Project Report

On

Classification of Brain MRI Images using Convolutional Neural Network

Submitted for the partial fulfilment of degree of

Bachelor of Computer Science and Engineering

Final Year

Submitted by
Arpit Maheshwari
14R/00025, 15CSE53005

Guided by
Alok Singh Gahlot
Assistant Professor

Department of Computer Science and Engineering,

MBM Engineering College, Faculty of Engineering,

Jai Narain Vyas University, Jodhpur (Rajasthan)

Session 2017-18

Abstract

A Glioma is a type of tumor that starts in the glial cells of the brain or the spine. Gliomas comprise about 30 per cent of all brain tumors and central nervous system tumors, and 80 per cent of all malignant brain tumors.

Gliomas are further categorized according to their grade, Low-grade gliomas [WHO grade II] and High-grade [WHO grade III–IV] gliomas.

The classification of different grades of Glioma in patients with might not be an easy task. Some Glioma diseases have similar disease patterns which lead to the misdiagnosis of these diseases. Nowadays, images are visually evaluated by an expert reader and this process is not entirely quantitative or reproducible. Even the experts can have up to 20% misclassification. Automated diagnosis by pattern recognition can produce quantitative and reproducible results, but if training data comes from clinical routine, it may produce less accurate results to discriminate similar disease patterns. Hence, the work of this thesis aims to find an optimal subset of features, for a given training data set, which improves the classification of different grades of Glioma in patients with suspected Glioma.

This project uses some specific methods of image pre-processing on Brain MRI scans. The processed images are fed into a Convolutional Neural Network which extracts features from them. These features are used as an input to an Artificial Neural Network which classifies the grade of disease.

DECLARATION

I, **Arpit Maheshwari** hereby declare that this project titled "**Classification of Brain MRI Images using Convolutional Neural Network**" is a record of original work done by me under the supervision and guidance of **Mr. Alok Singh Gahlot**. I further certify that this work has not formed the basis for the award of the Degree/Diploma/Associateship/Fellowship or similar recognition to any candidate of any university and no part of this report is reproduced as it is from any other source without appropriate reference and permission.

Arpit Maheshwari

14R/00025

15CSE53005

CERTIFICATE

This is to certify that report entitled "**Classification of Brain MRI Images using Convolutional Neural Network**" submitted to Department of Computer Science and Engineering, MBM Engineering College, JNV University in partial fulfilment of the requirements for the award of the degree of B.E. CSE is the bona fide record of the work done by Mr. Arpit Maheshwari under my supervision and guidance and no part of this report is reproduced as it is from any other source without appropriate reference and permission.

Alok Singh Gahlot

Assistant Professor

Table of Contents

1.	Introduction	1
2.	Magnetic Resonance Imaging (MRI)	1
3.	Image Recognition	5
4.	Convolutional Neural Networks	6
5.	Data Preprocessing	10
6.	Dataset Used	12
7.	Libraries used	13
8.	Code	16
9.	Screenshots	23
10.	Conclusion	25
11.	References	26

1. Introduction

At present, medical images have become the contemporary method for disease diagnosis. Having the advantage of noninvasive procedure, medical imaging techniques like MRI, CT, Ultrasound are used to diagnosis different diseases and provide the right treatment. Each disease affecting the body change the characteristics of the tissues it affects. Currently physicians use the medical images to give diagnosis from their observations of the medical images. Hence there is a need for tools that would extract these epitomes or features from medical images of diseased persons and classify them according to the disease for data analysis and resource management.

2. Magnetic Resonance Imaging (MRI)

One type of medical images that can be used to extract these epitomes is MRI. Magnetic resonance imaging (MRI) of the head is a painless, noninvasive test that produces detailed images of your brain and brain stem. An MRI machine creates the images using a magnetic field and radio waves. This test is also known as a brain MRI or a cranial MRI. You will go to a hospital or radiology center to take a head MRI.

An MRI scan is different from a CT scan or an X-ray in that it doesn't use radiation to produce images. An MRI scan combines images to create a 3-D picture of your internal structures, so it's more effective than other scans at detecting abnormalities in small structures of the brain such as the pituitary gland and brain stem. Sometimes a contrast agent, or dye, can be given through an intravenous (IV) line to better visualize certain structures or abnormalities.

