

# **INTEGRATED SVM-CNN FRAMEWORK OF BRAIN TUMOR DETECTION AND PROGNOSIS**

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

By

**VENKATA KRISHNAN T R (Reg. No – 40111418)**

**KARTHIKEYAN M (Reg. No – 40110558)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING**

## **SATHYABAMA**

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

**CATEGORY -1 UNIVERSITY BY UGC**

**Accredited with Grade “A++” by NAAC I 12B Status by UGC I Approved by AICTE  
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

**APRIL – 2024**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **VENKATA KRISHNAN T R (Reg.No.40111418)**, **KARTHIKEYAN M (Reg.No.40110558)** who carried out the Project entitled “**INTEGRATED SVM-CNN FRAMEWORK OF BRAIN TUMOR DETECTION AND PROGNOSIS**” under my supervision from NOVEMBER 2023 to APRIL 2024.

**Internal Guide**

**Dr. Sankari M. ME., PhD.**

**Head of the Department**

**Dr. L. LAKSHMANAN, M.E., Ph.D.**

Submitted for Viva voce Examination held on \_\_\_\_\_

**Internal Examiner**

**External Examiner**

## **DECLARATION**

I, **VENKATA KRISHNAN T R (40111418)** hereby declare that the Project Report entitled “**INTEGRATED SVM-CNN FRAMEWORK OF BRAIN TUMOR DETECTION AND PROGNOSIS**” is done by me under the guidance of **Dr. Sankari M. M.E., Ph.D.** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

**DATE:**

**PLACE: CHENNAI**

**SIGNATURE OF THE CANDIDATE**

## ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **The Board of Management of Sathyabama Institute of Science and Technology** for their encouragement in doing this project and for completing it successfully. I am grateful to them.

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

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

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

## **ABSTRACT**

Brain Tumor is one of the leading causes of cancer death worldwide. Early detection of brain cancer is challenging because cancer-specific symptoms occur only at an advanced stage, and a reliable screening tool to identify high-risk patients is lacking. However, brain cancer can be cured if it is detected at an early stage. In this project, an attempt is made to detect brain tumor from CT images. It uses image processing techniques and a CNN Model Architecture to detect the tumor. After the image pre-processed, CNN Model Architecture is used to detect the tumorous area in the image. It is observed that the training accuracy of detecting tumor. The Solution is to enhance brain tumor analysis, we propose a hybrid approach combining SVM and CNN. SVM initially categorizes brain scans into tumor types, forming distinct groups. Then, CNN models operate within these groups to predict tumor presence. This integration optimizes both classification and prediction accuracy, improving diagnosis and treatment planning. CNNs have also proved beneficial for brain tumor prediction. These deep learning methods develop complicated representations from vast datasets of brain scans, predicting tumor presence or absence. Evaluation criteria including as accuracy, precision, recall, and F1-score quantify the efficacy of CNN models in predicting brain tumor status. Further study is required for greater performance and bigger datasets. Combining SVM for classification with CNN for prediction gives a hybrid technique for brain tumor analysis. SVM algorithms categorize brain scans into tumor kinds, while CNN models identify tumor presence within these groupings. This combination technique delivers precise information on brain tumor categorization and prediction, boosting the accuracy of diagnosis and treatment planning. Continued research will further develop these algorithms for better brain cancer assessment.

## TABLE OF CONTENTS

| CHAPTER NO. | TITLE                                                           | PAGE NO. |
|-------------|-----------------------------------------------------------------|----------|
|             | ABSTRACT                                                        | v        |
|             | LIST OF FIGURES                                                 | viii     |
|             | LIST OF ABBREVIATIONS                                           | x        |
| 1           | INTRODUCTION                                                    | 1        |
|             | 1.1 INTRODUCTION TO DEEP LEARNING                               | 1        |
|             | 1.2 CONVOLUTIONAL NEUTRAL NETWORKS                              | 2        |
| 2           | LITERATURE SURVEY                                               | 4        |
| 3           | REQUIREMENT ANALYSIS                                            | 6        |
|             | 3.1 EXISTING SYSTEM                                             | 6        |
|             | 3.1.1 DISADVANTAGES OF EXISTING SYSTEM                          | 6        |
|             | 3.2 PROPOSED SYSTEM                                             | 6        |
|             | 3.2.1 ADVANTAGES OF PROPOSED SYSTEM                             | 7        |
| 4           | DESCRIPTION OF PROPOSED SYSTEM                                  | 8        |
|             | 4.1 PROPOSED ARCHITECTURE                                       | 8        |
|             | 4.2 INPUT DESIGN                                                | 8        |
|             | 4.3 OUTPUT DESIGN                                               | 9        |
|             | 4.4 SYSTEM REQUIREMENTS                                         | 9        |
|             | 4.4.1 HARDWARE REQUIREMENTS                                     | 9        |
|             | 4.4.2 SOFTWARE REQUIREMENTS                                     | 9        |
|             | 4.5 SYSTEM DESIGN                                               | 9        |
| 5           | IMPLEMENTATION DETAILS                                          | 41       |
|             | 5.1 MODULES                                                     | 41       |
|             | 5.1.1 DATASET                                                   | 41       |
|             | 5.1.2 IMPORTING THE NECESSARY LIBRARIES                         | 41       |
|             | 5.1.3 RETRIEVING THE IMAGES                                     | 41       |
|             | 5.1.4 SPLITTING THE DATASET                                     | 41       |
|             | 5.1.5 BUILDING THE MODEL                                        | 42       |
|             | 5.1.6 APPLY THE MODEL AND PLOT THE GRAPHS FOR ACCURACY AND LOSS | 42       |

|   |                                          |    |
|---|------------------------------------------|----|
|   | <b>5.1.7 ACCURACY ON TEST SET</b>        | 42 |
|   | <b>5.1.8 SAVING THE TRAINED MODEL</b>    | 42 |
|   | <b>5.2 CONVOLUTIONAL NEURAL NETWORKS</b> | 42 |
|   | <b>5.3 STRIDES CONVOLUTIONS</b>          | 46 |
| 6 | <b>RESULTS AND DISCUSSION</b>            | 50 |
| 7 | <b>CONCLUSION</b>                        | 52 |
|   | <b>REFERENCES</b>                        | 53 |
|   | <b>APPENDIX</b>                          |    |
|   | <b>A. SAMPLE CODE</b>                    | 54 |
|   | <b>B. SAMPLE CODE SNIPPETS</b>           | 59 |
|   | <b>C. RESEARCH PAPER</b>                 | 60 |
|   | <b>D. PRESENTATION CERTIFICATE</b>       | 65 |

## LIST OF FIGURES

| S.NO | FIGURE NAMES                                                         | PAGE |
|------|----------------------------------------------------------------------|------|
| 4.1  | Proposed Architecture                                                | 08   |
| 4.2  | Processing of Image – Flow diagram                                   | 11   |
| 4.3  | Quantization of Colors                                               | 16   |
| 4.4  | Convolution of Colors                                                | 17   |
| 4.5  | 1st and 2nd derivative of an edge illustrated in one dimension.      | 21   |
| 4.6  | Convolution kernel for a mean filter with 3×3 neighborhood           | 25   |
| 4.7  | Truth-tables for AND and NAND                                        | 26   |
| 4.8  | Simple gray level image and the corresponding surface in image space | 30   |
| 4.9  | Two connected components based on 4-connectivity.                    | 35   |
| 4.10 | Example structuring elements.                                        | 38   |
| 4.11 | Mapping function for wrapping the pixel values of an 8-bit image     | 39   |
| 4.12 | Mapping function for saturating an 8-bit image                       | 40   |
| 5.1  | Vision Edges Conversion                                              | 43   |
| 5.2  | Conversion table                                                     | 45   |
| 5.3  | Stride Convolutions                                                  | 47   |
| 5.4  | Pooling                                                              | 48   |
| 6.1  | segmented Image Conversion output                                    | 50   |
| 6.2  | Segmented Image Conversion output- Benign Tumor detection            | 51   |
| 6.3  | Malignant Tumor Detection                                            | 51   |
|      | Sample Code Snippet -1                                               | 54   |
|      | Sample Code Snippet -2                                               | 54   |



## **LIST OF ABBREVIATIONS**

- SVM - Support Vector Machine
- CNN - Convolution Neural Networks
- MRI - Magnetic Resonance Imaging
- AUC - Area Under the Curve
- ROC - Receiver Operating Characteristic
- HOG - Histogram of oriented gradients.
- LBP - Local Binary Patterns
- GLCM - Gray-Level Co-Occurrence Matrix
- PCA - Principal Component Analysis
- RFE - Recursive Feature Elimination
- SE - Squeeze-and-Excitation
- CT - Computerized Tomography
- FMRI - Functional Magnetic Resonance Imaging

## **CHAPTER 1**

### **INTRODUCTION**

Brain Cancer is the 10<sup>th</sup> most diagnosed cancer in men and the 9<sup>th</sup> in women, but the 4<sup>th</sup> leading cancer death for both men and women in the United States. Brain Cancer is only the major cancer with a five-year relative survival rate in the single digits. A recent report issued by brain cancer action in 2020, brain cancer is expected to become the second largest leading cancer cause of cancer death in the United States.

A major cause of this, is the late detection of the brain tumor because there are no effective early detection methods available. Also, most of the symptoms of brain tumor are vague and could be contributed to many other abdominal conditions. Also, if the cancer has spread to the other organs the treatment becomes very difficult. So, there is an urgent need of method that will help radiologists in diagnosis of the brain tumor at an early stage. There is not much work done on brain tumor detection. It is found from the literature survey that brain tumor detection is done by using the symptoms of the disease and taking patient history but not using image processing.

In this project an attempt is made to detect the brain tumor detection from CT scan images. These images are pre-processed using image processing and deep learning techniques and then a basic classifier is used to classify the tumor area in the image and detect whether it is in the normal condition or it is having any tumor in the pancreas.

#### **1.1 INTRODUCTION TO DEEP LEARNING**

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. and it is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

While deep learning algorithms feature self-learning representations, they depend upon ANNs that mirror the way the brain computes information. During the training

process, algorithms use unknown elements in the input distribution to extract features, group objects, and discover useful data patterns. Much like training machines for self-learning, this occurs at multiple levels, using the algorithms to build the models. Deep learning models make use of several algorithms. While no one network is considered perfect, some algorithms are better suited to perform specific tasks. To choose the right ones, it's good to gain a solid understanding of all primary algorithms. Here is the list of most popular deep learning algorithms:

- a. Convolutional Neural Networks (CNNs)
- b. Long Short-Term Memory Networks (LSTMs)
- c. Recurrent Neural Networks (RNNs)
- d. Generative Adversarial Networks (GANs)
- e. Radial Basis Function Networks (RBFN)
- f. Deep Belief Networks (DBNs)
- g. BFNs
- h. Multilayer Perceptrons (MLPs)
- i. Self-Organizing Maps (SOMs)
- j. Restricted Boltzmann Machines (RBMs)

## **1.2 CONVOLUTIONAL NEUTRAL NETWORKS**

In deep learning, a convolutional neural network (**CNN/ConvNet**) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When you input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.

The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.

Based on the activation map of the final convolution layer, the classification layer outputs a set of confidence scores (values between 0 and 1) that specify how likely the image is to belong to a “class.” For instance, if you have a ConvNet that detects images containing tumors, the output of the final layer is the possibility that the input image contains any of those tumors.

## **CHAPTER-2**

### **LITERATURE SURVEY**

In the [1] Cancer incidence and deaths in the United States were projected for the most common cancer types for the years 2020 and 2030 based on changing demographics and the average annual percentage changes in incidence and death rates. Breast, prostate, and lung cancers will remain the top cancer diagnoses throughout this time, but thyroid cancer will replace colorectal cancer as the fourth leading cancer diagnosis by 2030, and melanoma and uterine cancer will become the fifth and sixth most common cancers, respectively. Lung cancer is projected to remain the top cancer killer throughout this time period. However, pancreas and liver cancers are projected to surpass breast, prostate, and colorectal cancers to become the second and third leading causes of cancer-related death by 2030, respectively. Advances in screening, prevention, and treatment can change cancer incidence and/or death rates, but it will require a concerted effort by the research and healthcare communities now to effect a substantial change for the future.

In the [2] Brain ductal adenocarcinoma is an aggressive malignancy with a high mortality rate. Proper determination of the extent of disease on imaging studies at the time of staging is one of the most important steps in optimal patient management. Given the variability in expertise and definition of disease extent among different practitioners as well as frequent lack of complete reporting of pertinent imaging findings at radiologic examinations, adoption of a standardized template for radiology reporting, using universally accepted and agreed on terminology for solid brain neoplasms, is needed. A consensus statement describing a standardized reporting template authored by a multi-institutional group of experts in brain ductal adenocarcinoma that included radiologists, gastroenterologists, and hepato-pancreato-biliary surgeons was developed under the joint sponsorship of the Society of Abdominal Radiologists and the American Brain Association. Adoption of this standardized imaging reporting template should improve the decision-making process for the management of patients with brain ductal adenocarcinoma by providing a complete, pertinent, and accurate reporting of disease staging to optimize treatment recommendations that can be

offered to the patient. Standardization can also help to facilitate research and clinical trial design by using appropriate and consistent staging by means of resectability

In the [3] Radiological imaging diagnosis plays important roles in clinical patient management. Deep learning with convolutional neural networks (CNNs) is recently gaining wide attention for its high performance in recognizing images. If CNNs realize their promise in the context of radiology, they are anticipated to help radiologists achieve diagnostic excellence and to enhance patient healthcare. Here, we discuss very recent developments in the field, including studies published in the current PLOS Medicine Special Issue on Machine Learning in Health and Biomedicine, with comment on expectations and planning for artificial intelligence (AI) in the radiology clinic.

In the [4] clinical discipline, photograph processing performs an important function in tumour diagnosis. Medical imaging is a developing and complex area. Many human beings be afflicted by brain tumours; this ailment could be very critical and dangerous. A brain tumour is a random boom of cells that proliferate uncontrollably. Early evaluation of brain tumours plays an important function in therapeutic capacity to enhance results and growth patient survival

In the [5] Our research objectives to use advanced generation for brain tumour detection and prognosis from brain MRI image facts. In this observe, various deep gaining knowledge of fashions which include VGG16/19, Alex Net, Google Net and Resnet were tried to illustrate their effectiveness and accuracy. Furthermore, the YOLO model became carried out to choose the best area of the tumour in the MR picture and carry out photo improvements to enhance the type accuracy

In the [6] Based at the examine, the ResNet101 model carried out the highest accuracy with an accuracy fee of ninety eight.6%. In addition, the VGG19 model confirmed the very best accuracy of ninety seven.2%. Other models have accuracy values of InceptionV3 94.Three%, DenseNet169 92.8% and Alex Net 89.5% respectively. This low success fee suggests that the structures utilized in these fashions aren't suitable for MR imaging studies in comparison to other structures. For this motive, it was concluded that the usage of the ResNet architecture to detect tumors from mind MR snap shots is more powerful than other fashions

## **CHAPTER-3**

### **REQUIREMENT ANALYSIS**

#### **3.1 EXISTING SYSTEM**

- a. Numerous methods have been proposed to cope with noisy label classification for natural images. Ren et al. propose a technique to assign weights to training samples by using an additional clean validation set. Their intuition is to apply smaller weights to noisy samples and increase the weights of clean training samples to improve the gradient update. There is relatively limited work on the development and application of noisy label classification methods in medical imaging data.
- b. Dgani et al. model label noise as a part of the deep learning network to recover true labels of noisy samples for the task of classifying breast micro-calcifications in multi-view mammograms.

##### **3.1.1 DISADVANTAGES OF EXISTING SYSTEM**

- a. There is not much work done on brain tumor detection.
- b. It is found from the literature survey that brain tumor detection is done using the symptoms of the disease and by taking patient history but not using image processing.
- c. So, here we can't predict the disease in the early stage ,it will become difficult when it goes to further states,
- d. Detecting the disease is very slow.
- e.It did not use any image processing techniques to predict disease at the early stage.

#### **3.2 PROPOSED SYSTEM**

In this project, an attempt is made to detect brain tumor from CT scan images. These images are preprocessed using image processing techniques and then CNN model architecture is used to classify the tumor area in the image. The dataset is taken from

the cancer imaging achieve repository. These CT scan images of brain tumor are given as inputs to the system. These images are different format, so we have converted into jpg and also the image size is too huge, so we have used few of them only. Then classification is done using CNN model architecture. In this, the system is made learn according to the classes defined in the image. Once the system learns the classification done based on the features given to it, it can then classify the test data into one of those classes.

