifts of the locations of input features in the visual field, i.e. they grant translational equivariance - given that the layer has a stride of one.
- **Pooling:** In a CNN's pooling layers, feature maps are divided into rectangular sub-regions, and the features in each rectangle are independently down-sampled to a single value, commonly by taking their average or maximum value. In addition to reducing the sizes of feature maps, the pooling operation grants a degree of local translational invariance to the features contained therein, allowing the CNN to be more robust to variations in their positions.^[4]

Together, these properties allow CNNs to achieve better generalization on vision problems. Weight sharing dramatically reduces the number of free parameters learned, thus lowering the memory requirements for running the network and allowing the training of larger, more powerful networks.

5.4 Building Blocks

A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume (e.g. holding the class scores) through a differentiable function. A few distinct types of layers are commonly used. These are further discussed below.

5.4.1 Convolution Layer

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the filter entries and the input, producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

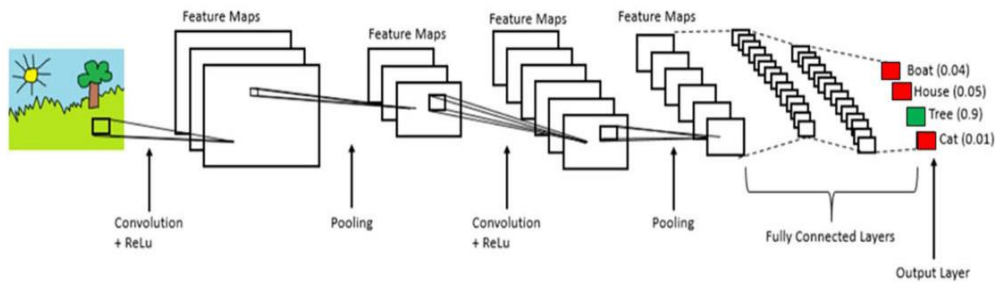


Fig 5.2 CNN Network.

When dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume because such a network architecture does not take the spatial structure of the data into account. Convolutional networks exploit spatially local correlation by enforcing a sparse local connectivity pattern between neurons of adjacent layers: each neuron is connected to only a small region of the input volume.

The extent of this connectivity is a hyperparameter called the receptive field of the neuron. The connections are local in space (along width and height), but always extend along the entire depth of the input volume. Such an architecture ensures that the learnt filters produce the strongest response to a spatially local input pattern.

Spatial Arrangement

Three hyperparameters control the size of the output volume of the convolutional layer: the depth, stride, and padding size:

- The depth of the output volume controls the number of neurons in a layer that connect to the same region of the input volume. These neurons learn to activate for different features in the input. For example, if the first convolutional layer takes the raw image as input, then different neurons along the depth dimension may activate in the presence of various oriented edges, or blobs of color.
- Stride controls how depth columns around the width and height are allocated. If the stride is 1, then we move the filters one pixel at a time. This leads to

heavily overlapping receptive fields between the columns, and to large output volumes. For any integer a stride S means that the filter is translated S units at a time per output. In practice, is rare. A greater stride means smaller overlap of receptive fields and smaller spatial dimensions of the output volume.

- Sometimes, it is convenient to pad the input with zeros (or other values, such as the average of the region) on the border of the input volume. The size of this padding is a third hyperparameter. Padding provides control of the output volume's spatial size. In particular, sometimes it is desirable to exactly preserve the spatial size of the input volume, this is commonly referred to as "same" padding.

The spatial size of the output volume is a function of the input volume size, the kernel field size of the convolutional layer neurons, the stride, and the amount of zero padding on the border. The number of neurons that "fit" in a given volume is then:

If this number is not an integer, then the strides are incorrect and the neurons cannot be tiled to fit across the input volume in a symmetric way. In general, setting zero padding to be when the stride is ensures that the input volume and output volume will have the same size spatially. However, it is not always completely necessary to use all of the neurons of the previous layer. For example, a neural network designer may decide to use just a portion of padding.

A parameter sharing scheme is used in convolutional layers to control the number of free parameters. It relies on the assumption that if a patch feature is useful to compute at some spatial position, then it should also be useful to compute at other positions. Denoting a single 2-dimensional slice of depth as a *depth slice*, the neurons in each depth slice are constrained to use the same weights and bias.

Since all neurons in a single depth slice share the same parameters, the forward pass in each depth slice of the convolutional layer can be computed as a convolution of the neuron's weights with the input volume. Therefore, it is common to refer to the sets of weights as a filter (or a kernel), which is convolved with the input. The result of this convolution is an activation map, and the set of activation maps for each different filter are stacked together along the depth dimension to produce the output volume. Parameter sharing contributes to the translation invariance of the CNN architecture.

Sometimes, the parameter sharing assumption may not make sense. This is especially the case when the input images to a CNN have some specific centered structure; for which we

expect completely different features to be learned on different spatial locations. One practical example is when the inputs are faces that have been centered in the image: we might expect different eye-specific or hair-specific features to be learned in different parts of the image. In that case it is common to relax the parameter sharing scheme, and instead simply call the layer a "locally connected layer".

4.4.2 Pooling Layer

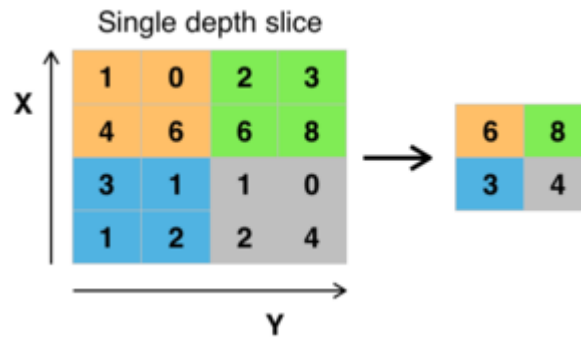


Fig 5.3 Max pooling with a 2x2 filter and stride = 2

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling, where *max pooling* is the most common. It partitions the input image into a set of rectangles and, for each such sub-region, outputs the maximum. Intuitively, the exact location of a feature is less important than its rough location relative to other features. This is the idea behind the use of pooling in convolutional neural networks. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters, memory footprint and amount of computation in the network, and hence to also control overfitting. This is known as down-sampling. It is common to periodically insert a pooling layer between successive convolutional layers (each one typically followed by an activation function, such as a ReLU layer) in a CNN architecture.^{[62]:460–461} While pooling layers contribute to local translation invariance, they do not provide global translation invariance in a CNN, unless a form of global pooling is used.^{[4][61]} The pooling layer commonly operates independently on every depth, or slice, of the input and resizes it spatially. A very common form of max pooling is a layer with filters of size 2×2, applied with a stride of 2, which subsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations: In this case, every max operation is over 4 numbers. The depth dimension remains unchanged (this is true for other forms of pooling as well). In addition to max pooling, pooling units can use other functions, such as average pooling or ℓ_2 -norm pooling. Average pooling was often used historically but has