An MRI scan works by using a powerful magnet, radio waves, and a computer to create detailed images. Your body is made up of millions of hydrogen atoms (the human body is 80% water), which are magnetic. When your body is placed in the magnetic field, these atoms align with the field, much like a compass points to the North Pole. A radio wave "knocks down" the atoms and disrupts their polarity. The sensor detects the time it takes for the atoms to return to their original alignment. In essence, MRI measures the water content (or fluid characteristics) of different tissues, which is processed by the computer to create a black and white image. The image is highly detailed and can show even the smallest abnormality.

Similar to CT, MRI allows your doctor to see your body in narrow slices, each about one quarter of an inch thick. For example, imagine that you are slicing a loaf of bread and taking a picture of each slice. It can view slices from the bottom (axial), front (coronal), or sides (sagittal), depending on what your doctor needs to see.

A dye (contrast agent) may be injected into your bloodstream to enhance certain tissues. The dye contains gadolinium, which has magnetic properties. It circulates through the blood stream and is absorbed in certain tissues, which then stand out on the scan.

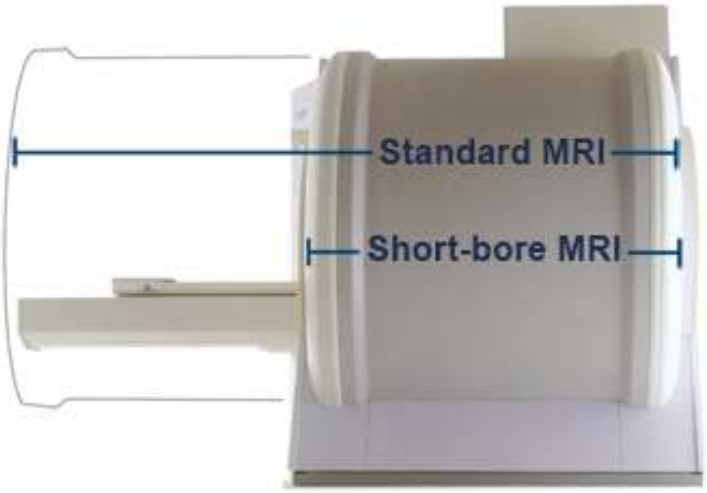
MR angiogram (MRA). MRI can be used to view arteries and veins. Standard MRI can't see fluid that is moving, such as blood in an artery, and this creates "flow voids" that appear as black holes on the image. Contrast dye (gadolinium) injected into the bloodstream helps the computer "see" the arteries and veins. Contrast is also used to view tumors and arteriovenous malformations (AVMs).

2.1 The MRI can be applied on:

- **Head and neck** . MRI can be used to detect brain tumors, traumatic brain injury, developmental anomalies, multiple sclerosis, stroke, dementia, infection, and the causes of headache.
- **Arteries and veins**. MRA can detect aneurysms, blockages of the blood vessels, carotid artery disease, and arteriovenous malformations.
- **Spine**. MRI is sensitive to changes in cartilage and bone structure resulting from injury, disease, or aging. It can detect herniated discs, pinched nerves, spinal tumors, spinal cord compression, and fractures.

2.2 Types of MRI scanners

- **Standard MRI:** this machine looks like a long cylinder with a narrow tube in the center. You lay on a moveable bed and your whole body slides inside the tube. Even though this machine can be confining to some people, it produces the best-looking images.



- **Short-bore MRI:** this machine is similar to the standard, but it's about half the length. If you are having pictures taken of your head, then your feet will stick out one end of the tube; if your back is being imaged, then your head will stick out. You may find this option more tolerable if tight spaces make you anxious.

2.3 Types of MRI

The most common MRI sequences are T1-weighted and T2-weighted scans. **T1-weighted images** are produced by using short TE and TR times. The contrast and brightness of the image are predominately determined by T1 properties of tissue. Conversely, **T2-weighted images** are produced by using longer TE and TR times. In these images, the contrast and brightness are predominately determined by the T2 properties of tissue.