We trained a CNN using contrast enhanced-CT images of patients to distinguish brain cancer from healthy pancreases. CNN achieved excellent accuracy and improved sensitivity compared with radiologist interpretation in independent test sets, with acceptable performance in a test set obtained from patients of various types. These results provide the first solid proof of concept that CNN can capture the elusive CT features of brain cancer to assist and supplement radiologists in the detection and diagnosis of brain cancer.

### **3.2.1 ADVANTAGES OF PROPOSED SYSTEM:**

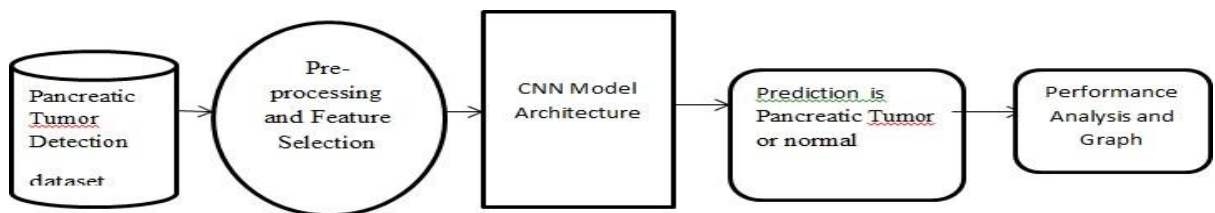
- a. Deep learning with convolutional neural networks (CNNs) has shown great promise in medical image analysis.
- b. The construction of neural networks is based on a stack of neurons composed of activation functions and parameters to extract and integrate features from the images and establish a model that captures the complex relationship between images and diagnoses. CNN has been reported to achieve a high accuracy in the imaging diagnosis.
- c. CNN can accurately differentiate brain cancer from non-cancerous pancreas, and with improvements might accommodate variations in patient race and ethnicity and imaging parameters that are inevitable in real-world clinical practice. CNN holds promise for developing computer-aided detection and diagnosis tools for brain cancer to supplement radiologist interpretation.

## CHAPTER 4

### DESCRIPTION OF PROPOSED SYSTEM

#### 4.1 PROPOSED ARCHITECTURE:

- It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
- It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
- This shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.



*Figure 4.1 Proposed SVM-CNN Architecture*

#### 4.2 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy.

#### 4.3 OUTPUT DESIGN

A quality output is one which meets the requirements of the end user and presents the information clearly. In any system the results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

The output form of an information system should accomplish one or more of the following objectives.

Convey information about past activities, current status or projections of the Future.

Signal important events, opportunities, problems, or warnings.

Trigger an action.

Confirm an action.

#### **4.4 SYSTEM REQUIREMENTS**

##### **4.4.1 HARDWARE REQUIREMENTS:**

|               |   |                          |
|---------------|---|--------------------------|
| System        | : | Intel core i5 Processor. |
| Hard Disk     | : | 1000 GB.                 |
| Monitor       | : | 15" LED                  |
| Input Devices | : | Keyboard, Mouse          |
| Ram           | : | 8 GB                     |

##### **4.4.2 SOFTWARE REQUIREMENTS:**

|                  |   |             |
|------------------|---|-------------|
| Operating system | : | Windows 10. |
| Coding Language  | : | Python      |
| Web Framework    | : | Flask.      |

#### **4.5 SYSTEM DESIGN:**

##### **i. IMAGE PROCESSING: What is Image Processing?**

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually, Image Processing systems include treating images as two-

dimensional signals while applying already set signal processing methods to them. It is among rapidly growing technologies today, with its applications in various aspects of business. Image Processing forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps.

- Importing the image with optical scanner or by digital photography.
- Analyzing and manipulating the image which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite photographs.
- Output is the last stage in which result can be altered image or report that is based on image analysis.

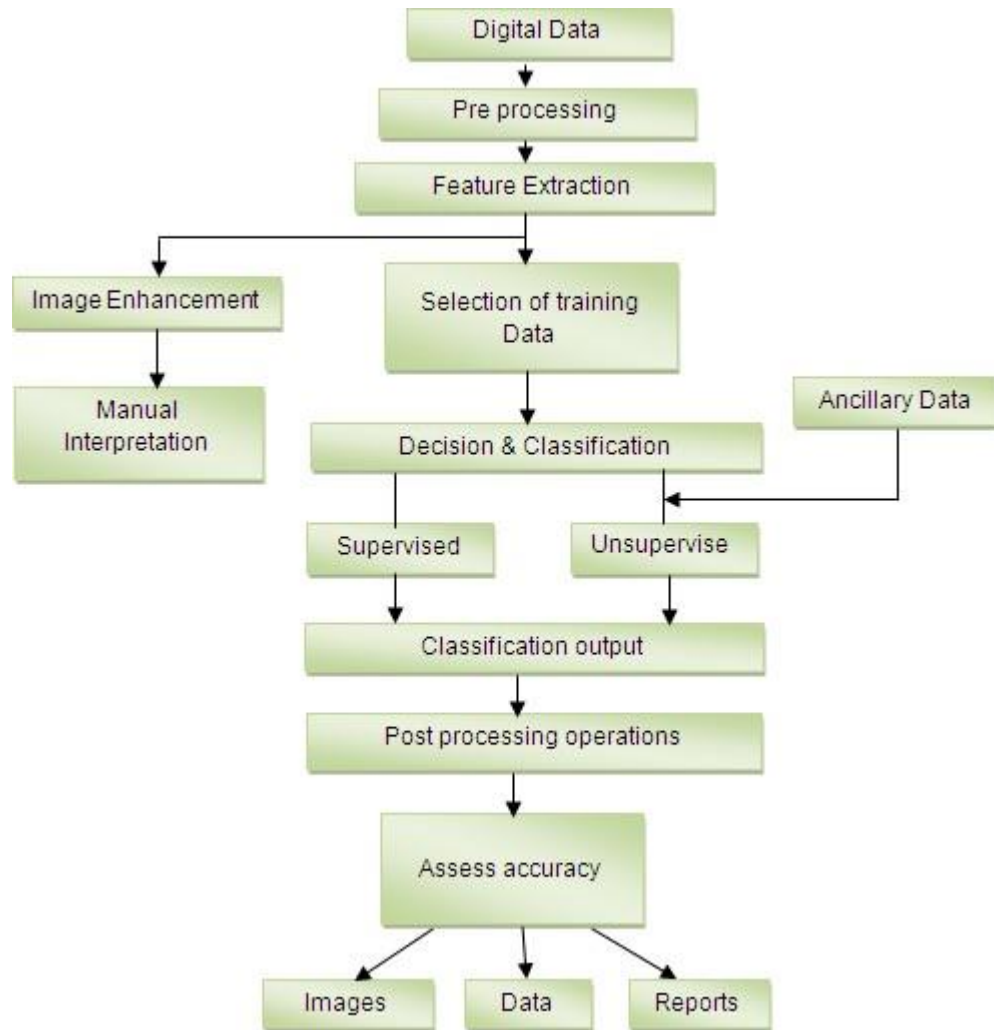
Purpose of Image processing

The purpose of image processing is divided into 5 groups. They are:

1. Visualization - Observe objects that are not visible.
2. Image sharpening and restoration - To create a better image.
3. Image retrieval - Seek for the image of interest.
4. Measurement of pattern – Measures various objects in an image.
5. Image Recognition – Distinguish the objects in an image.

Types:

The two types of methods used for Image Processing are Analog and Digital Image Processing. Analog or visual techniques of image processing can be used for hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. The image processing is not just confined to area that has to be studied but on knowledge of analyst. Association is another important tool in image processing through visual techniques. So analysts apply a combination of personal knowledge and collateral data to image processing. Digital Processing techniques help in manipulation of the digital images by using computers. As raw data from imaging sensors from satellite platform contains deficiencies. To get over such flaws and to get originality of information, it has to undergo various phases of processing. The three general phases that all types of data have to undergo while using digital technique are Pre- processing, enhancement and display, information extraction.



*Figure 4.2 Processing of Image – Flow diagram*

## **ii. IMAGE PROCESSING CONCEPTS- Binary Images**

Binary images are images whose pixels have only two possible intensity values. They are normally displayed as black and white. Numerically, the two values are often 0 for black, and either 1 or 255 for white.

Binary images are often produced by thresholding a grayscale or color image, in order to separate an object in the image from the background. The color of the object (usually white) is referred to as the foreground color. The rest (usually black) is referred to as the background color. However, depending on the image which is to be

thresholded, this polarity might be inverted, in which case the object is displayed with 0 and the background is with a non-zero value.

Some morphological operators assume a certain polarity of the binary input image so that if we process an image with inverse polarity the operator will have the opposite effect. For example, if we apply a closing operator to a black text on white background, the text will be opened.

### **iii. COLOR IMAGES**

It is possible to construct (almost) all visible colors by combining the three primary colors red, green and blue, because the human eye has only three different color receptors, each of them sensible to one of the three colors. Different combinations in the stimulation of the receptors enable the human eye to distinguish approximately 350000 colors. A RGB color image is a multi-spectral image with one band for each color red, green and blue, thus producing a weighted combination of the three primary colors for each pixel.

A full 24-bit color image contains one 8-bit value for each color, thus being able to display  $2^{24} = 16777216$  different colors.

However, it is computationally expensive and often not necessary to use the full 24-bit image to store the color for each pixel. Therefore, the color for each pixel is often encoded in a single byte, resulting in an 8-bit color image. The process of reducing the color representation from 24-bits to 8-bits, known as color quantization, restricts the number of possible colors to 256. However, there is normally no visible difference between a 24-color image and the same image displayed with 8 bits. An 8-bit color images are based on colormaps, which are look-up tables taking the 8-bit pixel value as index and providing an output value for each color.

### **iv. 8-BIT COLOR IMAGES**

Full RGB color requires that the intensities of three-color components be specified for each and every pixel. It is common for each component intensity to be stored as an 8-bit integer, and so each pixel requires 24 bits to completely and accurately specify its

color. If this is done, then the image is known as a 24-bit color image. However, there are two problems with this approach:

Storing 24 bits for every pixel leads to very large image files that with current technology are cumbersome to store and manipulate. For instance a 24-bit 512×512 image takes up 750KB in uncompressed form.

Many monitor displays use colormaps with 8-bit index numbers, meaning that they can only display 256 different colors at any one time. Thus it is often wasteful to store more than 256 different colors in an image anyway, since it will not be possible to display them all on screen.

Because of this, many image formats (e.g. 8-bit GIF and TIFF) use 8-bit colormaps to restrict the maximum number of different colors to 256. Using this method, it is only necessary to store an 8-bit index into the colormap for each pixel, rather than the full 24-bit color value. Thus 8-bit image formats consist of two parts: a colormap describing what colors are present in the image, and the array of index values for each pixel in the image.

When a 24-bit full color image is turned into an 8-bit image, it is usually necessary to throw away some of the colors, a process known as color quantization. This leads to some degradation in image quality, but in practice the observable effect can be quite small, and in any case, such degradation is inevitable if the image output device (e.g. screen or printer) is only capable of displaying 256 colors or less.

The use of 8-bit images with colormaps does lead to some problems in image processing. First of all, each image has to have its own colormap, and there is usually no guarantee that each image will have exactly the same colormap. Thus on 8-bit displays it is frequently impossible to correctly display two different color images that have different colormaps at the same time. Note that in practice 8-bit images often use reduced size colormaps with less than 256 colors in order to avoid this problem.

Another problem occurs when the output image from an image processing operation contains different colors to the input image or images. This can occur very easily, as for instance when two color images are added together pixel-by-pixel. Since the output image contains different colors from the input images, it ideally needs a new colormap,

different from those of the input images, and this involves further color quantization which will degrade the image quality. Hence the resulting output is usually only an approximation of the desired output. Repeated image processing operations will continually degrade the image colors. And of course we still have the problem that it is not possible to display the images simultaneously with each other on the same 8-bit display.

Because of these problems it is to be expected that as computer storage and processing power become cheaper, there will be a shift away from 8-bit images and towards full 24-bit image processing.

## **v. 24-BIT COLOR IMAGES**

Full RGB color requires that the intensities of three color components be specified for each and every pixel. It is common for each component intensity to be stored as an 8-bit integer, and so each pixel requires 24 bits to completely and accurately specify its color. Image formats that store a full 24 bits to describe the color of each and every pixel are therefore known as 24-bit color images.

Using 24 bits to encode color information allows  $2^{24} = 16777216$  different colors to be represented, and this is sufficient to cover the full range of human color perception fairly well.

The term 24-bit is also used to describe monitor displays that use 24 bits per pixel in their display memories, and which are hence capable of displaying a full range of colors.

There are also some disadvantages to using 24-bit images. Perhaps the main one is that it requires three times as much memory, disk space and processing time to store and manipulate 24-bit color images as compared to 8-bit color images. In addition, there is often not much point in being able to store all those different colors if the final output device (e.g. screen or printer) can only actually produce a fraction of them. Since it is possible to use colormaps to produce 8-bit color images that look almost as good, at the time of writing 24-bit displays are relatively little used. However it is to be expected that as the technology becomes

## vi. COLOR QUANTIZATION

Color quantization is applied when the color information of an image is to be reduced. The most common case is when a 24-bit color image is transformed into an 8-bit color image.

Two decisions have to be made:

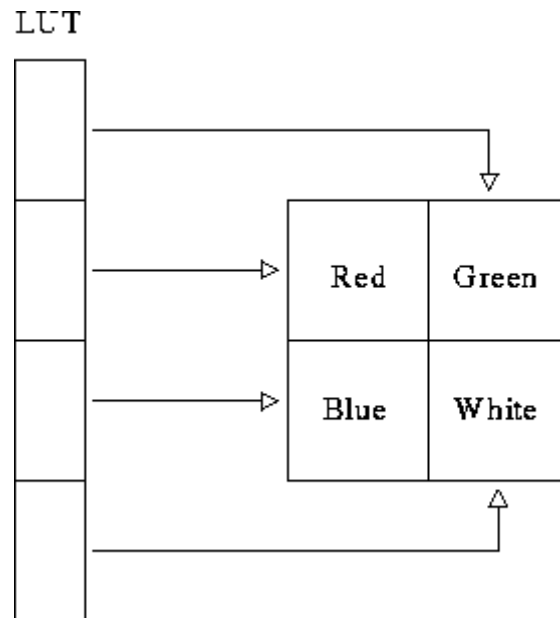
1. which colors of the larger color set remain in the new image, and
2. how are the discarded colors mapped to the remaining ones.

The simplest way to transform a 24-bit color image into 8 bits is to assign 3 bits to red and green and 2 bits to blue (blue has only 2 bits, because of the eye's lower sensitivity to this color). This enables us to display 8 different shades of red and green and 4 of blue. However, this method can yield only poor results. For example, an image might contain different shades of blue which are all clustered around a certain value such that only one shade of blue is used in the 8-bit image and the remaining three blues are not used.

Alternatively, since 8-bit color images are displayed using a colormap, we can assign any arbitrary color to each of the 256 8-bit values and we can define a separate colormap for each image. This enables us perform a color quantization adjusted to the data contained in the image. One common approach is the popularity algorithm, which creates a histogram of all colors and retains the 256 most frequent ones. Another approach, known as the median-cut algorithm, yields even better results but also needs more computation time. This technique recursively fits a box around all colors used in the RGB color space which it splits at the median value of its longest side. The algorithm stops after 255 recursions. All colors in one box are mapped to the centroid of this box.

All above techniques restrict the number of displayed colors to 256. A technique of achieving additional colors is to apply a variation of half-toning used for gray scale images, thus increasing the color resolution at the cost of spatial resolution. The 256 values of the colormap are divided into four sections containing 64 different values of red, green, blue and white. As can be seen in Figure 1, a 2×2 pixel area is grouped

together to represent one composite color, each of the four pixels displays either one of the primary colors or white. In this way, the number of possible colors is increased from 256 to  $64^4$ .



*Figure 4.3 Quantization of Colors*

A 2x2-pixel area displaying one composite color cheaper, their use in image processing will grow.

## **vii. CONVOLUTION**

Convolution is a simple mathematical operation which is fundamental to many common image processing operators. Convolution provides a way of 'multiplying together' two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality. This can be used in image processing to implement operators whose output pixel values are simple linear combinations of certain input pixel values.