recently fallen out of favor compared to max pooling, which generally performs better in practice. Due to the effects of fast spatial reduction of the size of the representation, there is a recent trend towards using smaller filters^[65] or discarding pooling layers altogether. "Region of Interest" pooling (also known as RoI pooling) is a variant of max pooling, in which output size is fixed and input rectangle is a parameter. Pooling is a down sampling method and an important component of convolutional neural networks for object detection based on the Fast R-CNN architecture.

5.4.3 ReLU Layer

ReLU is the abbreviation of rectified linear unit, which applies the non-saturating activation function. It effectively removes negative values from an activation map by setting them to zero.^[69] It introduces nonlinearities to the decision function and in the overall network without affecting the receptive fields of the convolution layers. Other functions can also be used to increase nonlinearity, for example the saturating hyperbolic tangent, and the sigmoid function. ReLU is often preferred to other functions because it trains the neural network several times faster without a significant penalty to generalization accuracy.

5.4.4 Fully connected layer

After several convolutional and max pooling layers, the final classification is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular (non-convolutional) artificial neural networks. Their activations can thus be computed as an affine transformation, with matrix multiplication followed by a bias offset vector addition of a learned or fixed bias term.

5.4.5 Loss Layer

The "loss layer", or "loss function", specifies how training penalizes the deviation between the predicted output of the network, and the true data labels (during supervised learning). Various loss functions can be used, depending on the specific task.

The Softmax loss function is used for predicting a single class of K mutually exclusive classes. Sigmoid cross-entropy loss is used for predicting K independent probability values in \mathbb{R} . Euclidean loss is used for regressing to real-valued labels.

5.4.6 SoftMax

The SoftMax provides us the probability of the object passed to the network. The SoftMax layer classifies the object belongs to which class based on the probability.

$$f(y_i) = \frac{e^{y_i}}{\sum e^{y_k}}$$

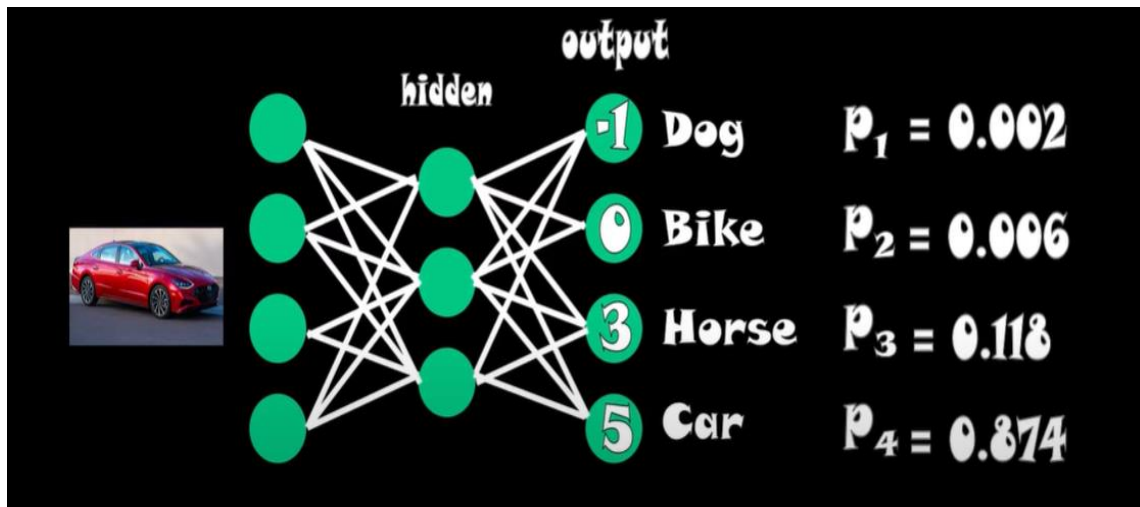


Fig 5.4 Example of SoftMax

6. IMPLEMENTATION

This chapter discusses the Implementation of the methods discussed in the previous chapter

Our Proposed system provides three features like:

1. Benign or Malignant
2. Type of Tumor
3. Tumor Spot

6.1 Classification between Benign or Malignant

Classification of the Tumor present in the MRI scan between Benign or Malignant is done by the use of Super Vector Machines.