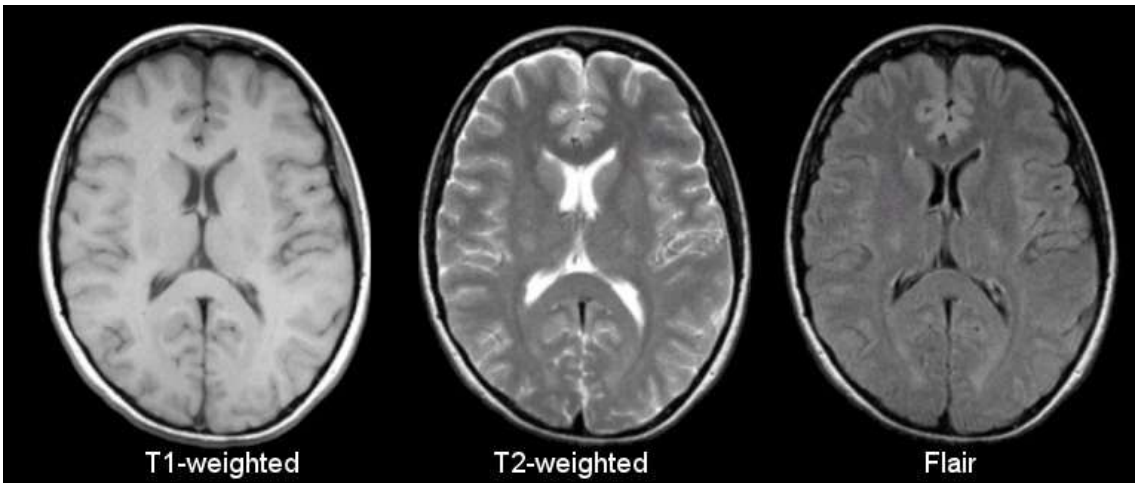
In general, T1- and T2-weighted images can be easily differentiated by looking the CSF. *CSF is dark on T1-weighted imaging and bright on T2-weighted imaging*.

A third commonly used sequence is the **Fluid Attenuated Inversion Recovery (Flair)**. The Flair sequence is similar to a T2-weighted image except that the TE and TR times are very long. By doing so, abnormalities remain bright but normal CSF fluid is attenuated and made dark. This sequence is very sensitive to pathology and makes the differentiation between CSF and an abnormality much easier.

T1-weighted imaging can also be performed while infusing **Gadolinium (Gad)**. Gad is a non-toxic paramagnetic contrast enhancement agent. When injected during the scan, Gad changes signal intensities by shortening T1. Thus, Gad is very bright on T1-weighted images. *Gad enhanced images are especially useful in looking at vascular structures and breakdown in the*

blood-brain barrier [e.g., tumours, abscesses, inflammation (herpes simplex encephalitis, multiple sclerosis, etc.)].

2.4 Comparison of T1 vs. T2 vs. Flair (Brain)



T2-weighted images are used to detect tumor because, in T2-weighted images, areas of the brain filled with water appear bright and tissues with high fat content appear dark. This can aid in localizing pathology since many brain lesions are associated with an increase in water content.

2.5 Limitations of MRI

- Subject to motion artifact.
- Inferior to CT in detecting acute hemorrhage.
- Inferior to CT in detection of bony injury.
- Requires prolonged acquisition time for many images.

3. Image Recognition

In the context of machine vision, image recognition is the capability of a software to identify people, places, objects, actions and writing in images. To achieve image recognition, the computers can utilize machine vision technologies in combination with artificial intelligence software and a camera.

While it is very easy for human and animal brains to recognize objects, the computers have difficulty with the same task. When we look at something like a tree or a car or our friend, we usually don't have to study it consciously before we can tell what it is. However, for a computer, identifying anything (be it a clock, or a chair, human beings or animals) represents a very difficult problem and the stakes for finding a solution to that problem are very high.

Image recognition is a machine learning method and it is designed to resemble the way a human brain functions. With this method, the computers are taught to recognize the visual elements within an image. By relying on large databases and noticing emerging patterns, the computers can make sense of images and formulate relevant tags and categories.

3.1 Image Recognition is a Tough Task to Accomplish

Image recognition is not an easy task to achieve. A good way to think about achieving it is through applying metadata to unstructured data. Hiring human experts for manually tagging the libraries of music and movies may be a daunting task but it becomes highly impossible when it comes to challenges such as teaching the driverless car's navigation system to differentiate pedestrians crossing the road from various other vehicles or filtering, categorizing or tagging millions of videos and photos uploaded by the users that appear daily on social media.

One way to solve this problem would be through the utilization of neural networks. We can make use of conventional neural networks for analyzing images in theory, but in practice, it will be highly expensive from a computational perspective. Take for example, a conventional neural network trying to process a small image (let it be 30*30 pixels) would still need 0.5 million parameters and 900 inputs. A reasonably powerful machine can handle this but once the images become much larger (for example, 500*500 pixels), the number of parameters and inputs needed increases to very high levels.

There is another problem associated with the application of neural networks to image recognition: overfitting. In simple terms, overfitting happens when a model tailors itself very closely to the data it has been trained on. Generally, this leads to added parameters(further increasing the computational costs) and model's exposure to new data results in a loss in the general performance.