In an image processing context, one of the input arrays is normally just a Gray level image. The second array is usually much smaller, and is also two-dimensional (although it may be just a single pixel thick), and is known as the kernel. Figure 1 shows an example image and kernel that we will use to illustrate convolution.

|                       |                       |                       |                       |                       |                       |                       |                       |                       |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| <b>I<sub>11</sub></b> | <b>I<sub>12</sub></b> | <b>I<sub>13</sub></b> | <b>I<sub>14</sub></b> | <b>I<sub>15</sub></b> | <b>I<sub>16</sub></b> | <b>I<sub>17</sub></b> | <b>I<sub>18</sub></b> | <b>I<sub>19</sub></b> |
| <b>I<sub>21</sub></b> | <b>I<sub>22</sub></b> | <b>I<sub>23</sub></b> | <b>I<sub>24</sub></b> | <b>I<sub>25</sub></b> | <b>I<sub>26</sub></b> | <b>I<sub>27</sub></b> | <b>I<sub>28</sub></b> | <b>I<sub>29</sub></b> |
| <b>I<sub>31</sub></b> | <b>I<sub>32</sub></b> | <b>I<sub>33</sub></b> | <b>I<sub>34</sub></b> | <b>I<sub>35</sub></b> | <b>I<sub>36</sub></b> | <b>I<sub>37</sub></b> | <b>I<sub>38</sub></b> | <b>I<sub>39</sub></b> |
| <b>I<sub>41</sub></b> | <b>I<sub>42</sub></b> | <b>I<sub>43</sub></b> | <b>I<sub>44</sub></b> | <b>I<sub>45</sub></b> | <b>I<sub>46</sub></b> | <b>I<sub>47</sub></b> | <b>I<sub>48</sub></b> | <b>I<sub>49</sub></b> |
| <b>I<sub>51</sub></b> | <b>I<sub>52</sub></b> | <b>I<sub>53</sub></b> | <b>I<sub>54</sub></b> | <b>I<sub>55</sub></b> | <b>I<sub>56</sub></b> | <b>I<sub>57</sub></b> | <b>I<sub>58</sub></b> | <b>I<sub>59</sub></b> |
| <b>I<sub>61</sub></b> | <b>I<sub>62</sub></b> | <b>I<sub>63</sub></b> | <b>I<sub>64</sub></b> | <b>I<sub>65</sub></b> | <b>I<sub>66</sub></b> | <b>I<sub>67</sub></b> | <b>I<sub>68</sub></b> | <b>I<sub>69</sub></b> |

|                       |                       |                       |
|-----------------------|-----------------------|-----------------------|
| <b>K<sub>11</sub></b> | <b>K<sub>12</sub></b> | <b>K<sub>13</sub></b> |
| <b>K<sub>21</sub></b> | <b>K<sub>22</sub></b> | <b>K<sub>23</sub></b> |

Figure 4.4 Convolution of Colors

Figure 4.4 An example small image (left) and kernel (right) to illustrate convolution. The labels within each grid square are used to identify each square.

The convolution is performed by sliding the kernel over the image, generally starting at the top left corner, so as to move the kernel through all the positions where the kernel fits entirely within the boundaries of the image. (Note that implementations differ in what they do at the edges of images, as explained below.) Each kernel position corresponds to a single output pixel, the value of which is calculated by multiplying together the kernel value and the underlying image pixel value for each of the cells in the kernel, and then adding all these numbers together. So, in our example, the value of the bottom right pixel in the output image will be given by:

$$O_{57} = I_{57}K_{11} + I_{58}K_{12} + I_{59}K_{13} + I_{67}K_{21} + I_{68}K_{22} + I_{69}K_{23} \quad [1]$$

If the image has M rows and N columns, and the kernel has m rows and n columns, then the size of the output image will have M - m + 1 rows, and N - n + 1 columns.

Mathematically we can write the convolution as:

$$O(i, j) = \sum_{k=1}^m \sum_{l=1}^n I(i + k - 1, j + l - 1)K(k, l) \quad [2]$$

where i runs from 1 to M - m + 1 and j runs from 1 to N - n + 1.

Note that many implementations of convolution produce a larger output image than this because they relax the constraint that the kernel can only be moved to positions where it fits entirely within the image. Instead, these implementations typically slide the kernel to all positions where just the top left corner of the kernel is within the image. Therefore the kernel 'overlaps' the image on the bottom and right edges. One advantage of this approach is that the output image is the same size as the input image. Unfortunately, in order to calculate the output pixel values for the bottom and right edges of the image, it is necessary to invent input pixel values for places where the kernel extends off the end of the image. Typically, pixel values of zero are chosen for regions outside the true image, but this can often distort the output image at these places. Therefore in general if you are using a convolution implementation that does this, it is better to clip the image to remove these spurious regions. Removing  $n - 1$  pixels from the right hand side and  $m - 1$  pixels from the bottom will fix things.

Convolution can be used to implement many different operators, particularly spatial filters and feature detectors. Examples include Gaussian smoothing and the Sobel edge detector.

### **viii. DISTANCE METRICS**

It is often useful in image processing to be able to calculate the distance between two pixels in an image, but this is not as straightforward as it seems. The presence of the pixel grid makes several so-called distance metrics possible which often give different answers to each other for the distance between the same pair of points. We consider the three most important ones.

### **ix EUCLIDEAN DISTANCE**

This is the familiar straight line distance that most people are familiar with. If the two pixels that we are considering have coordinates  $(x_1, y_1)$  and  $(x_2, y_2)$ , then the Euclidean distance is given by:

$$D_{Euclid} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad [3]$$

### **x. CITY BLOCK DISTANCE**

Also known as the Manhattan distance. This metric assumes that in going from one pixel to the other it is only possible to travel directly along pixel grid lines. Diagonal moves are not allowed. Therefore the 'city block' distance is given by:

$$D_{City} = |x_2 - x_1| + |y_2 - y_1| \quad [4]$$

#### xi. CHESSBOARD DISTANCE

This metric assumes that you can make moves on the pixel grid as if you were a King making moves in chess, *i.e.* a diagonal move counts the same as a horizontal move. This means that the metric is given by:

$$D_{Chess} = \max(|x_2 - x_1|, |y_2 - y_1|) \quad [5]$$

Note that the last two metrics are usually much faster to compute than the Euclidean metric and so are sometimes used where speed is critical but accuracy is not too important.

#### xii. DITHERING

Dithering is an image display technique that is useful for overcoming limited display resources. The word dither refers to a random or semi-random perturbation of the pixel values.

Two applications of this techniques are particularly useful:

🟡 Low quantization display: When images are quantized to a few bits (e.g. 3) then only a limited number of gray levels are used in the display of the image. If the scene is smoothly shaded, then the image display will generate rather distinct boundaries around the edges of image regions when the original scene intensity moves from one quantization level to the next. To eliminate this effect, one dithering technique adds random noise (with a small range of values) to the input signal before quantization into the output range. This randomizes the quantization of the pixels at the original quantization boundary, and thus pixels make a more gradual transition from neighborhoods containing 100% of the first quantization level to neighborhoods containing 100% of the second quantization level.

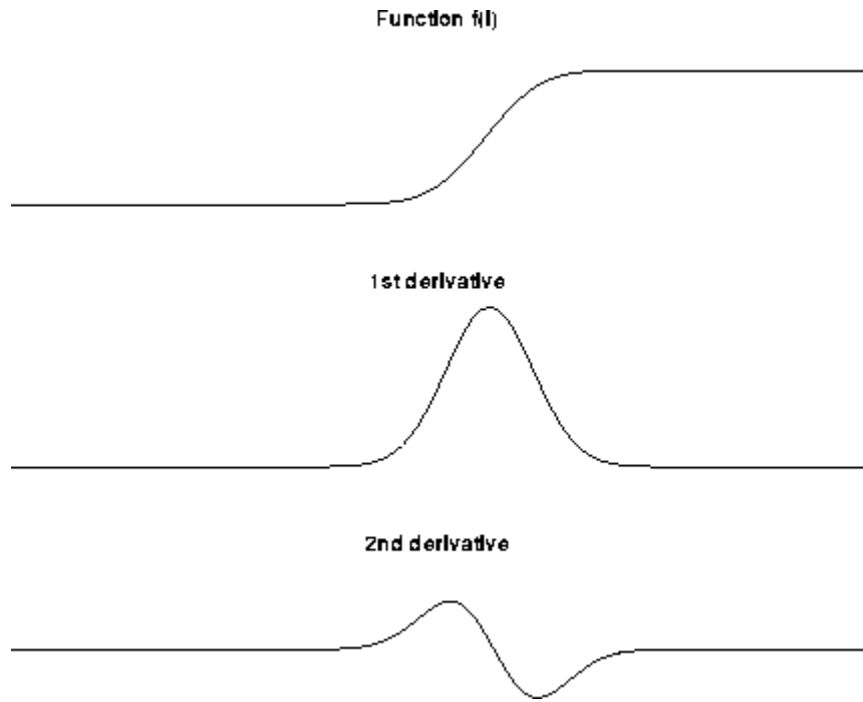
🟡 Limited color display: When fewer colors are able to be displayed (e.g. 256) than are present in the input image (e.g. 24-bit color), then patterns of adjacent pixels are used to simulate the appearance of the unrepresented colors.

### **xiii. EDGE DETECTORS**

Edges are places in the image with strong intensity contrast. Since edges often occur at image locations representing object boundaries, edge detection is extensively used in image segmentation when we want to divide the image into areas corresponding to different objects. Representing an image by its edges has the further advantage that the amount of data is reduced significantly while retaining most of the image information.

Since edges consist of mainly high frequencies, we can, in theory, detect edges by applying a high pass frequency filter in the Fourier domain or by convolving the image with an appropriate kernel in the spatial domain. In practice, edge detection is performed in the spatial domain, because it is computationally less expensive and often yields better results.

Since edges correspond to strong illumination gradients, we can highlight them by calculating the derivatives of the image. This is illustrated for the one-dimensional case in Figure 4.5.



*Figure 4.5 First and Second derivative of an edge illustrated in one dimension.*

We can see that the position of the edge can be estimated with the maximum of the 1st derivative or with the zero-crossing of the 2nd derivative. Therefore, we want to find a technique to calculate the derivative of a two-dimensional image. For a discrete one-dimensional function  $f(i)$ , the first derivative can be approximated by

$$\frac{df(i)}{d(i)} = f(i+1) - f(i) \quad [6]$$

Calculating this formula is equivalent to convolving the function with  $[-1 \ 1]$ . Similarly, the 2nd derivative can be estimated by convolving  $f(i)$  with  $[1 \ -2 \ 1]$ .

Different edge detection kernels which are based on the above formula enable us to calculate either the 1st or the 2nd derivative of a two-dimensional image. There are two common approaches to estimate the 1st derivative in a two-dimensional image, Prewitt compass edge detection and gradient edge detection.

Prewitt compass edge detection involves convolving the image with a set of (usually 8) kernels, each of which is sensitive to a different edge orientation. The kernel producing the maximum response at a pixel location determines the edge magnitude

and orientation. Different sets of kernels might be used: examples include Prewitt, Sobel, Kirsch and Robinson kernels.

Gradient edge detection is the second and more widely used technique. Here, the image is convolved with only two kernels, one estimating the gradient in the x-direction,  $G_x$ , the other the gradient in the y-direction,  $G_y$ . The absolute gradient magnitude is then given by

$$|G| = \sqrt{G_x^2 + G_y^2} \quad [7]$$

and is often approximated with

$$|G| = |G_x| + |G_y| \quad [8]$$

In many implementations, the gradient magnitude is the only output of a gradient edge detector, however the edge orientation might be calculated with

The most common kernels used for the gradient edge detector are the Sobel, Roberts Cross and Prewitt operators.

After having calculated the magnitude of the 1st derivative, we now have to identify those pixels corresponding to an edge. The easiest way is to threshold the gradient image, assuming that all pixels having a local gradient above the threshold must represent an edge. An alternative technique is to look for local maxima in the gradient image, thus producing one pixel wide edges. A more sophisticated technique is used by the Canny edge detector. It first applies a gradient edge detector to the image and then finds the edge pixels using non-maximal suppression and hysteresis tracking.

An operator based on the 2nd derivative of an image is the Marr edge detector, also known as zero crossing detector. Here, the 2nd derivative is calculated using a Laplacian of Gaussian (LoG) filter. The Laplacian has the advantage that it is an isotropic measure of the 2nd derivative of an image, i.e. the edge magnitude is obtained independently from the edge orientation by convolving the image with only one kernel. The edge positions are then given by the zero-crossings in the LoG image. The scale of the edges which are to be detected can be controlled by changing the variance of the Gaussian.

A general problem for edge detection is its sensitivity to noise, the reason being that calculating the derivative in the spatial domain corresponds to accentuating high frequencies and hence magnifying noise. This problem is addressed in the Canny and Marr operators by convolving the image with a smoothing operator (Gaussian) before calculating the derivative.

#### **xiv. FREQUENCY DOMAIN**

For simplicity, assume that the image  $I$  being considered is formed by projection from scene  $S$  (which might be a two- or three-dimensional scene, etc.).

The frequency domain is a space in which each image value at image position  $F$  represents the amount that the intensity values in image  $I$  vary over a specific distance related to  $F$ . In the frequency domain, changes in image position correspond to changes in the spatial frequency, (or the rate at which image intensity values) are changing in the spatial domain image  $I$ .

For example, suppose that there is the value 20 at the point that represents the frequency 0.1 (or 1 period every 10 pixels). This means that in the corresponding spatial domain image  $I$  the intensity values vary from dark to light and back to dark over a distance of 10 pixels, and that the contrast between the lightest and darkest is 40 gray levels (2 times 20).

The spatial frequency domain is interesting because: 1) it may make explicit periodic relationships in the spatial domain, and 2) some image processing operators are more efficient or indeed only practical when applied in the frequency domain.

In most cases, the Fourier Transform is used to convert images from the spatial domain into the frequency domain and vice-versa.

A related term used in this context is spatial frequency, which refers to the (inverse of the) periodicity with which the image intensity values change. Image features with high spatial frequency (such as edges) are those that change greatly in intensity over short image distances.

#### **xv. GRAYSCALE IMAGES**

A grayscale (or graylevel) image is simply one in which the only colors are shades of gray. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel. In fact a 'gray' color is one in which the red, green and blue components all have equal intensity in RGB space, and so it is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full color image.

Often, the grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white. If the levels are evenly spaced then the difference between successive graylevels is significantly better than the graylevel resolving power of the human eye.

Grayscale images are very common, in part because much of today's display and image capture hardware can only support 8-bit images. In addition, grayscale images are entirely sufficient for many tasks and so there is no need to use more complicated and harder-to-process color images.

## **xvi. IMAGE EDITING SOFTWARE**

There is a huge variety of software for manipulating images in various ways. Much of this software can be grouped under the heading image processing software, and the bulk of this reference is concerned with that group.

Another very important category is what we call image editing software. This group includes painting programs, graphic art packages and so on. They are often useful in conjunction with image processing software packages, in situations where direct immediate interaction with an image is the easiest way of achieving something. For instance, if a region of an image is to be masked out for subsequent image processing, it may be easiest to create the mask using an art package by directly drawing on top of the original image. The mask used in the description of the AND operator was created this way for instance. Art packages also often allow the user to move sections of the images around and brighten or darken selected regions interactively. Few dedicated image processing packages offer the same flexibility and ease of use in this respect.

## xvii. IDEMPOTENCE

Some operators have the special property that applying them more than once to the same image produces no further change after the first application. Such operators are said to be idempotent. Examples include the morphological operators opening and closing.

## xviii. ISOTROPIC OPERATORS

An isotropic operator in an image processing context is one which applies equally well in all directions in an image, with no particular sensitivity or bias towards one particular set of directions (e.g. compass directions). A typical example is the zero crossing edge detector which responds equally well to edges in any orientation. Another example is Gaussian smoothing. It should be borne in mind that although an operator might be isotropic in theory, the actual implementation of it for use on a discrete pixel grid may not be perfectly isotropic. An example of this is a Gaussian smoothing filter with very small standard deviation on a square grid.

## xix. KERNEL

A kernel is a (usually) smallish matrix of numbers that is used in image convolutions. Differently sized kernels containing different patterns of numbers give rise to different results under convolution. For instance, Figure 1 shows a 3×3 kernel that implements a mean filter.

|   |   |   |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

Set of coordinate points =  
{ (-1, -1), (0, -1), (1, -1),  
(-1, 0), (0, 0), (1, 0),  
(-1, 1), (0, 1), (1, 1) }

*Figure 4.6 Convolution kernel for a mean filter with 3×3 neighborhood.*

The word 'kernel' is also commonly used as a synonym for 'structuring element', which is a similar object used in mathematical morphology. A structuring element differs from a kernel in that it also has a specified origin. This sense of the word 'kernel' is not used in HIPR.