User Interface

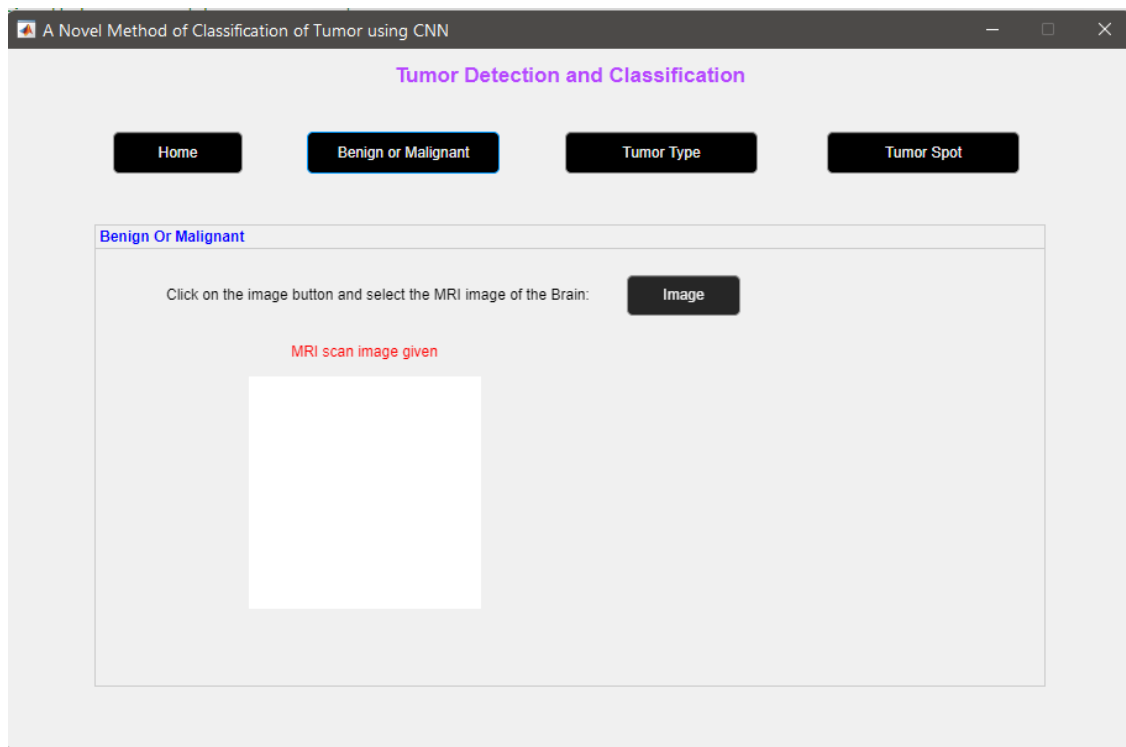


Fig 6.1 Benign or Malignant User Interface

The image button is the Push Button element which has an callback of browsing the system folders and helps in selecting the file of any format.

The CallBack of Image Button is :

```
[file,path]=uigetfile('*.','MRI input image');
```

```
MRI_scan=imread(fullfile(path,file));
```

Implementation of SVM

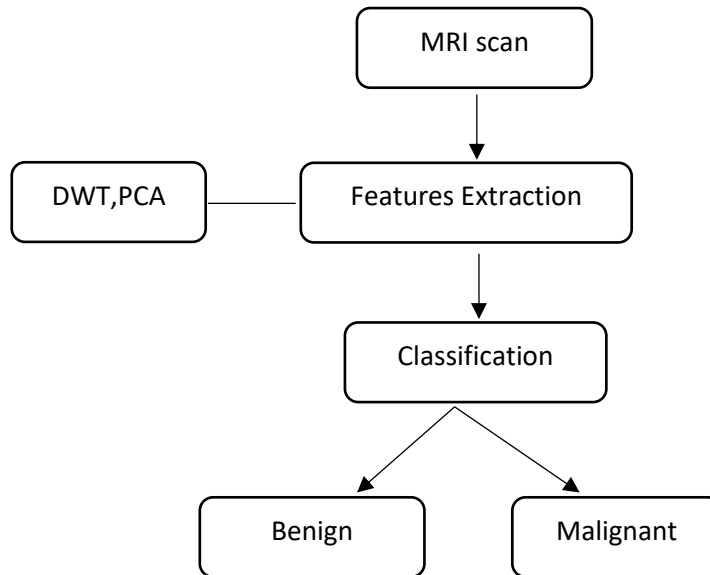


Fig 6.2 SVM Flow Chart

Step 1:

The given Input image is converted to Binary Image

```
img2 = im2bw(MRI_scan);
```

Step 2:

Features are extracted from the binary image using discrete wavelet transform and Principal Component Analysis.

DWT:

```
[cA1,cH1,cV1,cD1] = dwt2(signal1,'db4');
```

```
[cA2,cH2,cV2,cD2] = dwt2(cA1,'db4');
```

```
[cA3,cH3,cV3,cD3] = dwt2(cA2,'db4');
```

```
DWT_feat = [cA3,cH3,cV3,cD3];
```

PCA:

```
G = pca(DWT_feat);
```

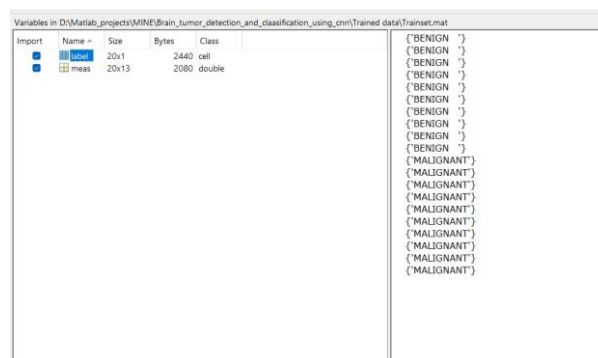
Step 3:

Extracted Features are

```
g = graycomatrix(G);
stats = graycoprops(g,'Contrast Correlation Energy Homogeneity');
Contrast = stats.Contrast;
Correlation = stats.Correlation;
Energy = stats.Energy;
Homogeneity = stats.Homogeneity;
Mean = mean2(G);
Standard_Deviation = std2(G);
Entropy = entropy(G);
RMS = mean2(rms(G));
Variance = mean2(var(double(G)));
a = sum(double(G(:)));
Smoothness = 1-(1/(1+a));
Kurtosis = kurtosis(double(G(:)));
Skewness = skewness(double(G(:)));
```

Step 4:

Loading the trained data and classifying the new object



Import	Name	Size	Bytes	Class
	label	20x1	2440	cell
	meas	20x13	2080	double

('BENIGN')
('BENIGN')
('BENIGN')
('BENIGN')
('BENIGN')
('BENIGN')
('BENIGN')
('BENIGN')
('BENIGN')
('BENIGN')
('MALIGNANT')
('MALIGNANT')
('MALIGNANT')
('MALIGNANT')
('MALIGNANT')
('MALIGNANT')
('MALIGNANT')
('MALIGNANT')
('MALIGNANT')
('MALIGNANT')

Fig 6.3 Labels in Trained data of SVM

Here the data of string format of names benign and malignant are store in a 20X1 matrix format with corresponding measurements associated to them.

