BRAIN TUMOR DETECTION USING DEEP LEARNING MODELS

Sneha Grampurohit
Electronics and Communication
Engineering
K.L.E Institute of Technology
Hubli, India
snehagrampurohit5@gmail.com

Venkamma Shalavadi
Electronics and Communication
Engineering
K.L.E Institute of Technology
Hubli, India
priyankashalavadi@gmail.com

Vaishnavi R. Dhotargavi
Electronics and Communication
Engineering
K.L.E Institute of Technology
Hubli, India
vaishnavidhotargavi98@gmail.com

Megha Kudari
Electronics and Communication
Engineering
K.L.E Institute of Technology
Hubli, India
meghakudari0@gmail.com

Mrs Soumya Jolad
Asst. Prof. Electronics and
Communication Engineering
K.L.E Institute of Technology
Hubli, India
soumyajolad4@gmail.com

Abstract: A brain tumor is a disease caused due to the growth of abnormal cells in the brain. There are two main categories of brain tumor, they are non-cancerous (benign) brain tumor and cancerous(malignant**)** brain tumor. Survival rate of a tumor prone patient is difficult to predict because brain tumor is uncommon and are different types. As per the cancer research by United Kingdom, around 15 out of every 100 people with brain cancer will be able to survive for ten or more years after being diagnosed. Treatment for brain tumor depends on various factors like: the type of tumor, how abnormal the cells are and where it is in the brain etc. With the growth of Artificial Intelligence, Deep learning models are used to diagnose the brain tumor by taking the images of magnetic resonance imaging. Magnetic Resonances Imaging (MRI) is a type of scanning method that uses strong magnetic fields and radio waves to produce detailed images of the inner body. The research work carried out uses Deep learning models like convolutional neural network (CNN) model and VGG-16 architecture (built from scratch) to detect the tumor region in the scanned brain images. We have considered Brain MRI images of 253 patients, out of which 155 MRI images are tumorous and 98 of them are non-tumorous. The paper presents a comparative study of the outcomes of CNN model and VGG-16 architecture used.

Keywords: CNN, VGG-16, Tumor cells, Data pre-processing, Convolution.

I. INTRODUCTION

A brain tumor is a disease caused due to the abnormal growth of mass in the brain. Normally in our body, new cells are produced which replace the old and damaged cells in a controlled manner. But in case of brain tumor, tumor cells go on multiplying uncontrollably. As per the National Brain Tumor Society nearly 70,000 people in United States are suffering from primary brain tumor. Brain tumor is ranked as 10th most common tumor in India. The presence of tumor is noticed by the Magnetic Resonance Imaging [MRI] scanning. The MRI scanning should be diagnosed by the physician and later based on the results; the treatments shall be started. This procedure can be a little time consuming. Hence to overcome this, the proposed work presents an automated system that will classify if the subjected patient is suffering from brain tumor. This system can assist the physician to make early decisions so that the treatments are carried out at an earlier stage. The proposed approach uses CNN and VGG-16 architecture and weights to train the model for this binary problem. Accuracy is used as the metric for evaluation.

In the presented approach, we have augmented the dataset (MRI images of brain), performed certain data pre-processing steps to convert the raw data, further investigated two deep learning models namely CNN and VGG-16 and have presented the comparative analysis in the results section. Depending upon the algorithm complexity, computation time and other results one can choose any of the above-mentioned algorithm in their work. This automatic detection system can assist the physician to make early decisions and hence start the treatments at an early stage.

II. OVERVIEW

The dataset considered consists of MRI scanned images of 253 patients out of which 155 of them are tumorous and 98 of them are non-tumorous. The presented work aims to develop a detection model that detects the tumour in the MRI scanned image of a patient. The detection model in general can be given as:
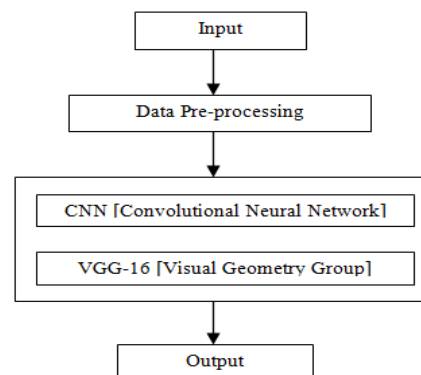

Fig 1. Flowchart of General Brain tumour detection model

A. Input

It is assumed that the patient is all fit and capable to undergo an MRI scan as per the doctor's assistance. The present work considers the Brain MRI images of a patient as the input.