## xx. LOGICAL OPERATORS

Logical operators are generally derived from Boolean algebra, which is a mathematical way of manipulating the truth values of concepts in an abstract way without bothering about what the concepts actually mean. The truth value of a concept in Boolean value can have just one of two possible values: true or false. Boolean algebra allows you to represent things like:

The block is both red and large by something like:  $A \text{ AND } B$

where A represents 'The block is red', and B represents 'The block is large'. Now each of these sub-phrases has its own truth value in any given situation: each sub-phrase is either true or false. Moreover, the entire composite phrase also has a truth value: it is true if both sub-phrases are true, and false in any other case. We can write this AND combination rule (and its dual operation NAND) using a truth-table as shown in Figure 1, in which we conventionally represent true by 1, and false by zero.

| A | B | Q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**AND**

| A | B | Q |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**NAND**

*Figure 4.7 Truth-tables for AND NAND*

The left hand table shows each of the possible combinations of truth values

of A and B, and the resulting truth value of A AND B. Similar truth-tables can be set up for the other logical operators: NAND, OR, NOR, XOR, XNOR and NOT.

Turning now to an image processing context, the pixel values in a binary image, which are either 0 or 1, can be interpreted as truth values as above. Using this convention, we can carry out logical operations on images simply by applying the truth-table combination rules to the pixel values from a pair of input images (or a single input image in the case of NOT). Normally, corresponding pixels from each of two identically sized binary input images are compared to produce the output image, which is another binary image of the same size. As with other image arithmetic operations, it is also possible to logically combine a single input image with a constant logical value, in which case each pixel in the input image is compared to the same constant in order to produce the corresponding output pixel. See the individual logical operator descriptions for examples of these operations.

Logical operations can also be carried out on images with integer pixel values. In this extension the logical operations are normally carried out in bitwise fashion on binary representations of those integers, comparing corresponding bits with corresponding bits to produce the output pixel value. For instance, suppose that we wish to XOR the integers 47 and 255 together using 8-bit integers. 47 is 00101111 in binary and 255 is 11111111. XORing these together in bitwise fashion, we have 11010000 in binary or 208 in decimal.

Note that not all implementations of logical operators work in such bitwise fashion. For instance, some will treat zero as false and any non-zero value as true and will then apply the conventional 1-bit logical functions to derive the output image. The output may be a simple binary image itself, or it may be a graylevel image formed perhaps by multiplying what would be the binary output image (containing 0's and 1's) with one of the input images.

## **xxi. LOOK-UP TABLES AND COLORMAPS**

Look-Up Tables or LUTs are fundamental to many aspects of image processing. An LUT is simply a table of cross-references linking index numbers to output values. The most common use is to determine the colors and intensity values with which a

particular image will be displayed, and in this context the LUT is often called simply a colormap.

The idea behind the colormap is that instead of storing a definite color for each pixel in an image, for instance in 24-bit RGB format, each pixel's value is instead treated as an index number into the colormap. When the image is to be displayed or otherwise processed, the colormap is used to look up the actual colors corresponding to each index number. Typically, the output values stored in the LUT would be RGB color values.

There are two main advantages to doing things this way. Firstly, the index number can be made to use fewer bits than the output value in order to save storage space. For instance an 8-bit index number can be used to look up a 24-bit RGB color value in the LUT. Since only the 8-bit index number needs to be stored for each pixel, such 8-bit color images take up less space than a full 24-bit image of the same size. Of course the image can only contain 256 different colors (the number of entries in an 8-bit LUT), but this is sufficient for many applications and usually the observable image degradation is small.

Secondly the use of a color table allows the user to experiment easily with different color labeling schemes for an image.

One disadvantage of using a colormap is that it introduces additional complexity into an image format. It is usually necessary for each image to carry around its own colormap, and this LUT must be continually consulted whenever the image is displayed or processed.

Another problem is that in order to convert from a full color image to (say) an 8-bit color image using a color image, it is usually necessary to throw away many of the original colors, a process known as color quantization. This process is lossy, and hence the image quality is degraded during the quantization process. Additionally, when performing further image processing on such images, it is frequently necessary to generate a new colormap for the new images, which involves further color quantization, and hence further image degradation.

As well as their use in colormaps, LUTs are often used to remap the pixel values within an image. This is the basis of many common image processing point operations such as thresholding, gamma correction and contrast stretching. The process is often referred to as anamorphosis.

## **xxii. MASKING**

A mask is a binary image consisting of zero- and non-zero values. If a mask is applied to another binary or to a grayscale image of the same size, all pixels which are zero in the mask are set to zero in the output image. All others remain unchanged.

Masking can be implemented either using pixel multiplication or logical AND, the latter in general being faster.

Masking is often used to restrict a point or arithmetic operator to an area defined by the mask. We can, for example, accomplish this by first masking the desired area in the input image and processing it with the operator, then masking the original input image with the inverted mask to obtain the unprocessed area of the image and finally recombining the two partial images using image addition. An example can be seen in the worksheet on the logical AND operator. In some image processing packages, a mask can directly be defined as an optional input to a point operator, so that automatically the operator is only applied to the pixels defined by the mask .

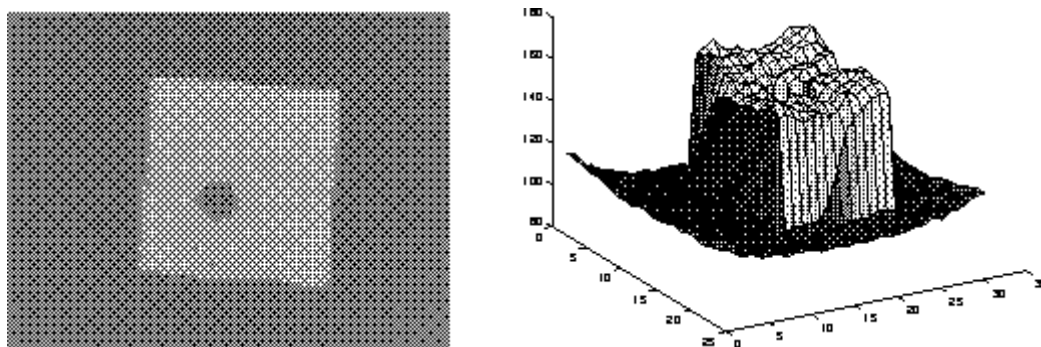
## **xxiii. MATHEMATICAL MORPHOLOGY**

The field of mathematical morphology contributes a wide range of operators to image processing, all based around a few simple mathematical concepts from set theory. The operators are particularly useful for the analysis of binary images and common usages include edge detection, noise removal, image enhancement and image segmentation.

The two most basic operations in mathematical morphology are erosion and dilation. Both of these operators take two pieces of data as input: an image to be eroded or dilated, and a structuring element (also known as a kernel). The two pieces of input data are each treated as representing sets of coordinates in a way that is slightly different for binary and grayscale images.

For a binary image, white pixels are normally taken to represent foreground regions, while black pixels denote background. (Note that in some implementations this convention is reversed, and so it is very important to set up input images with the correct polarity for the implementation being used). Then the set of coordinates corresponding to that image is simply the set of two-dimensional Euclidean coordinates of all the foreground pixels in the image, with an origin normally taken in one of the corners so that all coordinates have positive elements.

For a grayscale image, the intensity value is taken to represent height above a base plane, so that the grayscale image represents a surface in three-dimensional Euclidean space. Figure 1 shows such a surface. Then the set of coordinates associated with this image surface is simply the set of three-dimensional Euclidean coordinates of all the points within this surface and also all points below the surface, down to the base plane. Note that even when we are only considering points with integer coordinates, this is a lot of points, so usually algorithms are employed that do not need to consider all the points.



*Figure 4.8 Simple gray level image and the corresponding surface in image space*

The structuring element is already just a set of point coordinates (although it is often represented as a binary image). It differs from the input image coordinate set in that it is normally much smaller, and its coordinate origin is often not in a corner, so that some coordinate elements will have negative values. Note that in many implementations of morphological operators, the structuring element is assumed to be a particular shape (e.g. a  $3 \times 3$  square) and so is hardwired into the algorithm.

Binary morphology can be seen as a special case of gray level morphology in which the input image has only two gray levels at values 0 and 1.

Erosion and dilation work (at least conceptually) by translating the structuring element to various points in the input image, and examining the intersection between the translated kernel coordinates and the input image coordinates. For instance, in the case of erosion, the output coordinate set consists of just those points to which the origin of the structuring element can be translated, while the element still remains entirely 'within' the input image.

Virtually all other mathematical morphology operators can be defined in terms of combinations of erosion and dilation along with set operators such as intersection and union. Some of the more important are opening, closing and skeletonization.

#### **xxiv MULTI-SPECTRAL IMAGES**

A multi-spectral image is a collection of several monochrome images of the same scene, each of them taken with a different sensor. Each image is referred to as a band. A well-known multi-spectral (or multi-band image) is a RGB color image, consisting of a red, a green and a blue image, each of them taken with a sensor sensitive to a different wavelength. In image processing, multi-spectral images are most commonly used for Remote Sensing applications. Satellites usually take several images from frequency bands in the visual and non-visual range. Landsat 5, for example, produces 7 band images with the wavelength of the bands being between 450 and 1250 nm.

All the standard single-band image processing operators can also be applied to multi-spectral images by processing each band separately. For example, a multi-spectral image can be edge detected by finding the edges in each band and then ORing the three edge images together. However, we would obtain more reliable edges, if we associate a pixel with an edge based on its properties in all three bands and not only in one.

To fully exploit the additional information, which is contained in the multiple bands, we should consider the images as one multi-spectral image rather than as a set of monochrome graylevel images. For an image with  $k$  bands, we can then describe the

brightness of each pixel as a point in a k-dimensional space represented by a vector of length k.

Special techniques exist to process multi-spectral images. For example, to classify a pixel as belonging to one particular region, its intensities in the different bands are said to form a feature vector describing its location in the k-dimensional feature space. The simplest way to define a class is to choose a upper and lower threshold for each band, thus producing a k-dimensional 'hyper-cube' in the feature space. Only if the feature vector of a pixel points to a location within this cube, is the pixel classified as belonging to this class. A more sophisticated classification method is described in the corresponding worksheet.

The disadvantage of multi-spectral images is that, since we have to process additional data, the required computation time and memory increase significantly. However, since the speed of the hardware will increase and the costs for memory will decrease in the future, it can be expected that multi-spectral images will become more important in many fields of computer vision.

## xxv NON-LINEAR FILTERING

Suppose that an image processing operator  $F$  acting on two input images  $A$  and  $B$  produces output images  $C$  and  $D$  respectively. If the operator  $F$  is linear, then

$$F(a \times A + b \times B) = a \times C + b \times D \quad [9]$$

where  $a$  and  $b$  are constants. In practice, this means that each pixel in the output of a linear operator is the weighted sum of a set of pixels in the input image.

By contrast, non-linear operators are all the other operators. For example, the threshold operator is non-linear, because individually, corresponding pixels in the two images  $A$  and  $B$  may be below the threshold, whereas the pixel obtained by adding  $A$  and  $B$  may be above threshold. Similarly, an absolute value operation is non-linear:

$$|-1 + 1| \neq |-1| + |1| \quad [10]$$

as is the exponential operator:

$$\exp(1 + 1) \neq \exp(1) + \exp(1) \quad [11]$$

## **xxvi. PIXELS**

In order for any digital computer processing to be carried out on an image, it must first be stored within the computer in a suitable form that can be manipulated by a computer program. The most practical way of doing this is to divide the image up into a collection of discrete (and usually small) cells, which are known as pixels. Most commonly, the image is divided up into a rectangular grid of pixels, so that each pixel is itself a small rectangle. Once this has been done, each pixel is given a pixel value that represents the color of that pixel. It is assumed that the whole pixel is the same color, and so any color variation that did exist within the area of the pixel before the image was discretized is lost. However, if the area of each pixel is very small, then the discrete nature of the image is often not visible to the human eye.

Other pixel shapes and formations can be used, most notably the hexagonal grid, in which each pixel is a small hexagon. This has some advantages in image processing, including the fact that connectivity's less ambiguously defined than with a square grid, but hexagonal grids are not widely used. Part of the reason is that many image capture systems (e.g. most CCD cameras and scanners) intrinsically discretize the captured image into a rectangular grid in the first instance.

## **xxvii. PIXEL CONNECTIVITY**

The notation of pixel connectivity describes a relation between two or more pixels. For two pixels to be connected they have to fulfill certain conditions on the pixel brightness and spatial adjacency.

First, in order for two pixels to be considered connected, their pixel values must both be from the same set of values  $V$ . For a grayscale image,  $V$  might be any range of graylevels, e.g.  $V=\{22,23,\dots,40\}$ , for a binary image we simply have  $V=\{1\}$ .

To formulate the adjacency criterion for connectivity, we first introduce the notation of neighborhood. For a pixel  $p$  with the coordinates  $(x,y)$  the set of pixels given by:

$$N_4(p) = \{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\} \quad [12]$$

is called its 4-neighbors. Its 8-neighbors are defined as

$$N_8(p) = N_4 \cup \{(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)\} \quad [13]$$

From this we can infer the definition for 4- and 8-connectivity:

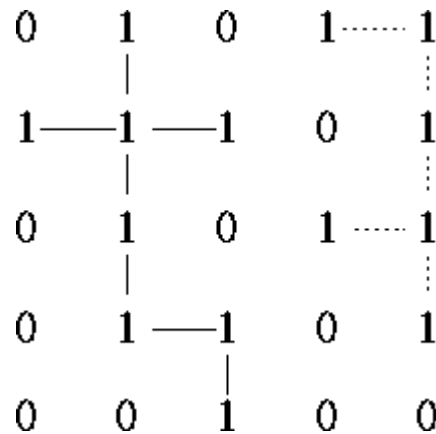
Two pixels  $p$  and  $q$ , both having values from a set  $V$  are 4-connected if  $q$  is from the set  $N_4(p)$  and 8-connected if  $q$  is from  $N_8(p)$ .

General connectivity can either be based on 4- or 8-connectivity; for the following discussion we use 4-connectivity.

A pixel  $p$  is connected to a pixel  $q$  if  $p$  is 4-connected to  $q$  or if  $p$  is 4-connected to a third pixel which itself is connected to  $q$ . Or, in other words, two pixels  $q$  and  $p$  are connected if there is a path from  $p$  and  $q$  on which each pixel is 4-connected to the next one.

A set of pixels in an image which are all connected to each other is called a connected component. Finding all connected components in an image and marking each of them with a distinctive label is called connected component labeling.

An example of a binary image with two connected components which are based on 4-connectivity can be seen in Figure 1. If the connectivity were based on 8-neighbors, the two connected components would merge into one.



*Figure 4.9 Two connected components based on 4-connectivity.*

### xxviii. PIXEL VALUES

Each of the pixels that represents an image stored inside a computer has a pixel value which describes how bright that pixel is, and/or what color it should be. In the simplest case of binary images, the pixel value is a 1-bit number indicating either foreground or background. For a grayscale images, the pixel value is a single number that represents the brightness of the pixel. The most common pixel format is the byte image, where this number is stored as an 8-bit integer giving a range of possible values from 0 to 255. Typically, zero is taken to be black, and 255 is taken to be white. Values in between make up the different shades of gray.

To represent color images, separate red, green and blue components must be specified for each pixel (assuming an RGB colorspace), and so the pixel 'value' is actually a vector of three numbers. Often the three different components are stored as three separates 'grayscale' images known as color planes (one for each of red, green and blue), which have to be recombined when displaying or processing.

Multi-spectral images can contain even more than three components for each pixel, and by extension these are stored in the same kind of way, as a vector pixel value, or as separate color planes.

The actual grayscale or color component intensities for each pixel may not actually be stored explicitly. Often, all that is stored for each pixel is an index into a colormap in which the actual intensity or colors can be looked up.

Although simple 8-bit integers or vectors of 8-bit integers are the most common sorts of pixel values used, some image formats support different types of value, for instance 32-bit signed integers or floating-point values. Such values are extremely useful in image processing as they allow processing to be carried out on the image where the resulting pixel values are not necessarily 8-bit integers. If this approach is used then it is usually necessary to set up a colormap which relates particular ranges of pixel values to particular displayed colors.

## **xxix. PRIMARY COLORS**

It is a useful fact that the huge variety of colors that can be perceived by humans can all be produced simply by adding together appropriate amounts of red, blue and green colors. These colors are known as the primary colors. Thus in most image processing applications, colors are represented by specifying separate intensity values for red, green and blue components. This representation is commonly referred to as RGB.

The primary color phenomenon results from the fact that humans have three different sorts of color receptors in their retinas which are each most sensitive to different visible light wavelengths.

The primary colors used in painting (red, yellow and blue) are different. When paints are mixed, the 'addition' of a new color paint actually subtracts wavelengths from the reflected visible light.