Import	Name	Size	Bytes	Class
<input checked="" type="checkbox"/>	label	20x1	2440	cell
<input checked="" type="checkbox"/>	meas	2080	double	

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.2333	0.1284	0.7491	0.9308	0.0019	0.0898	2.6632	0.0898	0.0081	0.8778	7.2707	0.6117	-0.0366
2	0.2717	0.0931	0.7686	0.9338	0.0024	0.0898	3.2698	0.0898	0.0081	0.8974	7.9567	0.8862	0.4926
3	0.2272	0.1326	0.7439	0.9290	0.0043	0.0897	3.6046	0.0898	0.0080	0.9406	5.9972	0.5218	0.3700
4	0.2442	0.1007	0.7409	0.9263	0.0032	0.0898	3.5797	0.0898	0.0080	0.9234	6.2735	0.6332	0.5257
5	0.2033	0.1126	0.7554	0.9331	0.0019	0.0898	3.6549	0.0898	0.0080	0.8783	5.8117	0.3408	1.0010
6	0.2556	0.0895	0.7557	0.9314	0.0025	0.0898	3.0756	0.0898	0.0081	0.9040	7.7971	0.5774	-0.2601
7	0.2155	0.0951	0.7378	0.9274	0.0038	0.0898	3.6283	0.0898	0.0080	0.9132	5.3238	0.3230	1.0419
8	0.2925	0.1584	0.7586	0.9330	0.0057	0.0896	2.6622	0.0898	0.0080	0.9551	13.0402	1.3124	1.2778
9	0.2341	0.1321	0.7530	0.9315	0.0035	0.0897	3.1562	0.0898	0.0080	0.9291	7.4848	0.5212	-1.0392
10	0.2689	0.0977	0.7861	0.9410	6.8659e-04	0.0898	2.7465	0.0898	0.0081	0.7186	10.9703	0.7365	0.1190
11	0.2433	0.1294	0.7606	0.9344	0.0034	0.0897	2.9949	0.0898	0.0081	0.9270	7.6801	0.6318	0.3816
12	0.2272	0.1326	0.7439	0.9290	0.0043	0.0897	3.6046	0.0898	0.0080	0.9406	5.9972	0.5218	0.3700
13	0.2750	0.1180	0.7688	0.9346	0.0046	0.0897	3.0290	0.0898	0.0081	0.9453	13.1839	1.0085	0.2863
14	0.2272	0.0908	0.7522	0.9308	0.0034	0.0897	3.6783	0.0898	0.0080	0.9270	5.5966	0.4004	1.0469
15	0.2517	0.0734	0.7402	0.9267	0.0035	0.0897	3.5239	0.0898	0.0080	0.9284	6.5220	0.4979	1.6524
16	0.2439	0.1072	0.7310	0.9246	0.0046	0.0897	3.5484	0.0898	0.0081	0.9446	6.5235	0.6204	0.5030
17	0.2925	0.1584	0.7586	0.9330	0.0057	0.0896	2.6622	0.0898	0.0080	0.9551	13.0402	1.3124	1.2778
18	0.2745	0.1095	0.7549	0.9308	0.0054	0.0897	3.1085	0.0898	0.0080	0.9523	11.1148	1.0231	-0.6151
19	0.2161	0.1382	0.7548	0.9325	0.0025	0.0898	3.3156	0.0898	0.0081	0.9032	6.2320	0.3121	0.5631
20	0.2786	0.1427	0.7604	0.9321	0.0053	0.0897	3.1943	0.0898	0.0081	0.9516	9.7318	0.9914	1.8546

Fig 6.4 Measurements in Trained data of SVM

Code:

```
load 'Trained data'\Trainset.mat

xdata = meas;

group = label;

svmStruct1 = fitsvm(xdata,group,'KernelFunction', 'linear');

species= predict(svmStruct1,feat);
```

Here predict returns the given MRI scan it was Benign or Malignant.

6.2 Classification of Tumor Type

For the classification of Type of tumor we are using CNN and we are using a pretrained Network it is Google Net.

6.2.1 Google Net

GoogLeNet is a convolutional neural network that is 22 layers deep. You can load a pretrained version of the network trained on either the ImageNet or Places365 data sets. The network trained on ImageNet classifies images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. The network trained on Places365 is similar to the network trained on ImageNet, but classifies images into 365 different place categories, such as field, park, runway, and lobby. These networks have learned different feature representations for a wide range of images. The pretrained networks both have an image input size of 224-by-224.