B. Data Pre-processing

For the easy interpretation of the data, the data should be transformed from its raw state. The data pre-processing steps we considered are:
1. Importing libraries
2. Data augmentation
3. Import the augmented data
4. Convert the images to grayscale
5. Removal of noise using dilations and erosions And smoothening of images.
6. Grab the largest contour.
7. Find the extreme points of the contoured image
8. Resize the image
9. Crop the images using the extreme points
10. Splitting of dataset.

C. Algorithms used:

The algorithms used in the proposed work are:
1. Convolutional Neural Network [CNN].
2. VGG-16 Architecture.

D. Output

The system is trained to detect the tumour in the MRI of the patient and thus predict whether the patient is suffering from tumour or no.

III. METHODOLOGY

A. Data pre-processing

Data pre-processing is a method in which the raw data is converted to a useful data by applying some pre-processing techniques. The pre-processing techniques we have used are:

Step1: *Importing libraries*
The libraries such as TensorFlow, NumPy, pandas, matplotlib, os and scikit-learn etc are imported.

Step 2: *Data augmentation*
The images in the dataset are increased by creating modified versions of the image using techniques like rotation etc, this process is known as Image data-augmentation. *Image Data Generator* class is used to generate the images. Before augmentation, 155 tumorous and 98 non-tumorous images were present and after augmentation we have 1085 tumorous and 980 non-tumorous images
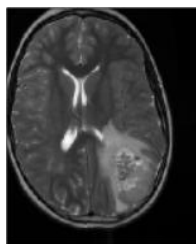
Step 3: *Import the augmented data*
.

Fig2: The original image
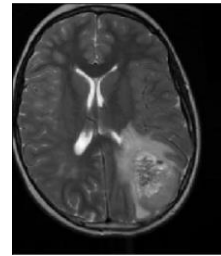
Step 4: *Convert the images to grayscale*

Fig 3 : Original image converted to gray scale

Step 5: *Removal of noise and Smoothening the image*
Noise is removed using series of erosions and dilations. Erosion involves removal of pixel at the edges of the image. Dilation is the reverse process of erosion where it adds the pixels at the edges of the image. Smoothening of the image is done using Gaussian blur technique. In Gaussian blur operation the image is convolved with the Gaussian filter. Gaussian filter is a low pass filter that removes high frequency components in the image. Thus, smoothens the image.
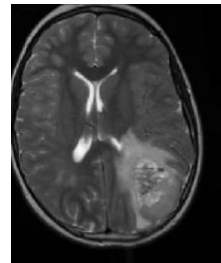

Fig 4: Image after applying Gaussian blur and removal of noise

Step 6: *Grab the largest contour.*
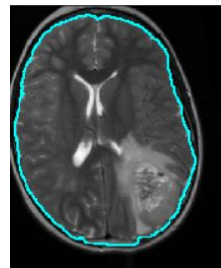Contour refers to the boundary or the outline of a shape.

Fig 5: Find the largest contour

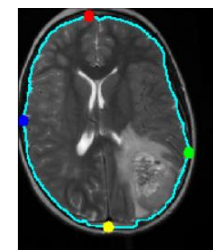**Step 7**: *Find the extreme points of the contoured image.*

Fig 6: Extreme points of contoured image

130

Step 8: *Resize the image*
All the images before pre-processing were of different size, hence all the augmented images were resized to 240x240 for CNN and 224x224 for VGG-16.