## **xxx. RGB AND COLORSPACES**

A color perceived by the human eye can be defined by a linear combination of the three primary colors red, green and blue. These three colors form the basis for the RGB-colorspace. Hence, each perceivable color can be defined by a vector in the three-dimensional colorspace. The intensity is given by the length of the vector, and the actual color by the two angles describing the orientation of the vector in the colorspace.

The RGB-space can also be transformed into other coordinate systems, which might be more useful for some applications. One common basis for the color space is IHS. In this coordinate system, a color is described by its intensity, hue (average wavelength) and saturation (the amount of white in the color). This color space makes it easier to directly derive the intensity and color of perceived light and is therefore more likely to be used by human beings.

### **xxxi. SPATIAL DOMAIN**

For simplicity, assume that the image  $I$  being considered is formed by projection from scene  $S$  (which might be a two- or three-dimensional scene, etc.).

The spatial domain is the normal image space, in which a change in position in  $I$  directly projects to a change in position in  $S$ . Distances in  $I$  (in pixels) correspond to real distances (e.g. in meters) in  $S$ .

This concept is used most often when discussing the frequency with which image values change, that is, over how many pixels a cycle of periodically repeating intensity variations occur. One would refer to the number of pixels over which a pattern repeats (its periodicity) in the spatial domain.

In most cases, the Fourier Transform will be used to convert images from the spatial domain into the frequency domain.

A related term used in this context is spatial frequency, which refers to the (inverse of the) periodicity with which the image intensity values change. Image features with high spatial frequency (such as edges) are those that change greatly in intensity over short image distances.

Another term used in this context is spatial derivative, which refers to how much the image intensity values change per change in image position.

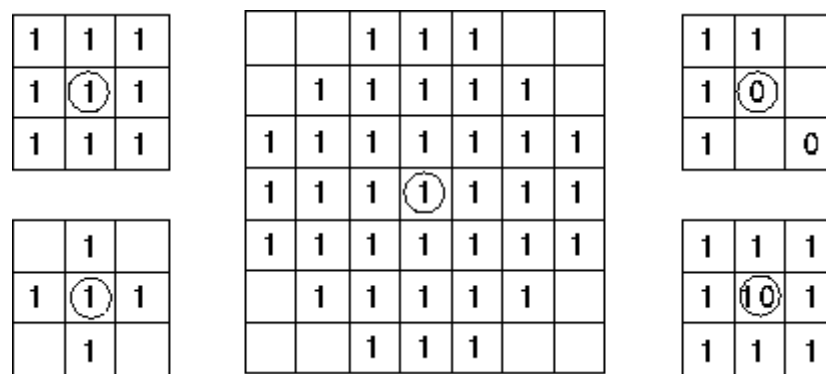
### **xxxii. STRUCTURING ELEMENTS**

The field of mathematical morphology provides a number of important image processing operations, including erosion, dilation, opening and closing. All these

morphological operators take two pieces of data as input. One is the input image, which may be either binary or grayscale for most of the operators. The other is the structuring element. It is this that determines the precise details of the effect of the operator on the image.

The structuring element is sometimes called the kernel, but we reserve that term for the similar objects used in convolutions.

The structuring element consists of a pattern specified as the coordinates of a number of discrete points relative to some origin. Normally cartesian coordinates are used and so a convenient way of representing the element is as a small image on a rectangular grid. Figure 1 shows a number of different structuring elements of various sizes. In each case the origin is marked by a ring around that point. The origin does not have to be in the center of the structuring element, but often it is. As suggested by the figure, structuring elements that fit into a 3×3 grid with its origin at the center are the most commonly seen type.



*Figure 4.10 example structuring elements.*

Note that each point in the structuring element may have a value. In the simplest structuring elements used with binary images for operations such as erosion, the elements only have one value, conveniently represented as a one. More complicated elements, such as those used with thinning or grayscale morphological operations, may have other pixel values.

An important point to note is that although a rectangular grid is used to represent the structuring element, not every point in that grid is part of the structuring element in general. Hence the elements shown in Figure 1 contain some blanks. In many texts, these blanks are represented as zeros, but this can be confusing and so we avoid it here.

When a morphological operation is carried out, the origin of the structuring element is typically translated to each pixel position in the image in turn, and then the points within the translated structuring element are compared with the underlying image pixel values. The details of this comparison, and the effect of the outcome depend on which morphological operator is being used.

### xxxiii. WRAPPING AND SATURATION

If an image is represented in a byte or integer pixel format, the maximum pixel value is limited by the number of bits used for the representation, e.g. the pixel values of a 8-bit image are limited to 255.

However, many image processing operations produce output values which are likely to exceed the given maximum value. In such cases, we have to decide how to handle this pixel overflow.

One possibility is to wrap around the overflowing pixel values. This means that if a value is greater than the possible maximum, we subtract the pixel value range so that the value starts again from the possible minimum value. Figure 1 shows the mapping function for wrapping the output values of some operation into an 8-bit format.

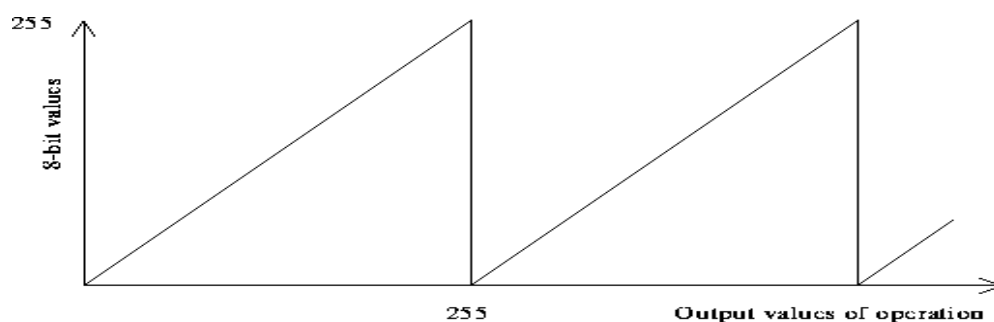
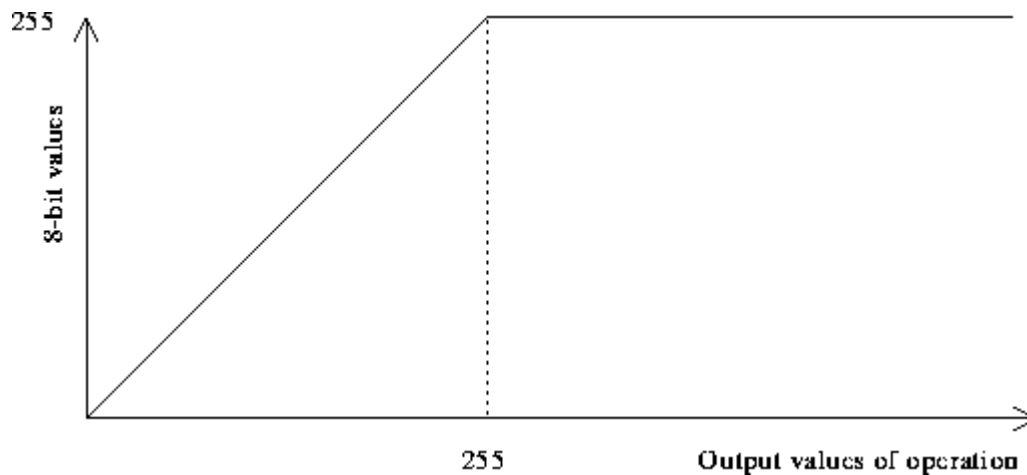


Figure 4.11 Mapping function for wrapping the pixel values of an 8-bit image.

Another possibility is to set all overflowing pixels to the maximum possible values --- an effect known as saturation. The corresponding mapping function for an 8-bit image can be seen in Figure 2.



*Figure 4.12 Mapping function for saturating an 8-bit image.*

If only a few pixels in the image exceed the maximum value it is often better to apply the latter technique, especially if we use the image for display purposes. However, by setting all overflowing pixels to the same value we lose an essential amount of information. In the worst case, when all pixels exceed the maximum value, this would lead to an image of constant pixel values. Wrapping around overflowing pixel retains the differences between values. On the other hand, it might cause the problem that pixel values passing the maximum 'jump' from the maximum to the minimum value. Examples for both techniques can be seen in the worksheets of various point operators.

If possible, it is easiest to change the image format, for example to float format, so that all pixel values can be represented. However, we should keep in mind that this implies an increase in processing time and memory.

## **CHAPTER 5**

### **IMPLEMENTATION DETAILS**

#### **5.1 MODULES**

- a. Dataset
- b. Importing the necessary libraries
- c. Retrieving the images
- d. Splitting the dataset
- e. Building the model
- f. Apply the model and plot the graphs for accuracy and loss
- g. Accuracy on test set
- h. Saving the Trained Model

##### **5.1.1 DATASET:**

In the first module, we developed the system to get the input dataset for the training and testing purpose. We have taken the dataset from Brain Tumor Detection Using Image Processing

Link — <https://www.cancerimagingarchive.net/collections/>

The dataset consists of 1411 brain tumour images

##### **5.1.2 IMPORTING THE NECESSARY LIBRARIES:**

We will be using Python language for this. First we will import the necessary libraries such as keras for building the main model, sklearn for splitting the training and test data, PIL for converting the images into array of numbers and other libraries such as pandas, numpy, matplotlib and tensorflow.

##### **5.1.3 RETRIEVING THE IMAGES:**

We will retrieve the images and their labels. Then resize the images to (224,224) as all images should have same size for recognition. Then convert the images into numpy array.

##### **5.1.4 SPLITTING THE DATASET:**

Split the dataset into train and test. 80% train data and 20% test data.

### **5.1.5 BUILDING THE MODEL:**

For building we will use sequential model from keras library. Then we will add the layers to make convolutional neural network. In the first 2 Conv2D layers we have used 32 filters, and the kernel size is (5,5).

In the MaxPool2D layer we have kept pool size (2,2) which means it will select the maximum value of every 2 x 2 area of the image. By doing this dimension of the image will reduce by factor of 2. In dropout layer we have kept dropout rate = 0.25 that means 25% of neurons are removed randomly.

We apply these 3 layers again with some change in parameters. Then we apply flatten layer to convert 2-D data to 1-D vector. This layer is followed by dense layer, dropout layer and dense layer again. The last dense layer outputs 2 nodes as the brain tumour or not. This layer uses the SoftMax activation function which gives probability value and predicts which of the 2 options has the highest probability.

### **5.1.6 APPLY THE MODEL AND PLOT THE GRAPHS FOR ACCURACY AND LOSS:**

We will compile the model and apply it using fit function. The batch size will be 2. Then we will plot the graphs for accuracy and loss. We got average validation accuracy of 100% and average training accuracy of 98.7%.

### **5.1.7 ACCURACY ON TEST SET:**

We got a accuracy of 100% on test set

### **5.1.8 SAVING THE TRAINED MODEL:**

Once you're confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or .pkl file using a library like pickle.

## **5.2 CONVOLUTIONAL NEURAL NETWORKS**

The objectives behind the first module of the course 4 are:

To understand the convolution operation

To understand the pooling operation

Remembering the vocabulary used in convolutional neural networks (padding, stride, filter, etc.)

### 5.2.1 COMPUTER VISION

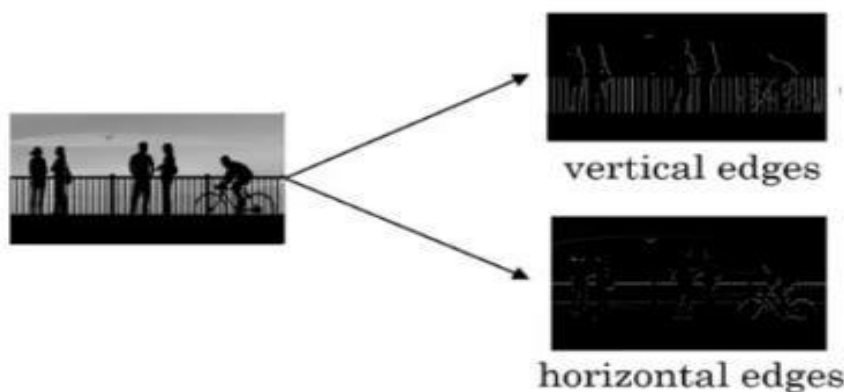
Some of the computer vision problems which we will be solving in this article are:

- a. Image classification
- b. Object detection
- c. Neural style transfer

One major problem with computer vision problems is that the input data can get really big. Suppose an image is of the size  $68 \times 68 \times 3$ . The input feature dimension then becomes 12,288. This will be even bigger if we have larger images (say, of size  $720 \times 720 \times 3$ ). Now, if we pass such a big input to a neural network, the number of parameters will swell up to a HUGE number (depending on the number of hidden layers and hidden units). This will result in more computational and memory requirements – not something most of us can deal with.

#### Edge Detection Example

In the previous article, we saw that the early layers of a neural network detect edges from an image. Deeper layers might be able to detect the cause of the objects and even more deeper layers might detect the cause of complete objects (like a person's face). In this section, we will focus on how the edges can be detected from an image. Suppose we are given the below image: As you can see, there are many vertical and horizontal edges in the image. The first thing to do is to detect these edges:



*Figure 5.1 Vision Edges Conversion*

But how do we detect these edges? To illustrate this, let's take a 6 X 6 grayscale image (i.e. only one channel):

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 3 | 0 | 1 | 2 | 7 | 4 |
| 1 | 5 | 8 | 9 | 3 | 1 |
| 2 | 7 | 2 | 5 | 1 | 3 |
| 0 | 1 | 3 | 1 | 7 | 8 |
| 4 | 2 | 1 | 6 | 2 | 8 |
| 2 | 4 | 5 | 2 | 3 | 9 |

Next, we convolve this 6 X 6 matrix with a 3 X 3 filter:

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 3 | 0 | 1 | 2 | 7 | 4 |
| 1 | 5 | 8 | 9 | 3 | 1 |
| 2 | 7 | 2 | 5 | 1 | 3 |
| 0 | 1 | 3 | 1 | 7 | 8 |
| 4 | 2 | 1 | 6 | 2 | 8 |
| 2 | 4 | 5 | 2 | 3 | 9 |



|   |   |    |
|---|---|----|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

3 X 3 filter

6 X 6 image

*Table 5.1 Conversion of image Matrix*

After the convolution, we will get a 4 X 4 image. The first element of the 4 X 4 matrix will be calculated as:

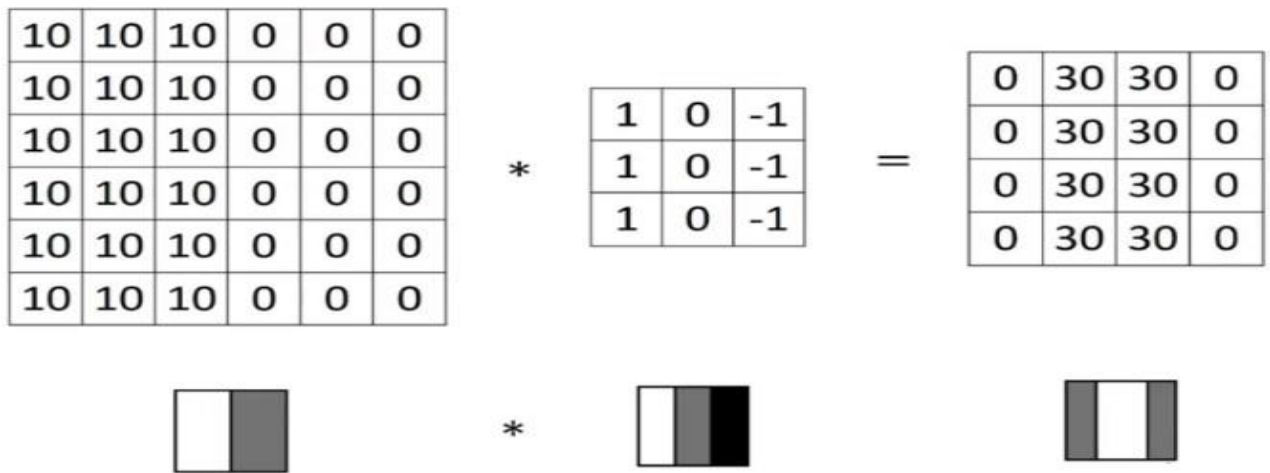
|                |                |                 |
|----------------|----------------|-----------------|
| 3 <sup>1</sup> | 0 <sup>0</sup> | 1 <sup>-1</sup> |
| 1 <sup>1</sup> | 5 <sup>0</sup> | 8 <sup>-1</sup> |
| 2 <sup>1</sup> | 7 <sup>0</sup> | 2 <sup>-1</sup> |

*Table 5.2 Resulted matrix*

Similarly, we will convolve over the entire image and get a 4 X 4 output:

|     |    |    |     |
|-----|----|----|-----|
| -5  | -4 | 0  | 8   |
| -10 | -2 | 2  | 3   |
| 0   | -2 | -4 | -7  |
| -3  | -2 | -3 | -16 |

So, convolving a 6 X 6 input with a 3 X 3 filter gave us an output of 4 X 4. Consider one more example:



*Figure 5.2 Conversion table*

**Note:** Higher pixel values represent the brighter portion of the image and the lower pixel values represent the darker portions. This is how we can detect a vertical edge in an image.