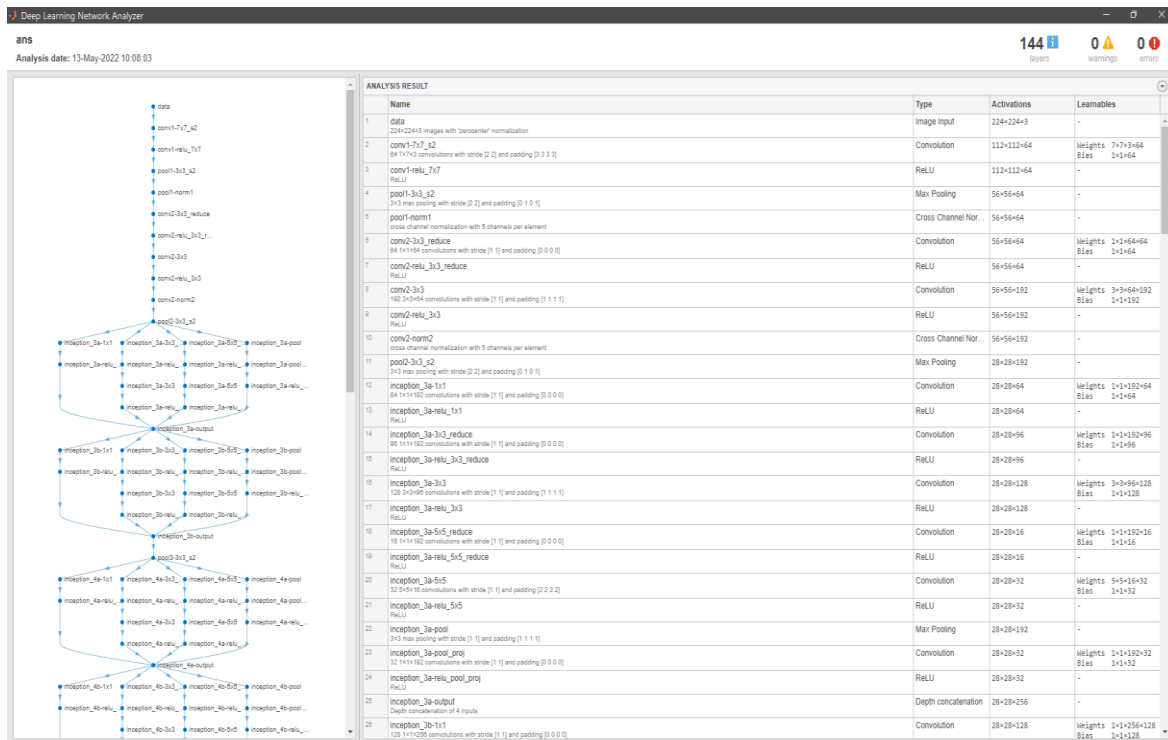


Fig 6.5 Google Net Architecture

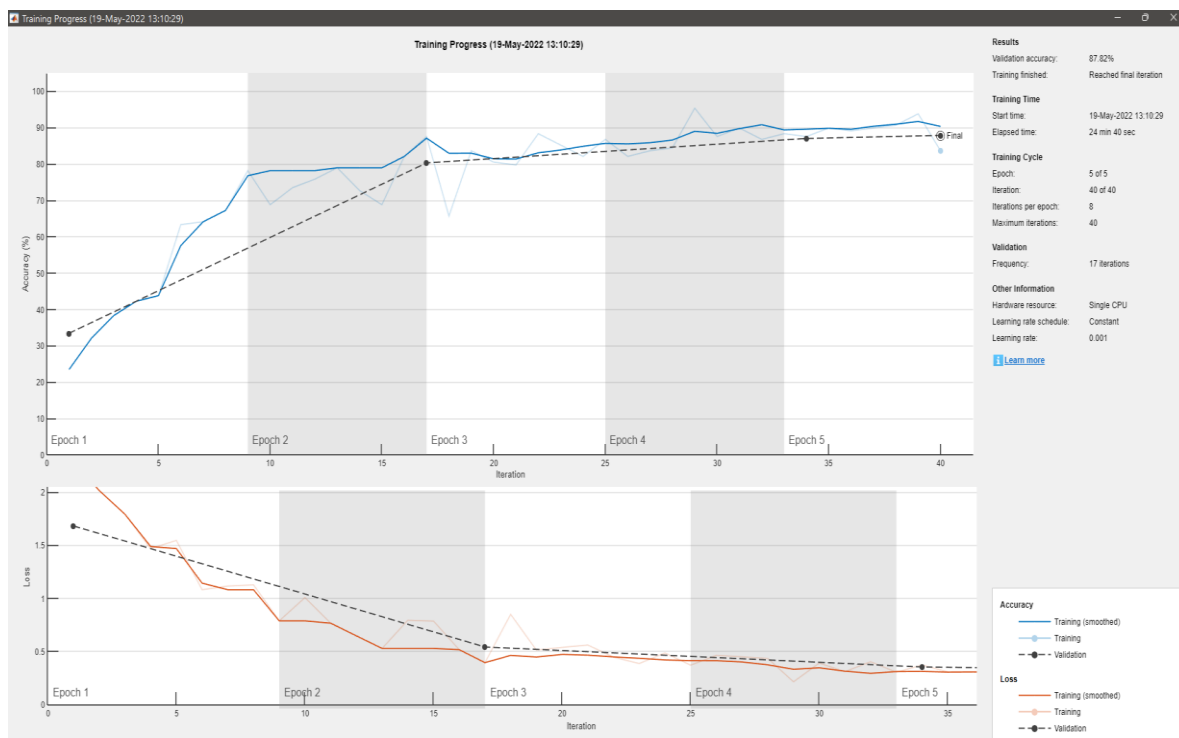


Fig 6.6 Training Progress

User Interface

The image button is the Push Button element which has an call back of browsing the system folders and helps in selecting the file of any format.

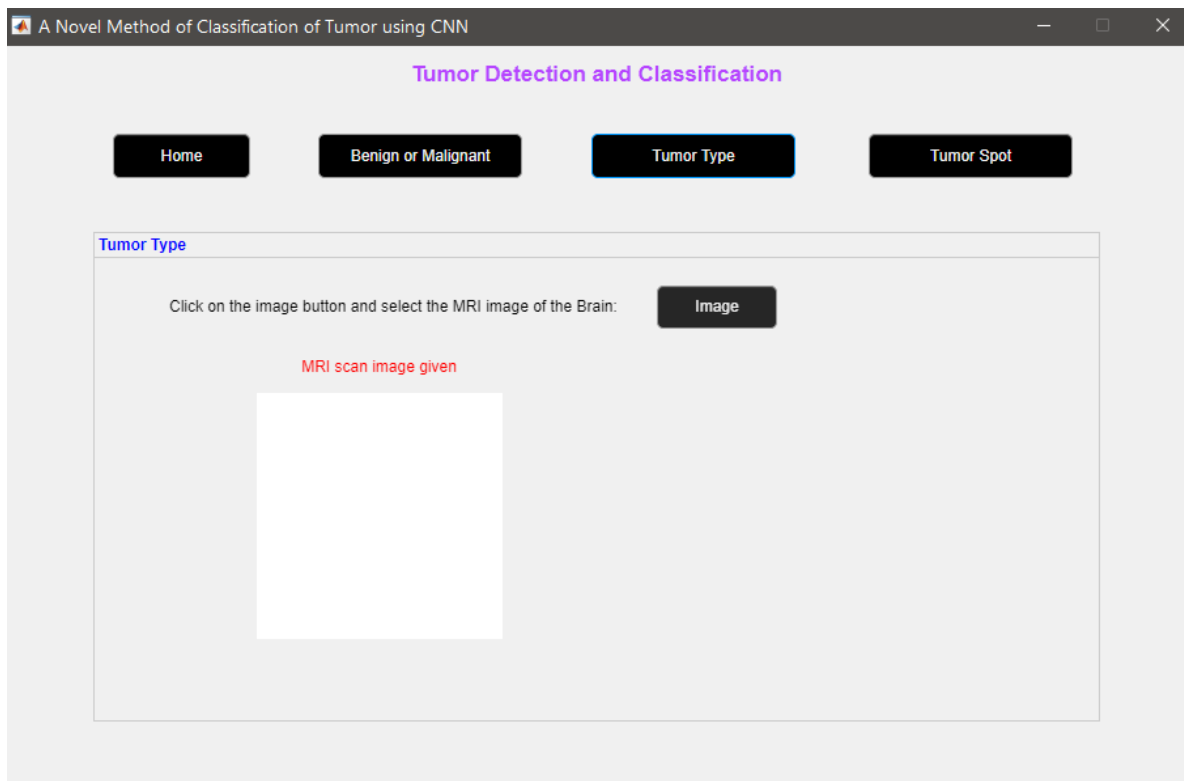


Fig 6.7 Tumor Type UI

The Call-back of Image Button is :

Code:

```
[file,path]=uigetfile('*.','MRI input image');  
MRI_scan=imread(fullfile(path,file));
```

Implementation of CNN google net

The google net is an convolution network, here we are using this network to perform our task of classifying the data based on four classes. After training the network to perform our task we are going save all those parameters and features in an MATLAB. And every time for the classification of new object, which is an user input and this input image and the trained data is passed to an function which returns us an class and probability of the input image belonging to the class based on the trained data.