Step 9: *Crop the images using the extreme points*
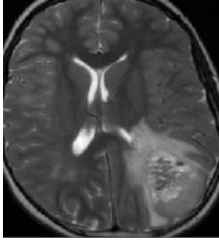The images were cropped along the extreme points

Fig 7: Cropped image

Step 10: *Splitting of dataset:*
The dataset is divided into 3 sets of data. They are
Training data:
No. Of images: 1445
No. Of tumorous images: 758
No. Of non-tumorous images: 687
Testing data:
No. Of images: 310
No. Of tumorous images: 175
No. Of non-tumorous images: 135
Validation data:
No. Of images: 310
No. Of tumorous images: 152
No. Of non-tumorous images: 158

B.   Convolutional Neural Network (CNN)
Each layer in a CNN applies a different set of filters, typically hundreds or thousands of them and combines the results, feeding the output into the next layer in the network. During training, a CNN automatically learns the values for these filters.
CNN gives us two key benefits
1. *Local variance*: The concept of local invariance allows us to classify an image as containing object regardless of where in the image the object appears. We obtain this local invariance through the usage of "pooling layers" which identifies regions of our input volume with a high response to a filter.
2. *Compositionality*: Each filter composes a local path of lower level features into a higher-level representation, this composition allows our network to learn more rich features deeper in network.

Kernel: A kernel can be visualized as a small matrix that slides across, from left to right and top to bottom of a large image. At each pixel in the input image, the neighborhood of the image is convolved with the kernel and the output is stored.
Example: Convolving (denoted mathematically as * operator) a 3x3 region of an image with a 3x3 kernel.

$$O_{ij} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 93 & 139 & 101 \\ 26 & 252 & 196 \\ 135 & 230 & 18 \end{bmatrix}$$

$$= \begin{bmatrix} 10.3 & 15.4 & 11.2 \\ 2.6 & 28.0 & 21.7 \\ 15.0 & 25.5 & 2.0 \end{bmatrix}$$

$$= 132$$

After applying this convolution, the pixel located at the co-ordinate (i, j) of the output image O to is set to $O_{ij}=132$. By applying convolutions filters, nonlinear activation functions, pooling and back propagations, CNN's are able to learn filters that can detect edges and blob like structures in lower level layers of the network and then use the edges and structures as "building blocks", eventually detecting high level objects in the deeper layer of the network.
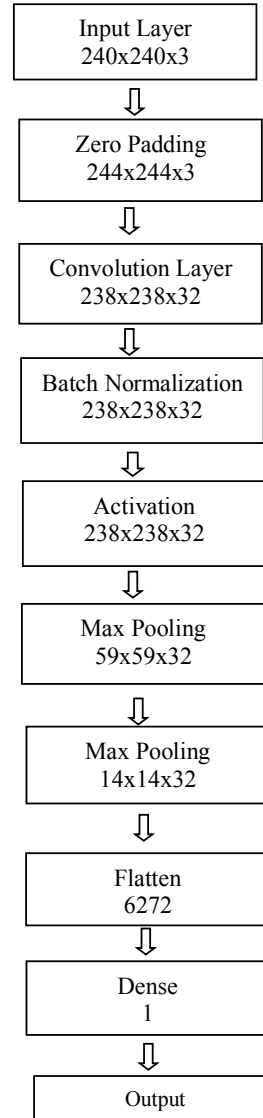The flow graph of Convolutional Neural Network

Fig 8: The flow graph of CNN

1. *Input layer:*
We have resized the images to 240x240 pixels height=240pixels and width=240 pixels with depth=3 [number of channels of the image].

131

2. Zero padding:

We need to "pad" the borders of an image to retain the original image size when applying a convolution. The same is true for filters inside a CNN. Using zero, we can "pad" our input along the borders such that output volume size matches input volume size.

We have performed zero padding on our images using pool size (2,2).