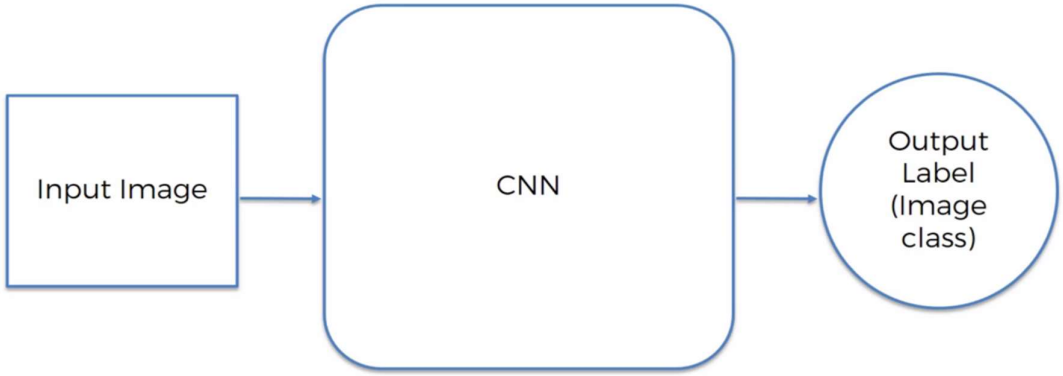
4. Convolutional Neural Networks

To the way a neural network is structured, a relatively straightforward change can make even huge images more manageable. The result is what we call as the CNNs or ConvNets (convolutional neural networks).

The general applicability of neural networks is one of their advantages, but this advantage turns into a liability when dealing with images. The convolutional neural networks make a conscious tradeoff: if a network is designed for specifically handling the images, some generalizability has to be sacrificed for a much more feasible solution.

If you consider any image, proximity has a strong relation with similarity in it and convolutional neural networks specifically take advantage of this fact. This implies, in a given image, two pixels that are nearer to each other are more likely to be related than the two pixels that are apart from each other. Nevertheless, in a usual neural network, every pixel is linked to every single neuron. The added computational load makes the network less accurate in this case.

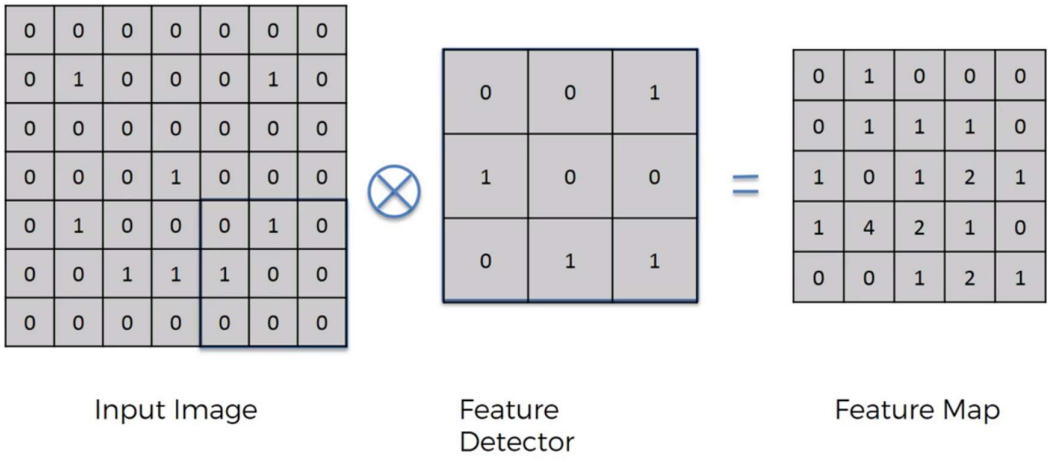
By killing a lot of these less significant connections, convolution solves this problem. In technical terms, convolutional neural networks make the image processing computationally manageable through filtering the connections by proximity. In a given layer, rather than linking every input to every neuron, convolutional neural networks restrict the connections intentionally so that any one neuron accepts the inputs only from a small subsection of the layer before it (say like 5*5 or 3*3 pixels). Hence, each neuron is responsible for processing only a certain portion of an image. (Incidentally, this is almost how the individual cortical neurons function in your brain. Each neuron responds to only a small portion of your complete visual field).