#### More Edge Detection

The type of filter that we choose helps to detect the vertical or horizontal edges. We can use the following filters to detect different edges:

|          |   |    |            |    |    |
|----------|---|----|------------|----|----|
| 1        | 0 | -1 | 1          | 1  | 1  |
| 1        | 0 | -1 | 0          | 0  | 0  |
| 1        | 0 | -1 | -1         | -1 | -1 |
| Vertical |   |    | Horizontal |    |    |

Some of the commonly used filters are:

|              |   |    |               |   |     |
|--------------|---|----|---------------|---|-----|
| 1            | 0 | -1 | 3             | 0 | -3  |
| 2            | 0 | -2 | 10            | 0 | -10 |
| 1            | 0 | -1 | 3             | 0 | -3  |
| Sobel filter |   |    | Scharr filter |   |     |

The Sobel filter puts a little bit more weight on the central pixels. Instead of using these filters, we can create our own as well and treat them as a parameter which the model will learn using backpropagation.

### 5.3 STRIDES CONVOLUTIONS

Suppose we choose a stride of 2. So, while convoluting through the image, we will take two steps – both in the horizontal and vertical directions separately. The dimensions for stride  $s$  will be:

- a. **Input:**  $n \times n$
- b. **Padding:**  $p$
- c. **Stride:**  $s$
- d. **Filter size:**  $f \times f$
- e. **Output:**  $[(n+2p-f)/s+1] \times [(n+2p-f)/s+1]$

Stride helps to reduce the size of the image, a particularly useful feature.

#### Convolutions Over Volume

Suppose, instead of a 2-D image, we have a 3-D input image of shape  $6 \times 6 \times 3$ . We will use a  $3 \times 3 \times 3$  filter instead of a  $3 \times 3$  filter. Let's look at an example:

**Input:**  $6 \times 6 \times 3$

**Filter:**  $3 \times 3 \times 3$

The dimensions above represent the height, width and channels in the input and filter. Keep in mind that the number of channels in the input and filter should be same. This will result in an output of  $4 \times 4$ . Let's understand it visually:

Since there are three channels in the input, the filter will consequently also have three channels. After convolution, the output shape is a  $4 \times 4$  matrix. So, the first element of the output is the sum of the element-wise product of the first 27 values from the input (9 values from each channel) and the 27 values from the filter. After that we convolve over the entire image.

Instead of using just a single filter, we can use multiple filters as well. How do we do that? Let's say the first filter will detect vertical edges and the second filter will detect horizontal edges from the image. If we use multiple filters, the output dimension will change. So, instead of having a  $4 \times 4$  output as in the above example, we would have a  $4 \times 4 \times 2$  output (if we have used 2 filters):

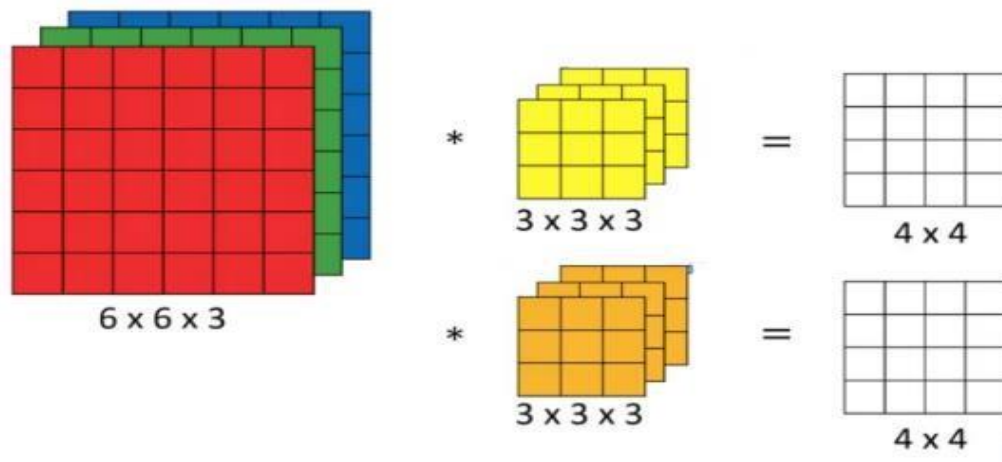


Figure 5.3 Stride Convolutions

Generalized dimensions can be given as:

**Input:**  $n \times n \times n_c$

**Filter:**  $f \times f \times n_c$

**Padding:**  $p$

**Stride:**  $s$

**Output:**  $[(n+2p-f)/s+1] \times [(n+2p-f)/s+1] \times n_c'$

Here,  $n_c$  is the number of channels in the input and filter, while  $n_c'$  is the number of filters.

#### 5.4 ONE LAYER OF A CONVOLUTIONAL NETWORK

Once we get an output after convolving over the entire image using a filter, we add a bias term to those outputs and finally apply an activation function to generate activations. This is one layer of a convolutional network. Recall that the equation for one forward pass is given by:

$$z^{[1]} = w^{[1]} * a^{[0]} + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$

In our case, input  $(6 \times 6 \times 3)$  is  $a^{[0]}$  and filters  $(3 \times 3 \times 3)$  are the weights  $w^{[1]}$ . These activations from layer 1 act as the input for layer 2, and so on. Clearly, the number of parameters in case of convolutional neural networks is independent of the size of the image. It essentially depends on the filter size. Suppose we have 10 filters, each of shape  $3 \times 3 \times 3$ . What will be the number of parameters in that layer? Let's try to solve this:

- Number of parameters for each filter =  $3 \times 3 \times 3 = 27$
- There will be a bias term for each filter, so total parameters per filter = 28
- As there are 10 filters, the total parameters for that layer =  $28 \times 10 = 280$

No matter how big the image is, the parameters only depend on the filter size. Awesome, isn't it? Let's have a look at the summary of notations for a convolution layer:

$f^{[l]}$  = filter size

$p^{[l]}$  = padding

$s^{[l]}$  = stride

$n_{[c]}^{[l]}$  = number of filters

In a convolutional network (ConvNet), there are basically three types of layers:

- Convolution layer
- Pooling layer
- Fully connected layer

Let's understand the pooling layer in the next section.

### 5.4.1 POOLING LAYERS

Pooling layers are generally used to reduce the size of the inputs and hence speed up the computation. Consider a 4 X 4 matrix as shown below:

|   |   |   |   |
|---|---|---|---|
| 1 | 3 | 2 | 1 |
| 2 | 9 | 1 | 1 |
| 1 | 3 | 2 | 3 |
| 5 | 6 | 1 | 2 |

Applying max pooling on this matrix will result in a 2 X 2 output:

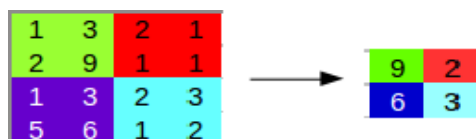


Figure 5.4 Pooling

For every consecutive 2 X 2 block, we take the max number. Here, we have applied a filter of size 2 and a stride of 2. These are the hyperparameters for the pooling layer. Apart from max pooling, we can also apply average pooling where, instead of taking

the max of the numbers, we take their average. In summary, the hyperparameters for a pooling layer are:

1. Filter size
2. Stride
3. Max or average pooling

If the input of the pooling layer is  $n_h \times n_w \times n_c$ , then the output will be  $\{(n_h - f) / s + 1\} \times \{(n_w - f) / s + 1\} \times n_c$ .

We'll take things up a notch now. Let's look at how a convolution neural network with convolutional and pooling layer works. Suppose we have an input of shape  $32 \times 32 \times 3$ : There are a combination of convolution and pooling layers at the beginning, a few fully connected layers at the end and finally a softmax classifier to classify the input into various categories. There are a lot of hyperparameters in this network which we have to specify as well.

## CHAPTER 6

### RESULTS AND DISCUSSION

An integrated framework for brain tumor detection and classification using Convolutional Neural Networks (CNN) and Support Vector Machines (SVM) combines the power of deep learning with the versatility of traditional machine learning algorithms. In this approach, the process typically begins with data preprocessing, where brain tumor images are prepared for analysis through resizing, normalization, and augmentation. Following this, CNNs are employed to automatically extract intricate features from the images, leveraging architectures. These extracted features are then transformed into a suitable representation for SVM classification, often involving techniques such as dimensionality reduction via Principal Component Analysis (PCA) or simply flattening the feature vectors. The SVM classifier is trained on this representation, learning to classify the brain tumor images into different categories based on the provided training examples. Evaluation and validation are crucial steps, ensuring the model's performance is robust and accurate, often utilizing separate test datasets and cross-validation techniques. Fine-tuning and optimization further refine the framework's effectiveness, enhancing parameters and employing ensemble methods where applicable. Ultimately, the integrated framework can be deployed in real-world applications, aiding clinicians in diagnosing brain tumors with improved efficiency and accuracy, thereby potentially enhancing patient outcomes.

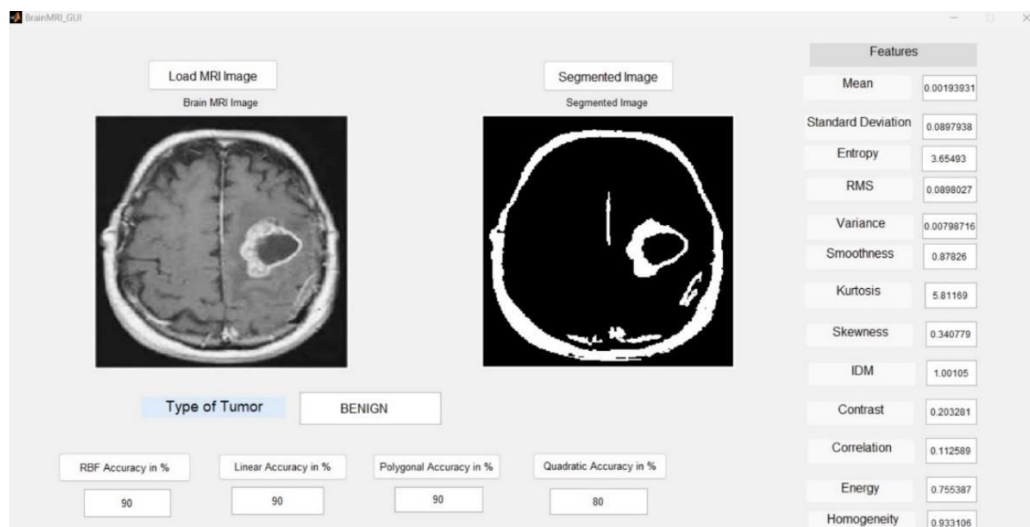


Figure 6.1 segmented Image Conversion output

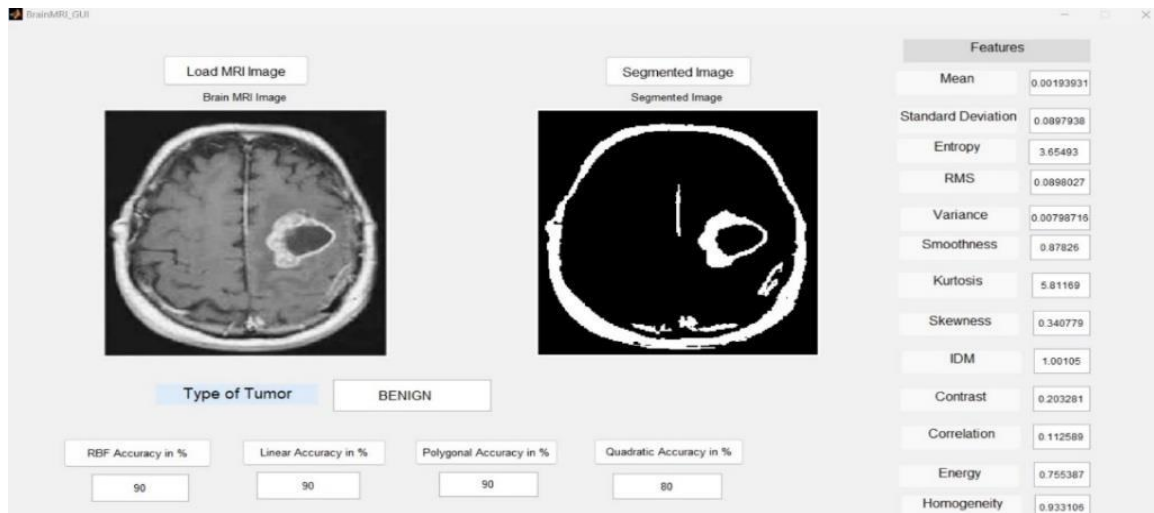


Figure 6.2 Segmented Image Conversion output- **Benign Tumor detection**

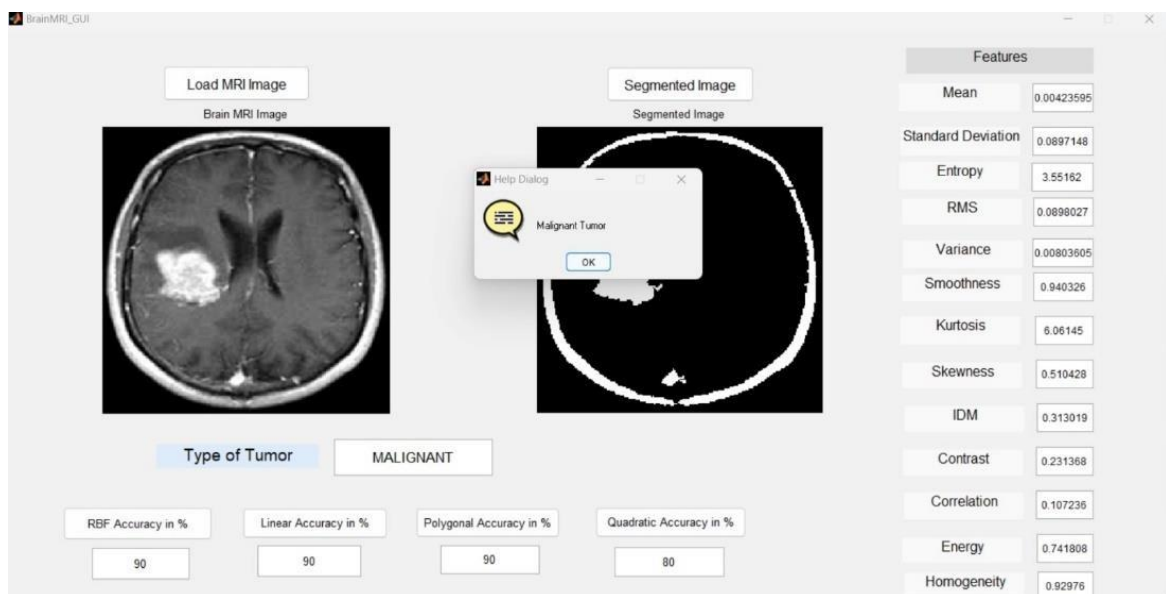


Figure 6.3 Malignant Tumor Detection

## **CHAPTER 7**

### **CONCLUSION**

A potential new strategy for enhancing brain tumour detection and therapy is the initiative on brain tumour categorization and prediction. Researchers can help patients with brain tumours live better lives by creating more accurate and reliable models, making models more available to physicians, and creating models that can forecast the growth of brain tumors and personalize therapy. Here are some of the key findings of the project:

Convolutional neural networks (CNNs) may effectively diagnose brain cancers using MRI data. CNNs are a form of deep learning model that can extract fine features from photographs. It has been proved that CNNs can classify brain cancers with good accuracy using benchmark datasets.

Data augmentation may help CNNs perform better when classifying brain cancers. By creating new photographs from existing ones, a procedure called as data augmentation may be applied to extend a dataset. By making CNNs more resistant to noise and data volatility, data augmentation may help them perform better.

Transfer learning may be applied to increase CNNs' categorization capacities for brain cancers. With the use of the transfer learning technique, information gathered from a huge dataset may be transferred to a smaller dataset. By decreasing the amount of data required to train the model, transfer learning may help CNNs perform better when diagnosing brain tumors.