Example: Consider an image represented in the form of matrix:

95	242	186	152	39
39	19	220	153	180
5	247	212	59	46
46	77	133	110	79
156	35	79	93	116

Fig 9: A 5x5 image in its matrix form

Padding the image with zeroes i.e. of pool size (2,2)

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	95	242	186	152	39	0	0
0	0	39	19	220	153	180	0	0
0	0	5	247	212	59	46	0	0
0	0	46	77	133	110	79	0	0
0	0	156	35	79	93	116	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Fig 10: Zero padding applies to image

3. Convolution:

The convolution layer is considered as the core building block of a "Convolutional Neural Network". The convolution layer parameter consists of a set of K learnable filter (i.e. kernels). Consider the forward pass of the CNN

Step1: K-Kernels each Waiting to be applied to The image.

Step2: Kernel applied to the input volume

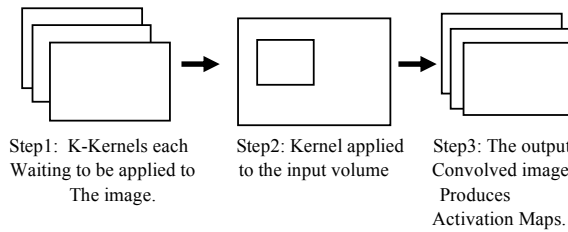Step3: The output Convolved image Produces Activation Maps.

Fig 8: Convolution of k-kernels with an image

In the convolution layer of our presented work, we have used 32 filters of size (7,7). The size (7,7) refers to receptive field [local region of input volume to which each neuron is connected]. We have used (7x7) receptive field, i.e. each neuron will connect to a 7x7 local region by the image for a total 7x7x3 =147 weights. After applying all 32 filters to the input volume, we now have 32, 2-dimenstional activation maps. Every entry in the output volume is thus an output of a neuron that "looks" at only a small region of an input. In this manner the network "learns" filters that activate when they see a specific type of feature. After the convolution operation, the size of image is now (238, 238,32) depth has increased due to the filter used.

Stride: we have used the stride values s=1
Example: Consider an Image being convolved with Laplacian Kernel with stride=1.

95	242	186	152	39
39	19	220	153	180
5	247	212	59	46
46	77	133	110	79
156	35	79	93	116

Fig 9: Image

0	1	0
1	-1	1
0	1	1

Fig 10: Laplacian kernel

Using s=1, our kernel slides from left to right and top to bottom, one pixel at a time producing the below output,

692	-315	-6
-680	-194	305
153	-59	-86

Fig 11: The resultant matrix

Batch normalization: Batch normalization layers (or BN in short) as the name suggests, are used to normalize the activation of a given input volume before passing it into next layers in network.

If we consider x to be our mini batch of activation, then we can compute the normalized $\hat{x}$ via the following equation.

$$\hat{x_i} = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \varepsilon}} \qquad (1)$$

During training, we compute the μ_β and σ_β over each mini batch β, where

$$\mu_\beta = \frac{1}{M}\sum_{i=1}^{m} x_i \qquad (2)$$

$$\sigma_\beta^2 = \frac{1}{M}\sum_{i=1}^{m}(x_i - \mu_\beta)^2 \qquad (3)$$

We set ε equal to a small positive value to avoid taking the square root of zero. Applying this equation implies that the activation having a batch normalization layer will have approximately zero mean and unit variance. Batch normalization also has the added benefits of helping "stabilize" training, allowing for a larger variety of learning rates and regularization strengths.

4. Activation:

After convolution layer, we have applied a nonlinear activation function. An activation layer accepts an input volume of size W_{input} x H_{input} x D_{input} and then applies the activation function (below fig). Since the activation function is applied in an element wise manner, the output of an activation layer is always same as input dimension, i.e.,

$W_{input} = W_{output}$, $H_{input} = H_{output}$, $D_{input} = D_{output}$.

Example: Consider an image to which RELU activation with max (0, x) function applied

-249	-91	-37
250	-134	101
27	61	-153

Fig 12: An example image

0	0	0
250	0	101
27	61	0

Fig 13: Resultant matrix

132

5. *Pooling layers*

There are two methods to reduce the size of an input volume.

- Convolutional layers with stride >1

- POOL layers

The primary function of the POOL layers is to progressively reduce the spatial size (i.e. width and height) of the input volume. Doing this allows us to reduce the amount of parameters and computation in the network. Pooling also helps us control over fitting.