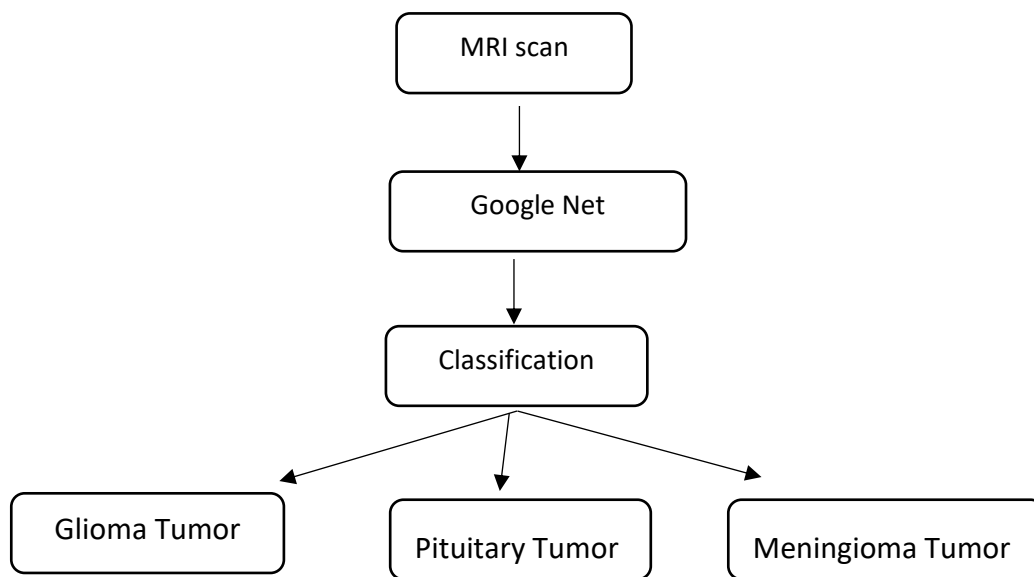


Fig 6.8 Flow chart of Classification of type

Code:

```

load 'Trained data'\cnn87.mat
MRI_scan=imresize(MRI_scan,[224,224]);
[label,~]=classify(net,MRI_scan);

```

The Input Layer of google net is [224,224,3], so we are resizing the image and passing to network for classification.

The classify method returns the class type of the passed MRI scan to the network named as net.

6.3 Tumor Spot

For detection of tumor in MRI scan we are Thresholding Method, in thresholding we are using Otsu method.

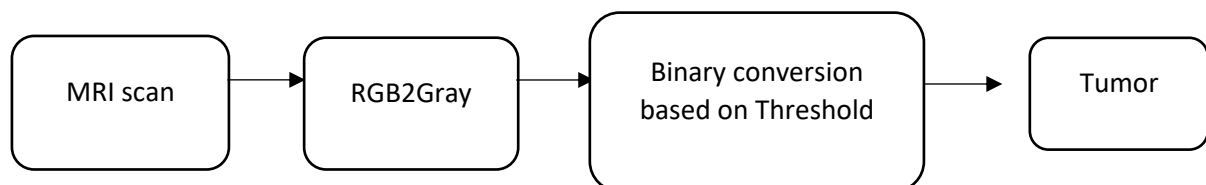


Fig 6.9 Block diagram of Tumor detection

Code:

```
I=MRI_scan;  
gray = rgb2gray(I);  
img = im2bw(gray,.6);  
img = bwareaopen(img,80);
```