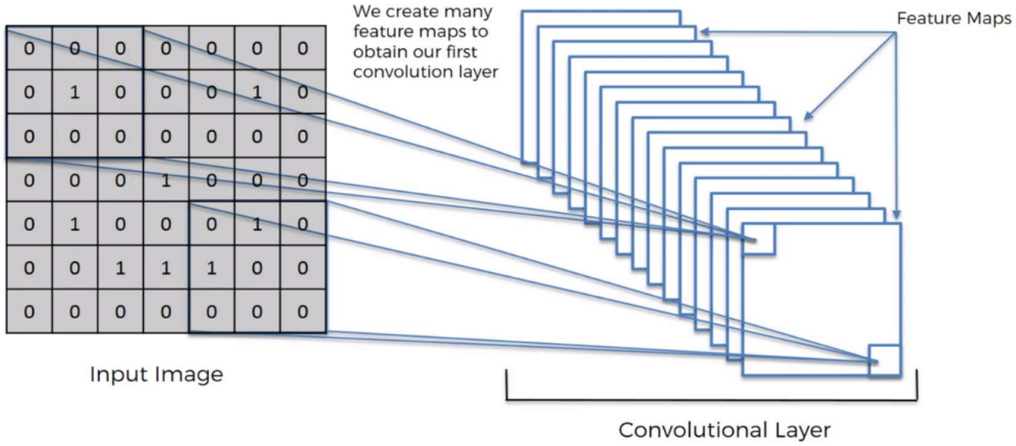


4.1 Steps involved in a CNN: -

A. Convolution:

- The Image is converted in the form of a Matrix.
- The **Feature Detector** is moved over the entire Image part by part.
- The similarities of the **Feature Detector** and the part of Input Image are stored in the **Feature Map** one by one.
- The aim of convolution process is to reduce the size of the image to process it faster while retaining the **features which are integral.**
- We apply different Feature Detectors to obtain the Convolution Layer.





B. Max Pooling

- The first Image shows the collage of various images of ‘Cheetah’. Here one of the features can be the shadow of tears under it’s eyes.
- The neural network should not depend on the relative position of the features with respect to the image.
- We use a 2*2 box as shown and store the maximum value in the box in the Pooled Feature Map corresponding to it’s position. Then we move the box to cover all the parts of the Feature Map.
- By this we still are able to preserve the features through the maximum numbers.

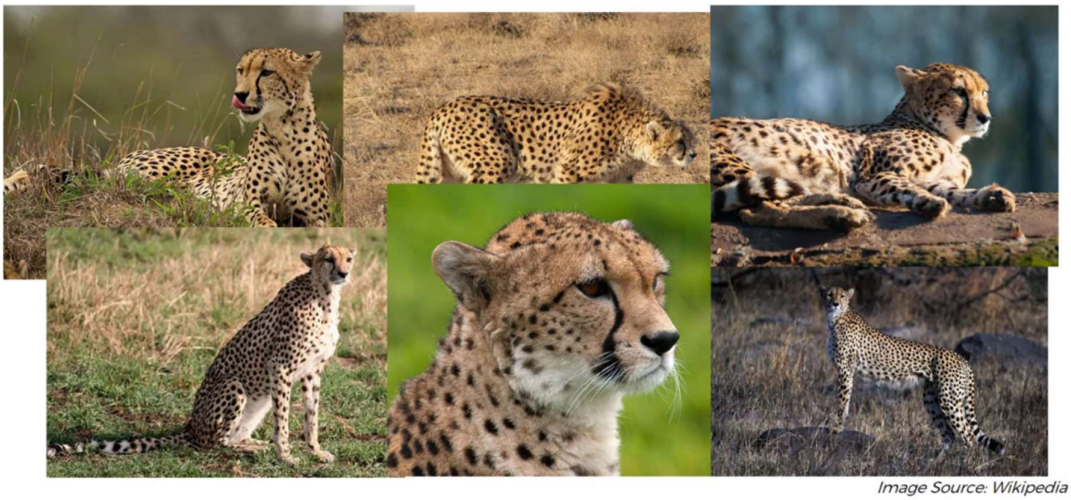
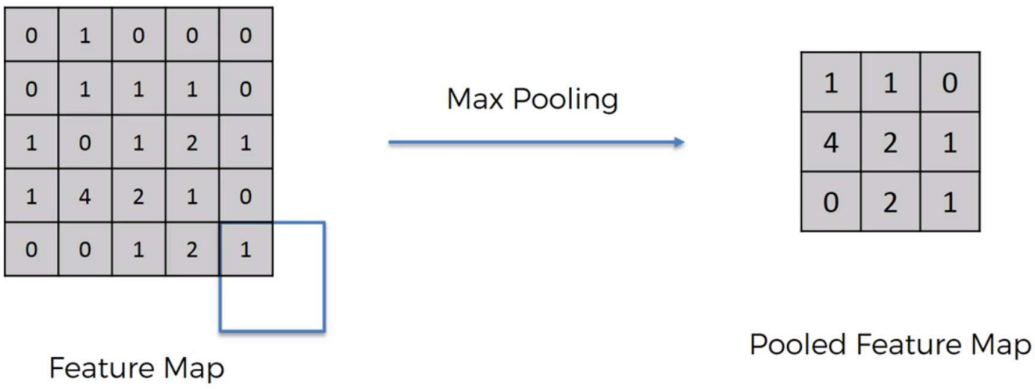
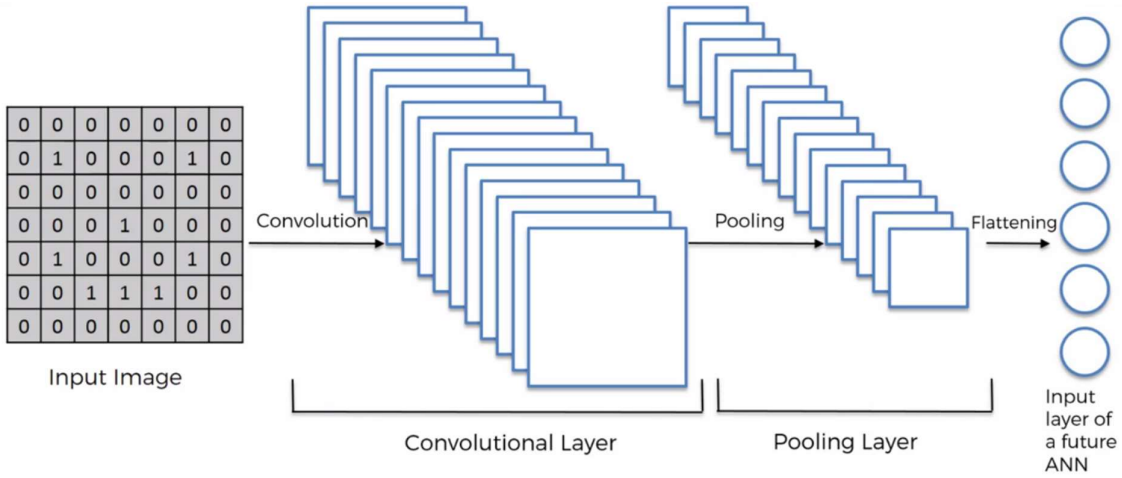