We have used Max function in the pooling layer i, e Max-pooling with the pool size of FxS i.e. [receptive field size into stride]. Applying the pooling operation yields an output volume of size W_{output} x H_{output} x D_{output},

Where

$$W_{output} = ((W_{input}-F)/S) +1$$
$$H_{output} = ((H_{input}-F)/S) +1$$
$$D_{output} = D_{input}$$

Example: Consider an image to which a mac pooling of pool size (2x2) is applied with stride=1

181	237	170	223
229	181	89	108
109	93	48	66
108	21	71	19

237	237	223
229	181	108
109	93	71

Fig14: Input image matrix   Fig 15: Resultant Matrix

In the presented work in CNN, we have used 2 max pooling layers with pool size 4x4 and stride=1.

6. *Flatten*

This layer is used in order to flatten 3-dimension matrix to 1-dimensionmatrix. In the presented work, after flatten layer the array size obtained is 6272.
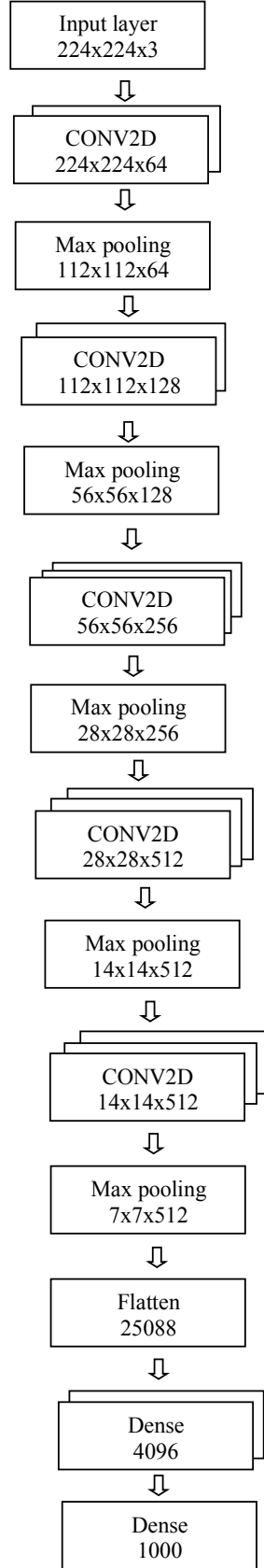
7. *Dense*

Dense is an output unit. It is fully connected with one neuron with sigmoid activation as we have binary problem of detecting brain tumor detection. If dense output is 1 then it indicates tumorous MRI. If dense output is 0 then it indicates non tumorous MRI.

C. *VGG-16 ARCHITECTURE*

The VGG network was introduced by Simonyi and Zisserman in this 2014 paper, "Very deep convolutional network for large scale image recognition".

The VGG family of Convolutional Neural Networks can be characterized by two key points:

1. All convolution layers in the network use only 3X3 filters.

2. Stacking multiple convolution + RELU layer sets (the number of consecutive convolution layers+ RELU layers normally increases the deeper we go) before applying a pool operation.

In the presented work, we haven't used the transfer learning technique, instead we have built the VGG-16 architecture and made the necessary changes in the architecture to have the better accuracy. The architecture of VGG-16 architecture used id given by:
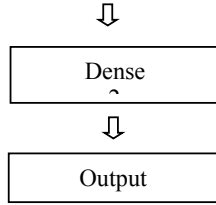
Input layer
224x224x3
⇩
CONV2D
224x224x64
⇩
Max pooling
112x112x64
⇩
CONV2D
112x112x128
⇩
Max pooling
56x56x128
⇩
CONV2D
56x56x256
⇩
Max pooling
28x28x256
⇩
CONV2D
28x28x512
⇩
Max pooling
14x14x512
⇩
CONV2D
14x14x512
⇩
Max pooling
7x7x512
⇩
Flatten
25088
⇩
Dense
4096
⇩
Dense
1000