User Interface

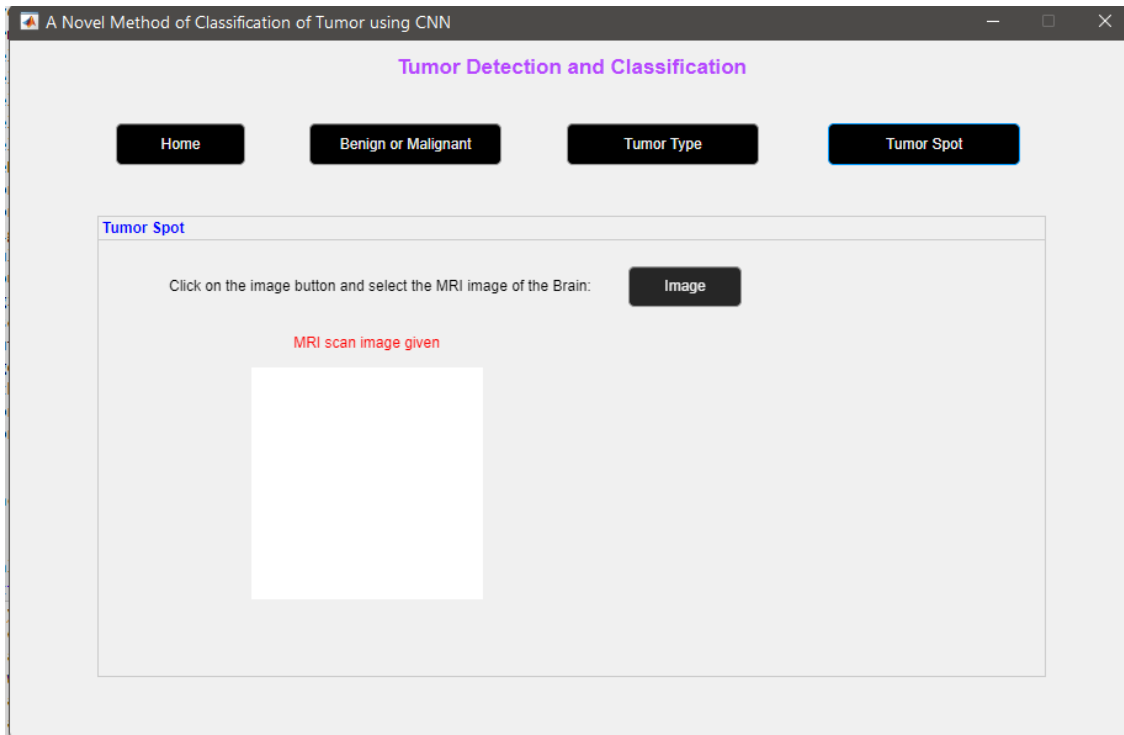


Fig 6.10 UI of Tumor Spot

7. RESULTS

Benign or Malignant

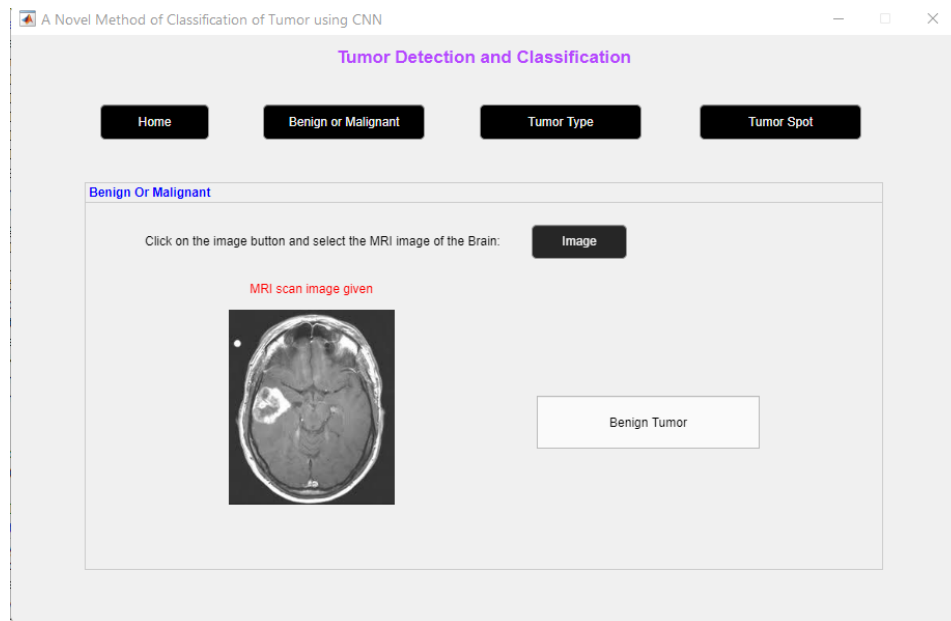


Fig 7.1 Result of Benign or Malignant

In this the MRI scan is taken as input from the user and based on the trained dataset it will be classified with the help of SVM either it was Benign Tumor or Malignant Tumor

Tumor Type

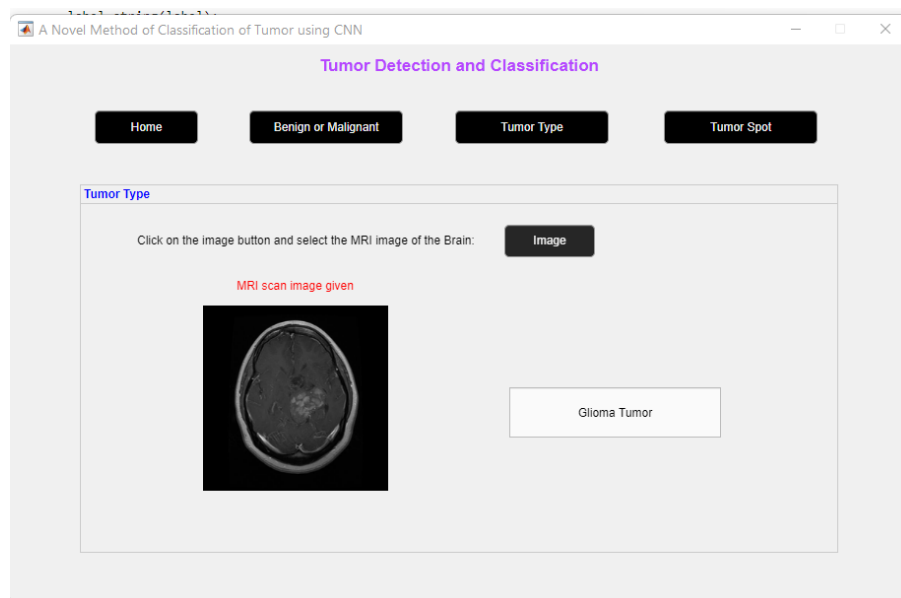


Fig 7.2 Result of Tumor Type

In this the MRI scan is taken as the user input and based the on trained data of network with the help Google Net and the tumor is classified as of which type based on three categories.

Tumor Spot

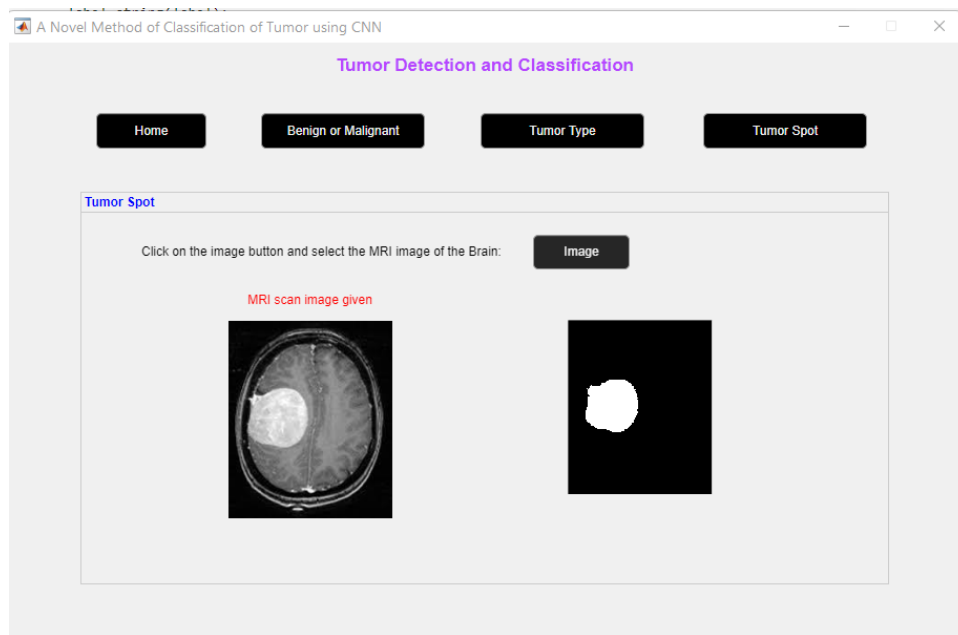


Fig 7.3 Result of Detection of Tumor

In this the MRI scan of the brain is taken as input from the user and using the Thresholding Technique the shape and the spot is displayed as the result.