Image Source: Wikipedia



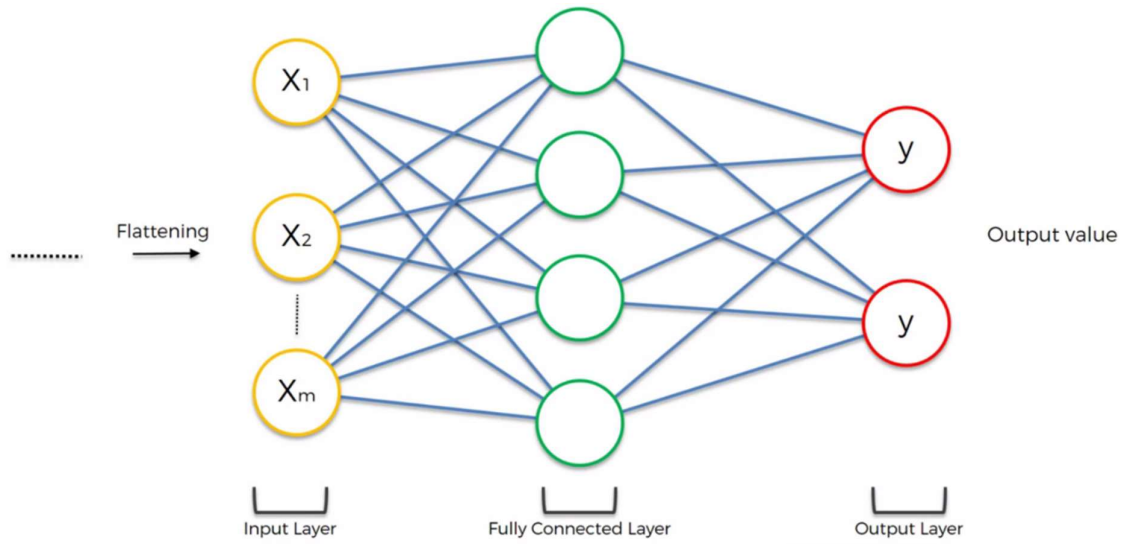
C. Flattening

- The numbers from the Pooled Featured Maps are taken row by row and are flattened into a single column.
- These values are sent to the Input Layer of a future ANN.
- We get one huge vector for the input layer of an artificial neural network.