Although the initiative is still in its early phases, it has the potential to have a substantial effect on the identification and treatment of brain tumors. It is feasible to design models that can efficiently identify and diagnose brain cancers, anticipate the progression of tumors, and adapt therapy for particular people with continued research and improvement. For patients with brain tumors, this would result in earlier discovery, more effective treatment, and improved prognosis.

## REFERENCES

- [1] Rahib L, Smith BD, Aizenberg R, Rosenzweig AB, Fleshman JM, Matrisian LM. Projecting cancer incidence and deaths to 2030: the unexpected burden of thyroid, liver, and pancreas cancers in the United States. *Cancer Res* 2014; 74: 2913–21.
- [2] Siegel RL, Miller KD, Jemal A. Cancer statistics, 2019. *CA Cancer J Clin* 2019; 69: 7–34.
- [3] Ryan DP, Hong TS, Bardeesy N. Brain adenocarcinoma. *N Engl J Med* 2014; 371: 1039–49.
- [4] Al-Hawary MM, Francis IR, Chari ST, et al. Brain ductal adenocarcinoma radiology reporting template: consensus statement of the Society of Abdominal Radiology and the American Brain Association. *Radiology* 2014; 270: 248–60.
- [5] Dewitt J, Devereaux BM, Lehman GA, Sherman S, Imperiale TF. Comparison of endoscopic ultrasound and computed tomography for the preoperative evaluation of brain cancer: a systematic review. *Clin Gastroenterol Hepatol* 2006; 4: 717–25; quiz 664.
- [6] Yasaka K, Abe O. Deep learning and artificial intelligence in radiology: current applications and future directions. *PLoS Med* 2018; 15: e1002707.
- [7] Esteva A, Kuprel B, Novoa RA, et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* 2017; 542: 115–18.
- [8] B. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, et al., "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)," *IEEE Transactions on Medical Imaging, Institute of Electrical and Electronics Engineers (IEEE)*, 2018, pp.33.
- [9] W. Huang, Y. Yang, et al., "Random Feature Subspace EnsembleBased Extreme Learning Machine for Liver Tumor Detection and Segmentation". *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2019, Aug 29.
- [10] M. Soltaninejad, et al., "Brain Tumour Grading in Different MRI Protocols using SVM on Statistical Features," *Proceedings of the MIUA*, 2022, Jul.

## APPENDIX

### A.SAMPLE CODE

```

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

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function BrainMRI_GUI_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;
ss = ones(200,200);
axes(handles.axes1);
imshow(ss);
axes(handles.axes2);
imshow(ss);
% Update handles structure
guidata(hObject, handles);

function varargout = BrainMRI_GUI_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
[FileName,PathName] = uigetfile('*.jpg;*.png;*.bmp','Pick an MRI Image');
if isequal(FileName,0) || isequal(PathName,0)
    warndlg('User Press Cancel');
else
    P = imread([PathName,FileName]);
    P = imresize(P,[200,200]);
    % input =imresize(a,[512 512]);

    axes(handles.axes1)
    imshow(P);title('Brain MRI Image');

    handles.ImgData = P;
    % handles.FileName = FileName;

    guidata(hObject,handles);
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)

```

```

if isfield(handles, 'ImgData')
    %if isfield(handles, 'imgData')
    I = handles.ImgData;

gray = rgb2gray(I);
% Otsu Binarization for segmentation
level = graythresh(I);
%gray = gray>80;
img = im2bw(I, .6);
img = bwareaopen(img, 80);
img2 = im2bw(I);

axes(handles.axes2)
imshow(img); title('Segmented Image');
%imshow(tumor); title('Segmented Image');

handles.ImgData2 = img2;
guidata(hObject, handles);

signal1 = img2(:, :);
%Feat = getmswpfeat(signal, winsize, wininc, J, 'matlab');
%Features = getmswpfeat(signal, winsize, wininc, J, 'matlab');

[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];
G = pca(DWT_feat);
whos DWT_feat
whos G
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));
%Skewness = skewness(img)
Variance = mean2(var(double(G)));
a = sum(double(G(:)));
Smoothness = 1-(1/(1+a));
Kurtosis = kurtosis(double(G(:)));
Skewness = skewness(double(G(:)));
% Inverse Difference Movement
m = size(G, 1);
n = size(G, 2);
in_diff = 0;
for i = 1:m
    for j = 1:n
        temp = G(i, j)./(1+(i-j).^2);
        in_diff = in_diff+temp;
    end
end
IDM = double(in_diff);

feat = [Contrast, Correlation, Energy, Homogeneity, Mean, Standard_Deviation,
Entropy, RMS, Variance, Smoothness, Kurtosis, Skewness, IDM];

```

```

load Trainset.mat
xdata = meas;
group = label;
svmStruct1 = svmtrain(xdata,group,'kernel_function','linear');
species = svmclassify(svmStruct1,feat,'showplot',false);

if strcmpi(species,'MALIGNANT')
    helpdlg(' Malignant Tumor ');
    disp(' Malignant Tumor ');
else
    helpdlg(' Benign Tumor ');
    disp(' Benign Tumor ');
end
set(handles.edit4,'string',species);
% Put the features in GUI
set(handles.edit5,'string',Mean);
set(handles.edit6,'string',Standard_Deviation);
set(handles.edit7,'string',Entropy);
set(handles.edit8,'string',RMS);
set(handles.edit9,'string',Variance);
set(handles.edit10,'string',Smoothness);
set(handles.edit11,'string',Kurtosis);
set(handles.edit12,'string',Skewness);
set(handles.edit13,'string',IDM);
set(handles.edit14,'string',Contrast);
set(handles.edit15,'string',Correlation);
set(handles.edit16,'string',Energy);
set(handles.edit17,'string',Homogeneity);
end

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
load Trainset.mat
%data = [meas(:,1), meas(:,2)];
Accuracy_Percent= zeros(200,1);
itr = 80;
hWaitBar = waitbar(0,'Evaluating Maximum Accuracy with 100 iterations');
for i = 1:itr
    data = meas;
    %groups = ismember(label,'BENIGN ');
    groups = ismember(label,'MALIGNANT');
    [train,test] = crossvalind('HoldOut',groups);
    cp = classperf(groups);
    %svmStruct =
    svmtrain(data(train,:),groups(train),'boxconstraint',Inf,'showplot',false,'kernel_
    _function','rbf');
    svmStruct_RBF =
    svmtrain(data(train,:),groups(train),'boxconstraint',Inf,'showplot',false,'kernel_
    _function','rbf');
    classes2 = svmclassify(svmStruct_RBF,data(test,),'showplot',false);
    classperf(cp,classes2,test);
    %Accuracy_Classification_RBF = cp.CorrectRate.*100;
    Accuracy_Percent(i) = cp.CorrectRate.*100;
    sprintf('Accuracy of RBF Kernel is: %g%%',Accuracy_Percent(i))
    waitbar(i/itr);
end
delete(hWaitBar);
Max_Accuracy = max(Accuracy_Percent);
sprintf('Accuracy of RBF kernel is: %g%%',Max_Accuracy)
set(handles.edit1,'string',Max_Accuracy);
guidata(hObject,handles);

```

```

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
load Trainset.mat
%data = [meas(:,1), meas(:,2)];
Accuracy_Percent= zeros(200,1);
itr = 100;
hWaitBar = waitbar(0,'Evaluating Maximum Accuracy with 100 iterations');
for i = 1:itr
data = meas;
%groups = ismember(label,'BENIGN ');
groups = ismember(label,'MALIGNANT');
[train,test] = crossvalind('HoldOut',groups);
cp = classperf(groups);
svmStruct =
svmtrain(data(train,:),groups(train),'showplot',false,'kernel_function','linear')
;
classes = svmclassify(svmStruct,data(test,:), 'showplot',false);
classperf(cp,classes,test);
%Accuracy_Classification = cp.CorrectRate.*100;
Accuracy_Percent(i) = cp.CorrectRate.*100;
sprintf('Accuracy of Linear Kernel is: %g%%',Accuracy_Percent(i))
waitbar(i/itr);
end
delete(hWaitBar);
Max_Accuracy = max(Accuracy_Percent);
sprintf('Accuracy of Linear kernel is: %g%%',Max_Accuracy)

set(handles.edit2,'string',Max_Accuracy);

load Trainset.mat
%data = [meas(:,1), meas(:,2)];
Accuracy_Percent= zeros(200,1);
itr = 100;
hWaitBar = waitbar(0,'Evaluating Maximum Accuracy with 100 iterations');
for i = 1:itr
data = meas;
groups = ismember(label,'BENIGN ');
groups = ismember(label,'MALIGNANT');
[train,test] = crossvalind('HoldOut',groups);
cp = classperf(groups);
svmStruct_Poly =
svmtrain(data(train,:),groups(train),'Polyorder',2,'Kernel_Function','polynomial'
);
classes3 = svmclassify(svmStruct_Poly,data(test,:), 'showplot',false);
classperf(cp,classes3,test);
Accuracy_Percent(i) = cp.CorrectRate.*100;
sprintf('Accuracy of Polynomial Kernel is: %g%%',Accuracy_Percent(i))
waitbar(i/itr);
end
delete(hWaitBar);
Max_Accuracy = max(Accuracy_Percent);
%Accuracy_Classification_Poly = cp.CorrectRate.*100;
sprintf('Accuracy of Polynomial kernel is: %g%%',Max_Accuracy)
set(handles.edit3,'string',Max_Accuracy);

function edit1_Callback(hObject, eventdata, handles)
% hObject handle to edit1 (see GCBO)
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

```

```

function edit2_Callback(hObject, eventdata, handles)
function edit2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function pushbutton4_CreateFcn(hObject, eventdata, handles)

```

## B.SAMPLE CODE SNIPPET

```

BrainMR_GUI.m
1 function varargout = BrainMRI_GUI(varargin)
2 gui_Singleton = 1;
3 gui_State = struct('gui_Name',       mfilename, ...
4                   'gui_Singleton',   gui_Singleton, ...
5                   'gui_OpeningFcn', @BrainMRI_GUI_OpeningFcn, ...
6                   'gui_OutputFcn',  @BrainMRI_GUI_OutputFcn, ...
7                   'gui_LayoutFcn',  [] , ...
8                   'gui_Callback',    []);
9 if nargin <= ischar(varargin{1})
10     gui_State.gui_Callback = str2func(varargin{1});
11 end
12
13 if nargin
14     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
15 else
16     gui_mainfcn(gui_State, varargin{:});
17 end
18
19 % ----- Executes opening function when double clicked on icon
20 function BrainMRI_GUI_OpeningFcn(hObject, eventdata, handles, varargin)
21 handles.output = hObject;
22 ax = axes(200,200);
23 axes(handles.ax);
24 imshow(ax);
25 axes(handles.axes2);
26 imshow(ax);
27 % Update handles structure
28 guidata(hObject, handles);
29
30 function varargout = BrainMRI_GUI_OutputFcn(hObject, eventdata, handles)
31
32
33
34 varargout{1} = handles.output;
35
36
37 % --- Executes on button press in pushbutton1.
38 function pushbutton1_Callback(hObject, eventdata, handles)
39
40

```

### Sample Code Snippet -1 Importing and Prediction of Tumor

```

BrainMR_GUI.m
253
254 % Hint: get(hObject,'String') returns contents of edit as text
255 % str2double(get(hObject,'String')) returns contents of edit as a double
256
257
258 % --- Executes during object creation, after setting all properties.
259 function edit1_CreateFcn(hObject, eventdata, handles)
260 % hObject handle to edit1 (see GCBO)
261 % eventdata reserved - to be defined in a future version of MATLAB
262 % handles empty - handles not created until after all CreateFcns called
263
264 % Hint: edit controls usually have a white background on Windows.
265 % See the URL "http://www.mathworks.com/help/matlab/creating_gui.html#CreateEditControls"
266 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
267     set(hObject,'BackgroundColor','white');
268 end
269
270
271 function edit2_Callback(hObject, eventdata, handles)
272 % hObject handle to edit2 (see GCBO)
273 % eventdata reserved - to be defined in a future version of MATLAB
274 % handles structure with handles and user data (see GUIDATA)
275
276 % Hint: get(hObject,'String') returns contents of edit2 as text
277 % str2double(get(hObject,'String')) returns contents of edit2 as a double
278
279
280 % --- Executes during object creation, after setting all properties.
281 function edit2_CreateFcn(hObject, eventdata, handles)
282 % hObject handle to edit2 (see GCBO)
283 % eventdata reserved - to be defined in a future version of MATLAB
284 % handles empty - handles not created until after all CreateFcns called
285
286 % Hint: edit controls usually have a white background on Windows.
287 % See the URL "http://www.mathworks.com/help/matlab/creating_gui.html#CreateEditControls"
288 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
289     set(hObject,'BackgroundColor','white');
290 end
291
292

```

### Sample Code Snippet-2 Using of SVM algorithm

```

BrainMR_GUI.m
40
41
42
43 [FileName,PathName] = uigetfile('*.jpg;*.png;*.bmp','Pick an MRI Image');
44 if isequal(FileName,0) || isequal(PathName,0)
45     warndlg('User Press Cancel!');
46 else
47     F = imread([PathName,FileName]);
48     F = imresize(F,[200,200]);
49     % input =imresize(s,[512 512]);
50
51     axes(handles.axes1)
52     imshow(F);title('Brain MRI Image');
53
54     handles.imgData = F;
55     % handles.FileName = FileName;
56     guidata(hObject,handles);
57 end
58
59
60 % --- Executes on button press in pushbutton2.
61 function pushbutton2_Callback(hObject, eventdata, handles)
62
63
64
65 if isfield(handles,'imgData')
66     %if isfield(handles,'imgData')
67     I = handles.imgData;
68
69     gray = rgb2gray(I);
70     % Otsu Binarization for segmentation
71     level = graythresh(I);
72     %gray = gray>0;
73     img = im2bw(I,level);
74     img = bwareaopen(img,80);
75     img2 = im2bw(I);
76
77     axes(handles.axes2)
78     imshow(img);title('Segmented Image');
79     imshow(img2);title('Segmented Image');
80

```

### Sample Code Snippet Using of CNN algorithm

# BRAIN TUMOUR CLASSIFICATION AND PREDICTION USING CNN AND SVM

Venkata Krishan T R, Student, [tsrvenkat07@gmail.com](mailto:tsrvenkat07@gmail.com)

Karthikeyan m, Student, [Karthick6102002@gmail.com](mailto:Karthick6102002@gmail.com)

Dr. Sankari M, Assistant Professor, [sankari.cse@sathyabama.ac.in](mailto:sankari.cse@sathyabama.ac.in)

Dr. Kamatchi K.S., Assistant Professor, School of computing, SIST

Department of computer science and Engineering

Sathyabama Institute of Science and Technology – Chennai

**ABSTRACT:** *Brain tumors are one of the main causes of most tumors deaths international. Early detection of brain tumor is tough due to the fact most tumors-specific symptoms arise best in superior degrees and there may be no dependable screening device to pick out high-chance sufferers. But if brain most tumors is detected early, it can be cured. This project tries to stumble on brain tumors from CT pics. It uses a number picture processing techniques and architectural fashions to stumble on tumors. After image pre-processing, the architectural version is used to locate the tumor area in the photo. The tumor detection accuracy throughout education is approximately 98.7%.*

**KEYWORDS:** Brain Tumor, SVM, Deep Learning, Tumor, CNN.

## I.INTRODUCTION

Brain most tumors is the 11th maximum commonplace most tumors in guys and the ninth most common tumor in girls, however it is the fourth main cause of tumor death among women and men in the United States. Brain most tumors is the only tumor with a 5-yr survival rate within the single digits. According to a current file published by means of Brain Tumor, mind most tumors is predicted to become the second main purpose of most tumors deaths inside the United States in 2020. The fundamental purpose why brain tumours are recognized past due is because there aren't any powerful strategies for early detection. Additionally, the signs of maximum brain tumours are diffused and associated with many other belly situations. Additionally, if tumour spreads to different organs, treatment will become more difficult. Therefore, there may be a continuing want for a technique to help radiologists stumble on brain tumours at an early stage. Not plenty paintings has been carried out to identify brain tumours. From a review of the literature, brain tumours are identified based totally

on disease signs and the affected person's clinical history, but without using imaging strategies.

## II.OBJECTIVE

Identify brain tumors by way of feature extraction the usage of convolutional neural networks (CNN). Deep learning to awaken a patient or pick out a tumor at an early stage.