8. CONCLUSION & FUTURE SCOPE

8.1 Conclusion

Abnormal growth of tissue in the brain that affects the brain's normal functioning is considered a brain tumor. The main goal of medical image processing is to identify accurate and meaningful information using algorithms with the minimum error possible. Brain tumor detection and classification through MRI images can be categorized into four sections: pre-processing, image segmentation, feature extraction and image classification. Machine Learning methodologies are explored in the project. It can be concluded that the algorithms and the parameters used in the proposed system are all meant to increase the system's efficiency by achieving better results.

The Thresholding segmentation approach gives more results in our testing for the tumor spot detection; in the thresholding approach, we are using Otsu's Method, which uses the threshold level from 0 to 1 and binarizes the image and helps in identifying and displaying the tumor spots. Features extracted using the GLCM method help increase efficiency as minute details of the tumor using various features can be extracted. And we are using these features to classify cancer as Benign or Malignant. And in another way, our software helps the user to classify the Tumor present in the provided MRI scan between Glioma, Pituitary and Meningioma. We are using CNN, which is far better than the SVM in classifying the object between the number of classes for this classification. Accuracy and reliability are of utmost importance in tumor diagnosis, as a patient's life depends on the results predicted by the system. Thus, the proposed methodology helps in increasing the accuracy and obtaining the desired results.

8.2 Future Scope

Encouraged by these results, future work will involve improving classification results and overall accuracy. With a more extensive and diverse dataset, the overall classification accuracy can be drastically increased. The number of output classes can also be increased if more data is available.

Another approach to improve the result would be to increase the number of hidden layers of the neural network. By increasing the number of hidden layers, the weights will be better adjusted and thus increase the classification. One can also do fine-tuning and transfer learning approaches to better tune the model based on already trained models.

REFERENCES

- [1] M. Al-Ayyoub, G. Husari, O. Darwish, and A. Alabed-Alaziz, "Machine learning approach for brain tumor detection," Proceedings of the 3rd International Conference on Information and Communication Systems - ICICS 12, 2012.
- [2] Kaur, Mandhir, and Rinkesh Mittal. "Survey of Intelligent Methods for Brain Tumor Detection." International Journal of Computer Science Issues (IJCSI) 11.5 (2014): 108.
- [3] A. Ladgham, G. Torkhani, A. Sakly, and A. Mtibaa, "Modified support vector machines for MR brain images recognition," 2013 International Conference on Control, Decision and Information Technologies (CoDIT), 2013.
- [4] P. Su, Z. Xue, L. Chi, J. Yang, and S. T. Wong, "Support vector machine (SVM) active learning for automated Glioblastoma segmentation," 2012 9th IEEE International Symposium on Biomedical Imaging (ISBI), 2012.
- [5] R. Lang, L. Zhao, and K. Jia, "Brain tumor image segmentation based on convolution neural network," 2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP- BMEI), 2016.
- [6] Liu, Tianyi, et al. "Implementation of training convolutional neural networks." arXiv preprint arXiv:1506.01195 (2015).

GLOSSARY

MATLAB: Matrix Laboratory, is a programming and numeric computing platform used by millions of engineers and scientists to analyze data, develop algorithms, and create models.

CT: computed tomography, scan combines a series of X-ray images taken from different angles around your body and uses computer processing to create cross-sectional images (slices) of the bones, blood vessels and soft tissues inside your body.

MRI: Magnetic resonance imaging, is a medical imaging technique that uses a magnetic field and computer-generated radio waves to create detailed images of the organs and tissues in your body.

PSNR: Peak single - to noise – ratio, is an expression for the ratio between the maximum possible value (power) of a signal and the power of distorting noise that affects the quality of its representation.

DWT: Discrete wavelet transform, is a transform that decomposes a given signal into a number of sets, where each set is a time series of coefficients.

SVM: Support Vector Machine, is one of the classification technique applied on different fields such as face recognition, text categorization, cancer diagnosis, glaucoma diagnosis, microarray gene expression data analysis.

CNN: Convolution Neural Network, is one of the most popular algorithms for deep learning with images and video. Like other neural networks, a CNN is composed of an input layer, an output layer, and many hidden layers in between.

ReLU: Rectified linear unit, allows for faster and more effective training by mapping negative values to zero and maintaining positive values. These three operations are repeated over tens or hundreds of layers, with each layer learning to detect different features.

OTSU: Operational Test Support Unit, the method instinctively performs clustering-based image thresholding, or, the diminishing of a grayscale image to a binary or threshold image.

GLCM: Gray-level co-occurrence matrix, Co-occurrence distribution is a matrix showing different combination of gray levels found within the image [63, 64].

PCA: Principal Component Analysis, is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation.

RMS: Root Mean Square, is calculated on a set of pixels by taking the square of each pixel, calculating the sum of those squares, and taking the square root. The result is scaled by the number of pixels. RMS gives an accurate measurement of the amount of noise present.

IDM: Inverse Difference Movement, is the local homogeneity. It is high when local gray level is uniform and inverse GLCM is high.

MLP: Multi-layer perceptron, is a feed forward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input nodes connected as a directed graph between the input and output layers.

NN: Neural Network, is a series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

FC: Fully connected layer, multiplies the input by a weight matrix and then adds a bias vector. The convolutional (and down-sampling) layers are followed by one or more fully connected layers.

LL: Loss Layer, specifies how training penalizes the deviation between the predicted output of the network, and the true data labels (during supervised learning). Various loss functions can be used, depending on the specific task.

GUI: Graphic User Interface, graphics-based operating system interface that uses icons, menus and a mouse (to click on the icon or pull down the menus) to manage interaction with the system.

APPENDIX

1. System Requirements

Software Tools:

1. Programming Language: MATLAB
2. IDE: MATLAB 2017a and above
3. MATLAB Toolboxes:
 - Image Processing Toolbox
 - Deep Learning Toolbox
 - App Designer
4. Database for Training and Testing
<https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri?resource=download>

Hardware Tools:

1. Intel Dual Core Dual Processor or advanced version
2. Minimum 8GB of RAM
- 3 .Minimum 1 GB of Hard disk Space

2. Online Resources Used

MathWorks: <https://in.mathworks.com/>