D. Full Connection

- In ANN it is not necessary to have fully connected layers whereas in CNN it is necessary to have fully connected layers.
- This ANN further enhances the learning and along with the weights, now the feature detectors are also updated.



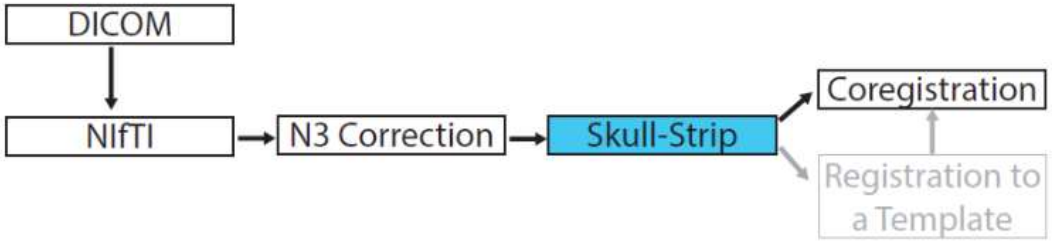
5. Data Preprocessing:

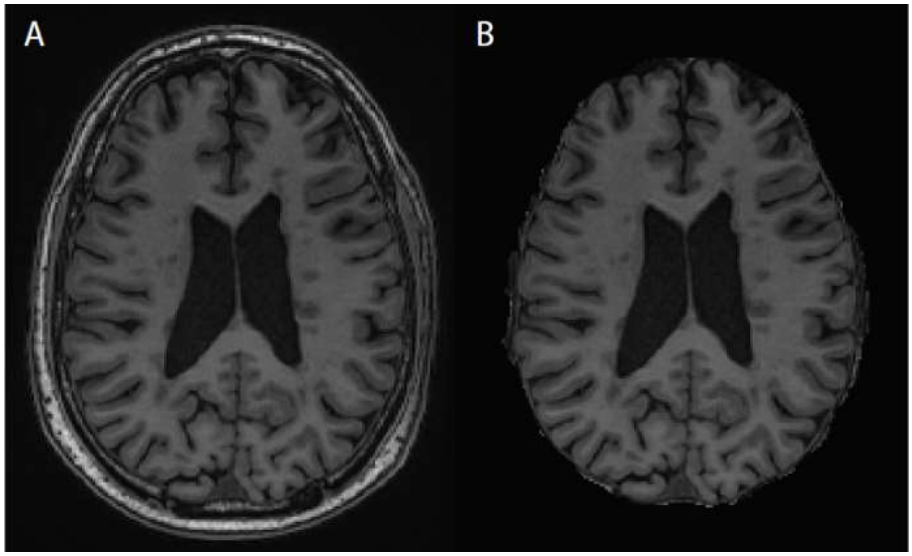
Pre-processing is a collection of transformations applied to an original image to obtain data in a pre-specified analytic format. Different pipelines result in different files.

We divide **image pre-processing** into two main conceptual steps

5.1 Skull Stripping:

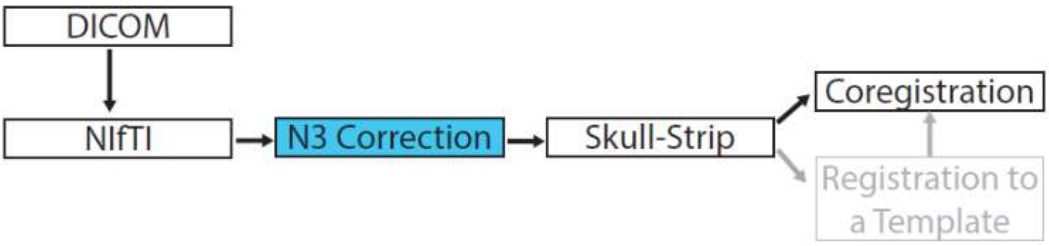
The skull stripping method is an important area of study in brain image processing applications. It acts as preliminary step in numerous medical applications as it increases speed and accuracy of diagnosis in manifold. It removes non-cerebral tissues like skull, scalp, and dura from brain images.
Skull Stripping removes extra-cerebral voxels from the volume.