## III.RELATED WORK

**1. A Survey of Automatic MRI based Brain Tumor Segmentation Techniques/ M Subashree; J Sangeetha /2018/**

In the clinical discipline, photograph processing performs an important function in tumor diagnosis. Medical imaging is a developing and complex area. Many human beings be afflicted by brain tumors; this ailment could be very critical and dangerous. A brain tumor is a random boom of cells that proliferate uncontrollably. Early evaluation of brain tumors plays an important function in therapeutic capacity to enhance results and growth patient survival.

**2. Brain Tumor Diagnose Applying CNN through MRI/ Hamming Hu; Xin Li; Wenyi Yao; Zhuocheng Yao/2021**

Our research objectives to use advanced generation for brain tumour detection and prognosis from brain MRI image facts. In this observe, various deep gaining knowledge of fashions which include VGG16/19, Alex Net, Google Net and Resnet were tried to illustrate their effectiveness and accuracy. Furthermore, the YOLO model became carried out to choose the best area of the tumour in the MR picture and carry out photo improvements to enhance the type accuracy.

### **3. Comparison of deep learning models for brain tumour classification using MRI images/Necip Çınar; Buket Kaya; Mehmet Kaya/2022/**

Based at the examine, the ResNet101 model carried out the highest accuracy with an accuracy fee of ninety eight.6%. In addition, the VGG19 model confirmed the very best accuracy of ninety seven.2%. Other models have accuracy values of InceptionV3 94.Three%, DenseNet169 92.8% and Alex Net 89.5% respectively. This low success fee suggests that the structures utilized in these fashions aren't suitable for MR imaging studies in comparison to other structures. For this motive, it was concluded that the usage of the ResNet architecture to detect tumours from mind MR snap shots is more powerful than other fashions.

### **4. Identification Using MRI Images/T. M. Shahriar Sazzad; K. M. Tanzibul Ahmmed/2019**

This observe proposes an automated approach that consists of the front-end enhancement for varying shades of grey. A clear out feature is used to do away with as a whole lot undesirable noise as possible and improve segmentation. That is why this take a look at examines pictures with emotion; for shade segmentation, threshold-based OTSU segmentation became used. Finally, expert pathologists furnished characteristic records used to define the user's region (brain tumour region). Experimental consequences show that the proposed technique yields higher outcomes than the available methods in terms of accuracy, whilst keeping an acceptable level of accuracy for expert pathologists.

### **5. Implementation of Latest Deep Learning Techniques for Brain Tumour Identification from MRI Images/P. Venkateswarlu Reddy; D. Ganesh; L. Sumaiya Azeem/2023**

*This take a look at targets to use Convolutional Neural Networks (CNs), ResNet50 and VGG16 to are expecting whether or not a person has a brain tumour based on a brain map consisting of brain MRI images and evaluate their overall performance. These techniques make it simpler for professionals to hit upon tumours. The accuracy consequences are brilliant and provide tremendous perception into the human promise of deep studying structures on specific datasets. Brain MRI pix of 3000 sufferers were tested, of which 1500 MRI pictures have been tumour and 1500 were non-tumour. It is obvious from this that deep learning models can make contributions a lot to the analysis of brain tumours, improving the accuracy and speed of analysis. As an end result, the usage of these models to diagnose different complicated illnesses will growth inside the destiny.*

### **6. Convolutional Neural Networks for Brain Tumour Segmentation Using Different Sets of**

### **MRI Sequences/Masoomah Rahimpour; Karolien Goffin; Michel Koole/2019**

In this look at, we present and evaluate the first convolutional neural network (CNN) based on Deep Medic framework for brain tumour segmentation using numerous combos of multi-sequential MRI datasets. In the first fusion method we executed a custom grid for all viable combos of MRI sequences, the behind schedule fusion method is extra widespread, where we set an independent grid for every kind of MRI collection and eventually generate fully fused characteristic maps. Remedy. Compared to the first fusion series, the overall performance of the fusion therapy approach changed into very comparable, although it offers greater flexibility in terms of fusion of all to be had MRI records.

### **7. Brain Tumour Segmentation in MRI images/Muhammad Zawish; Asad Ali Siyal/2018**

Accurate information of these areas plays an vital position in the diagnosis and treatment of malignant tumours. In this study, we provided an advanced photograph segmentation method based totally on overall comparison methods to appropriately section tumours from brain MRI dataset. The artwork is based totally at the energetic form of san-ves without boundary lines. In this system, one pressure compresses the form and another force expands the intensity. These two forces are balanced whilst the shape reaches the cease of the item of interest, as a result locating the apex of the item and resulting in correct segmentation.

### **8. Optimizing Convolutional Neural Networks for Brain Tumour Segmentation in MRI Images/Mohamed Babikir Ali; Ruba Ali Hamad; Mohanned Ahmed/2018**

Dozens of neural networks were proposed for tumour segmentation, however they have got not but reached enough accuracy to be used in real packages. In this text, we've got cantered on enhancing the CNN patch classifier, and the results acquired are discussed to expose the impact of some layout selections. We evaluated segmentation effects the use of Dice's similarity coefficient (DSC). The outcomes of this paper can be used as a guide to enhance present models or to increase new CNN fashions. Finally, we indicate possible destiny research guidelines.

## **IV.EXISTING SYSTEM**

Many strategies were proposed to cope with label noise type for herbal pics. Ren et al., advise a method of featuring weights for training samples using a further natural validation set. His approach is to apply smaller weights for click on patterns and boom

the weight of mundane education strategies to improve tilt healthy. The development and application of strategies for studying click on labels in scientific image records is tremendously limited. Degani et al. – Sampling of label noise as part of a deep studying community to achieve actual label noise-free fashions for the project of reading breast micro calcifications in Multiview mammograms.

### **Disadvantages of Existing System**

Not plenty paintings has been performed to discover brain tumours. From an overview of the literature, brain tumours are identified primarily based on ailment signs and the affected person's scientific history, however without the usage of imaging processes. Here the disorder cannot be expected earlier and it turns into more hard because the condition progresses. Diagnosis is very gradual. He does now not use any imaging techniques to diagnose the ailment in its early stages.

### **V.PROPOSED SYSYTEM**

This application tries to stumble on brain tumours from CT test pictures. These pictures are pre-processed using trendy picture processing techniques, after which a texture model framework is used to perceive the tumour region in the photo. The dataset turned into taken from the Tumour Image Repository. These CT experiment photographs of the brain tumour are fed into the computer. Since these photos are in different codecs, we need to convert them to jpg and the picture length is very massive, so we have used only some of them. Classification is carried out the usage of the sequential version framework. In this example, the shape is constructed according to the sections defined in the diagram. When the laptop learns category primarily based on the functions supplied to it, it could suit the check statistics into this type of categories. We have set up a chain using assessment-stronger CT photos of patients to distinguish mind most tumours from wholesome pancreas. RONCAS validated high accuracy and fantastic sensitivity compared to radiologists' independent assessments of visual interpretation, with perfect performance in check units of a selection of patients. These effects provide the first proof of concept that CT can seize elusive capabilities of brain tumour to aid and supplement radiologists within the prognosis and analysis of mind most tumours.

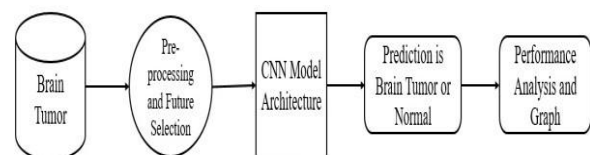
### **Advantages of Proposed System**

Deep studying the use of Convolutional Neural Networks (CN) has shown awesome ability in clinical picture evaluation. Building a neural network on a hard and fast of neurons with activation features and parameters to extract and combine strains of images and create a model that reflects the complex relationship among photographs and

diagnosis. Ronkus stated more correct diagnoses were received on imaging. RONGS can appropriately distinguish brain most tumours from non-tumorous pancreas, and accounting for variations in affected person gender and ethnicity and imaging parameters, is essential in actual-world clinical practice. Roncus guarantees to broaden laptop-based totally tools for brain most tumours detection and diagnosis that complement radiologists' interpretation.

### **VI.BLOCKDIAGRAM**

It is a simple graphical system used to symbolize a gadget in phrases of facts input into a computer, various methods performed on that data, and data generated through the machine. Modelling gadget additives. These additives are the method of the gadget, the facts utilized by the manner, the outside entity related to the gadget, and the facts flowing into the gadget. Shows how statistics moves in a gadget and how it undergoes a chain of adjustments. It is a graphical method of depicting facts flow and the changes used even as taking facts from enter to output.



### **VII.MODULES:**

1. Set the Date
2. Import Required Libraries
3. Photo Acquisition
4. Information Set Partitioning
5. Pattern Making
6. Use Accuracy And Damage Of Version And Machine.
7. Be Careful With the Take A Look at Set
8. Save Organization Model

#### **1. Set the Date**

In the first module we created a gadget some data to examine Try extra functions. We took Using a dataset from a examine on brain tumour diagnosis photograph processing. <https://www.tumourimagingarchive.net/collections/> . The dataset consists of 1411 brain tumour images

#### **2. Import Required Libraries**

For this we are able to use Python language. Main libraries should be found They have been eager to construct a start-up Learn to model, layout and examine Photo PIL to exchange figures, numbers;

And diverse libraries with pandas; NumPy, matplotlib and tensor flush.

### 3. Photo Acquisition

Let's ship their images and labels. Then resize the pics (224224). This need to be equal to the length of the picture. Then rearrange the pics primarily based on the order. The statistics set is split; View column partition statistics 5.80% transition education and 20% examination Information

### 4. Information Set Partitioning

Let's use this case to make it From the Keras library. Let's add layers Building a lively neural community. In' we used the primary two Conv2D filter layers 32 and Centre length (five.5). We saved MaxPool2D inside the pool tank length (2.2), which is a lot Price is ready for every 2x2 region the rate placed us to sleep Retained dropout component = zero.25, which It is 25% of neurons Destroyed at will. And use these 3 commands reporting inside limits. So we View 2D Warp Transform Layer Information is virtually a one-dimensional vector. Its miles The layer should be very thick Layer, leaf layer and repeat A thick layer closure indicates a dense layer The two ends of the mind are swollen or absent. Its far layer uses SoftMax implementation A function that provides capacity at a fee and

He guesses it is certainly one of alternatives fat chance Use patterns and designs correctly loss.

### 5. Pattern Making

We will assemble a version and comply with up using version functions. Criteria we can set the mass equal to 2 Accuracy and loss of plot. We were with average retrieval accuracy 100% and common education accuracy ninety eight. Seven%

### 7. Be Careful With the Take A Look at Set

We are given 100% accuracy check in body

### 8. Save Organization Model

You simply should be confident Version taught and tested Production Environment, First Steps Save using .H5 or .Pkl file Library like Muria.

## VIII.EXISTING ALGORITHM

### Support Vector Machine (SVM)

A help vector device (SVM) is an effective mastering set of rules used for linear or nonlinear classification, regression, and outlier detection. SVM can be used for various duties which includes textual content category, image type, hand detection, handwriting reputation, gene expression analysis, face detection, and anomaly detection. SVMs are adaptable and effective for various applications

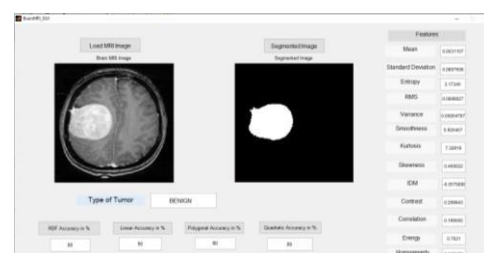
because they are able to manage high dimensional records and non-linear relationships. The SVM set of rules could be very useful due to the fact we're attempting to find the choicest hyper plane among exclusive goal line types.

## IX.PROPOSED ALGORITHM

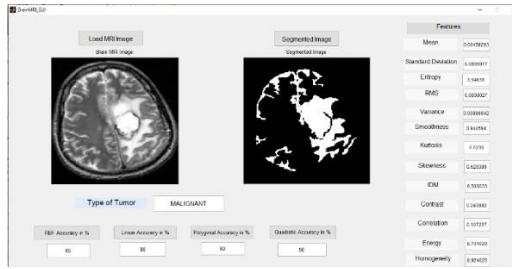
In deep studying, Convolutional Neural Network (CNN/ConvNet) is a normally used deep neural community for visible photograph evaluation. Now whilst we consider a neutral community, we consider matrix multiplication, however that isn't always the case with ConvNets. He makes use of a unique approach known as convolution. In arithmetic, convolution is a mathematical operation that includes two operations that bring about a third operation that expresses how the shape of 1 is converted by way of the other. Convolutional neural networks encompass numerous layers of artificial neurons. Artificial neurons, a replica in their organic opposite numbers, are mathematical features that calculate the sum of a couple of inputs and output an energetic cost. When you positioned a photo right into a ConvNet, each layer creates a couple of activation features which can be passed to the following layer. The first layer usually selects fundamental features which includes horizontal or diagonal edges. This output is sent to the subsequent table, which detects extra complex factors consisting of corners or linked edges. As we cross deeper into the community, it is able to apprehend more complex functions like objects, faces, etc. Based on the activation graph of the very last convolution graph, the category layer outputs a hard and fast of self-belief ratings (values among zero and 1) that determine whether the photograph belongs to a "class". For example, if you have a ConvNet that detects lumpy pictures, the output of the remaining layer may additionally comprise any of these lumps in the input picture. Identify brain tumours by using feature extraction using convolutional neural networks (CNN). Deep learning to wake up a affected person or pick out a tumour at an early stage.

## X.RESULT AND DISCUSSION

### For Benign Tumour Detection



### For Malignant Tumour



## XII.CONCLUSION

In this article, the prevailing approach solves most of the problems such as accuracy, tumour grade and tumour detection time. In this article we tell approximately numerous techniques a new technique for detecting advanced brain tumours. To MRI Image Pre-processing Basic segmentation is pre-processed Median filtering methods and accuracy verification this is ninety two percent. Have comparable residences Three thousands of those characteristics are categorized in the list From phrases of sensitive emotions, specific and The fine restriction of validation checking out techniques is the extent of accuracy With low percent mistakes. In this manner the aim region is split and determining the presence of tumours the era proposed right here lets in physicians to broaden a remedy plan and tumour surveillance degree detection. The blessings of this approach are multiplied section length neighbourhood and local picture and for this reason higher another performance ratio to examine. It takes an awful lot much less time Compute and learn faster than different networks with a whole lot much less latency Option. The accuracy of the approach frequently comes into play when using a selection. In this requires higher accuracy at decrease speeds Errors in the usage of terrific category methods.

## XIII.REFERENCES

1. A Survey of Automatic MRI based Brain Tumour Segmentation Techniques/ M Subashree; J Sangeetha /2018/  
<https://ieeexplore.ieee.org/document/9034303>
2. Brain Tumour Diagnose Applying CNN through MRI/ Hanming Hu; Xin Li; Wenyi Yao; Zhuocheng Yao/2021  
<https://ieeexplore.ieee.org/document/9797656/>
3. Comparison of deep learning models for brain tumour classification using MRI images/Necip Çınar; Buket Kaya; Mehmet Kaya/2022/  
<https://ieeexplore.ieee.org/document/9765250>
4. Identification Using MRI Images/T. M. Shahriar Sazzad; K. M. Tanzibul Ahmmed/2019  
<https://ieeexplore.ieee.org/document/8679240>
5. Implementation of Latest Deep Learning Techniques for Brain Tumour Identification from MRI Images/P. Venkateswarlu Reddy; D. Ganesh; L. Sumaiya Azeem/2023  
<https://ieeexplore.ieee.org/document/10192620>
6. Convolutional Neural Networks for Brain Tumour Segmentation Using Different Sets of MRI Sequences/Masoomah Rahimpour; Karolien Goffin; Michel Koole/2019  
<https://ieeexplore.ieee.org/document/9059769>
7. Brain Tumour Segmentation in MRI images/Muhammad Zawish; Asad Ali Siyal/2018  
<https://ieeexplore.ieee.org/document/8610987>
8. Optimizing Convolutional Neural Networks for Brain Tumour Segmentation in MRI Images/Mohamed Babikir Ali; Ruba Ali Hamad; Mohammed Ahmed/2018  
<https://ieeexplore.ieee.org/document/8515826>
9. A Database for Brain Tumour Diagnosis and Detection/R. Tamilselvi; A. Nagaraj; M. Parisa Beham/2020/  
<https://ieeexplore.ieee.org/document/9167530>
10. MRI Images Based Brain Tumour Detection Using CNN for Multiclass Classification/Nikita Mahajan; Hariram Chavan/2023/  
<https://ieeexplore.ieee.org/document/10270492>

d.PRESENTATION CERTIFICATE:

