Brain Tumor Detection using Machine Learning Models

B.Sc. Semester VI Project

Rajarshi Sardar Bhargav Basu Shoptorshi Bhattacharjee Brahmajit Das

Gurudas College, Calcutta University

Domain Description

1. **Neurological Examination**: It is a series of test to measures the function of the patients nervous system and also his/her physical and mental alertness.

Domain Description

1. **Neurological Examination**: It is a series of test to measures the function of the patients nervous system and also his/her physical and mental alertness.
2. **Machine Learning**: Machine learning approaches address these problems by mainly using hand-crafted features (or pre-defined features).

Domain Description

1. **Neurological Examination**: It is a series of test to measures the function of the patients nervous system and also his/her physical and mental alertness.
2. **Machine Learning**: Machine learning approaches address these problems by mainly using hand-crafted features (or pre-defined features).
3. **Brain Scan**: Brain scan is a picture of the internal structure of the brain. A specialized machine takes a scan in the same way as a digital camera takes a photograph.

Motivation

The motivation is to develop a software with better segmentation capability for use in medical imaging to detect diseases like brain tumor. Image segmentation has been identified as the key problem of medical image analysis and remains a popular and challenging area of research. Image segmentation is increasingly used in many clinical and research applications to analyze medical imaging datasets; which motivated us to present a snapshot of dynamically changing field of medical image segmentation.

The motivation of this work is to increase patient safety by providing better and more precise data for medical decision.

Scope of Work

- Deliverables: Working program to take an MRI scan as input and predict presence of tumorous cells with $\geq$ 90% accuracy.
- Scope: The working program has external dependencies (libraries) and it's expected to have a them installed for the program to work.
- Timeline
  - April 27, 2021 Project Assigned
  - May 2, 2021 Project finalized by supervisor, and group is divided into groups of two.
  - May 3, 2021 Data collection started.
  - May 12, 2021 Project Repository created and coding is started.
  - July 7, 2021 Coding is finished, documentation is started.
  - Just 21, 2021 Documentation complete.

Background

We propose the use of ML algorithms to overcome the drawbacks of traditional classifiers. We investigate and compare the performance of two machine learning algorithms namely MLP and Naive Bayes in this work. Since these ML algorithms are found to perform well in most of the pattern classification tasks. Neural networks are useful as they can learn complex mappings between input and output.

They are capable of solving much more complicated classification tasks. However, when certain rules cannot be modeled exactly, the concept of probability is used, which is the basis for Naive Bayes classification.

Methodology

Image Acquisition

The MRI brain images are acquired and are given as input to pre-processing stage

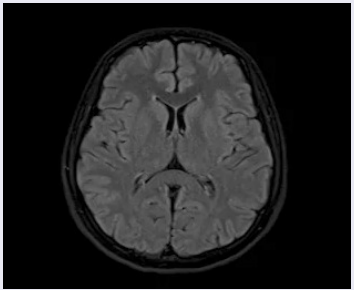

Figure 1: A MRI Scan

Methodology

Pre-processing

Preprocessing is needed as it provides improvement in image data which enhances some of the image features which are important for further processing.

Figure 2: A MRI than have been through image segmentation or thresholding

Methodology

Feature Extraction

When input to an algorithm is very large and redundant to be processed, it is transformed into reduced representative set of features called feature vector.
These features are extracted using Gray Level Co-occurrence Matrix (GLCM) as it is robust method with high performance.

Energy

$$E = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} p(i,j)^2 \quad \text{here, range} = [0,\ 1] \tag{1}$$

Contrast

$$Con = \sum_{n=0}^{N_g-1} n^2 \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} p(i,j)^2 \quad \text{here, range} = [0,\ 1] \tag{2}$$

Methodology

Feature Extraction

Correlation

$$C = \frac{1}{\sigma^x \sigma^y} \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} (i,j) p(i,j)^2 - \mu_x \mu_y \text{ here, range} = [-1, 1] \tag{3}$$

Homogeneity

$$H = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} \frac{p(i,j)}{1 + \quad (\text{mod } i,j)} \text{ here, range} = [0, 1] \tag{4}$$

Methodology

Classification

The Machine learning algorithms are used for classification of MR brain image either as normal or abnormal. The major aim of ML algorithms is to automatically learn and make intelligent decisions. For classification three models are used; **CNN**, **VGG 16** and **Resnet 50**.

Methodology

CNN

CNN or Convolutional Neural Network a class of artificial neural network, most commonly applied to analyze visual imagery.They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation equivariant responses known as feature maps.

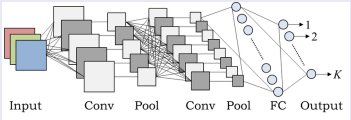

Figure 3: Architecture of a CNN

Methodology

VGG 16

VGG 16 is is a significantly more accurate ConvNet architecture, which not only achieve state-of-the-art accuracy on ILSVRC classification and localisation tasks, but are also applicable to other image recognition datasets, where they achieve excellent performance even when used as a part of a relatively simple pipelines (e.g. deep features classified by a linear SVM without fine-tuning).

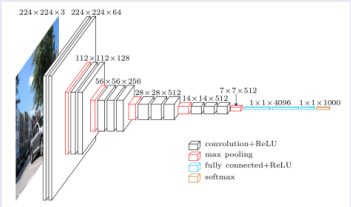

Figure 4: Architecture of VGG 16

Methodology

ResNet 50

ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It has $3.8x10^9$ Floating points operations. It is a widely used ResNet model and we have explored ResNet50 architecture in depth.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 5: Architecture of ResNet 50, in tabular form

Implementation

Assumption and Dependences

It is assumed that the MRI scans are collected and processed before feeding into the module.
The program is dependent on external modules and are expected to be pre-installed on the system. The modules namely include:

- fast.ai
- tensorflow
- OpenCV
- matplotlib

Implementation

- Acquire Data (MRI Scans)
- Divide them into classes for training, while keeping a batch for prediction
- Build the models (using a library, through code)
- Train the model with the gathered data
- Test the trained model with the prediction batch

This is just an over view, a more detailed implementation is described in the description.

Results

Table 1: Model accuracy.

Model	Implemented using Tensorflow	Implemented using fast.ai
CNN	90.31	92.66
VGG 16	97.38	98.16
ResNet 50	97.54	99