133

Fig 16: Flow graph of VGG-16 Architecture

IV.   RESULTS

The comparative results after using the same dataset on CNN model and VGG-16 architecture are as shown below.
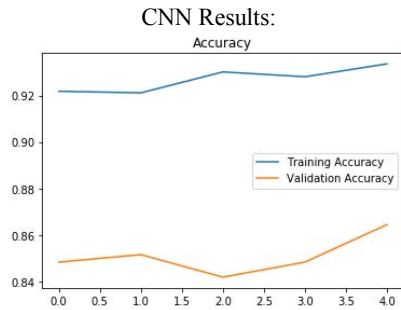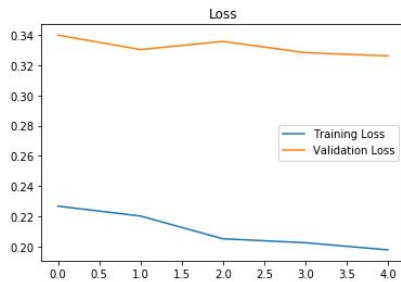
CNN Results:

FIG 17: Accuracy chart

Fig 18: Loss chart

From the above plots of results, it can be inferred that:

- The accuracy obtained is 93 percent with validation accuracy equal to 86 percent. This accuracy is obtained at the 23rd epoch of the training.

VGG-16 Results:

From the VGG-16 result plots below one can infer that:

- All though the accuracy rate is increasing up to 100 percent (Fig 20), we have estimated the best model at 17th epoch which yields 97.16 percent training accuracy and 97.42 percent validation accuracy as after the 17th epoch, although the accuracy was increasing, validation accuracy was gradually decreasing.
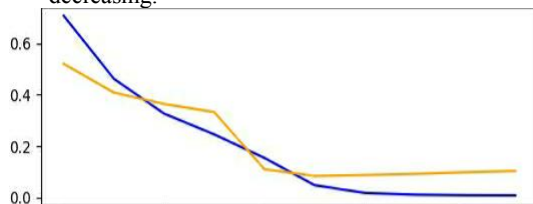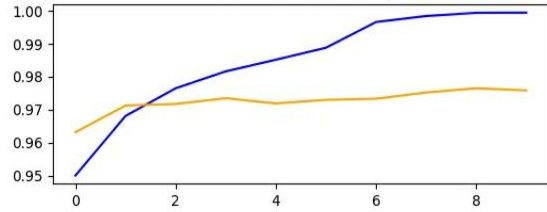

Fig 19: Loss chart

Fig 20: Accuracy chart

In fig 19 and 20 blue color refers to Training loss and Training accuracy and orange lines represent Validation loss and Validation accuracy.

Table 1: Parameter wise comparison

Parameter	CNN	VGG-16
No. of images used	2065 Training-1445 Test-310 Validation- 310	
Time consumed [From pre-processing till obtaining results]	0:5:03 [5mins: 3secs] GPU (GOOGLE COLAB)	0:15:25 [15mins:25secs] GPU (GOOGLE COLAB)
Epochs carried out	25	
Accuracy	0.9336	0.9716
Validation Accuracy	0.8645	0.9742
Test set accuracy	0.916	0.919

V.   CONCLUSION

In proposed work, Deep neural networks such as CNN and VGG-16 are investigated on MRI images of Brain. Both the models have given an effective result, However VGG-16 takes a greater computational time and memory but has given satisfactory results compared to CNN. Due to the availability of huge data being produced and stored by the medical sector, Deep learning will play an important role in data analysis in the upcoming days.

VI.   REFERENCE

1.Yuehao Pan, Weimin Huang et.al "Brain Tumor Grading based on Neural networks and Convolutional Neural Networks",*IEEE*.

2. Darko Zikic, Yani Ioannou et.al "Segmentation of Brain tumor tisssues with convolutional neural networks", *MiCCAI,2014,Brats challenge.*

3. Karen, Zisserman "Very deep convolutional networks for large scale image recognition",*ICLR*.[Published in 2014].

4. Taranjit Kaur, Tapan Kuamar Gandhi, "Automated brain image classification based on VGG-16 and tranfer learning*",ICIT* [Published in 2019].

134