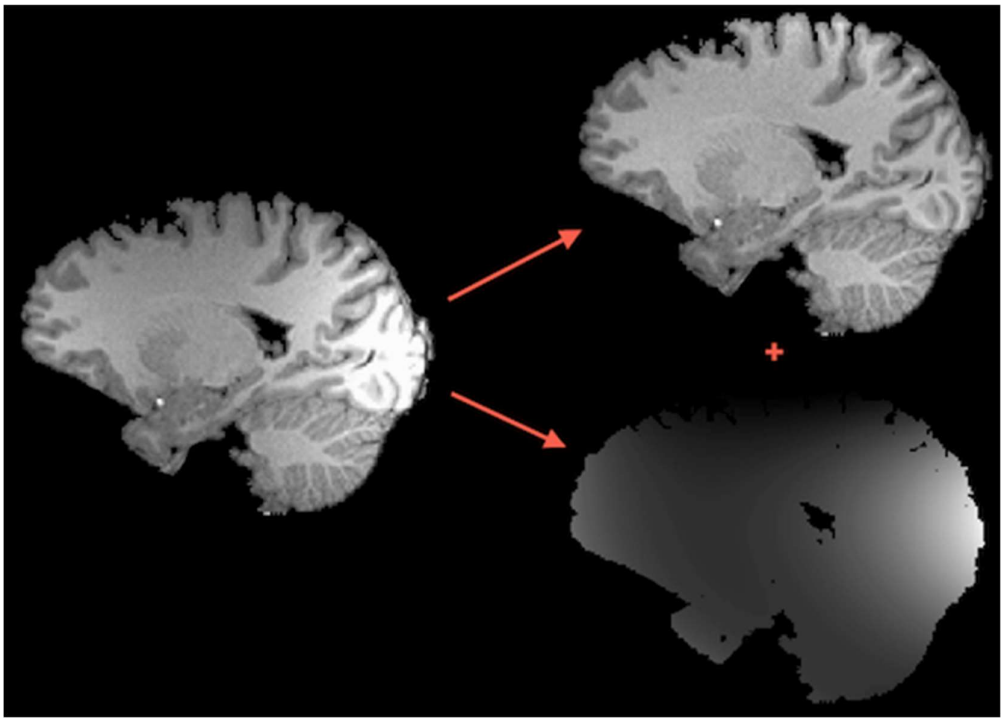




5.2 Inhomogeneity Correction:

Intensity inhomogeneity is a smooth intensity change inside originally homogeneous regions. The intensity inhomogeneity degrades performance of image processing algorithms. Intensity inhomogeneity correction methods are important image processing algorithms which are used to reduce the inhomogeneity. Brain image intensity inhomogeneity correction is one of the most important parts of clinical diagnostic tools. Brain images mostly contain inhomogeneity. Therefore, accurate process of brain images is a very difficult task. However, accurate process of these images is very important and crucial for a correct diagnosis by clinical tools.





6. Dataset Used

The dataset is taken from TCIA collections, cancerimagingarchive.net. TCIA is a service which de-identifies and hosts a large archive of medical images of cancer accessible for public download. The data are organized as “Collections”, typically patients related by a common disease (e.g. lung cancer), image modality (MRI, CT, etc) or research focus. DICOM is the primary file format used by TCIA for image storage. Supporting data related to the images such as patient outcomes, treatment details, genomics, pathology, and expert analyses are also provided when available.

The MRIs used in this project are pre-operative examinations performed in 159 subjects with Low Grade Gliomas (WHO grade II & III). Segmentation of tumors in three axial slices that include the one with the largest tumor diameter and ones below and above are provided in NiFTI format. Tumor grade and histologic type are also available. All of these subjects have biopsy proven 1p19q results, performed using FISH. For the 1p/19q status "n/n" means neither 1p nor 19q were deleted. "d/d" means 1p and 19q are co-deleted.

Detailed Description

Collection Statistics	**Updated 2017/07/31**
Image Size (GB)	2.7
Modalities	MRI, SEG, NIfTI
Number of Images	17360
Number of patients	159
Number of Series	319
Number of studies	160

7. Libraries Used

7.1 Tensorflow:

TensorFlow is the second machine learning framework that Google created and used to design, build, and train deep learning models. We can use the TensorFlow library do to numerical computations, which doesn't seem all too special, but these computations are done with data flow graphs. In these graphs, nodes represent mathematical operations, while the edges represent the data, which usually are multidimensional data arrays or tensors, that are communicated between these edges.

7.2 Keras:

Keras is a high-level Neural network API, written in python. It is built on top of either Theano or Tensorflow. Most powerful and easy to use for developing and evaluating Deep Learning Models. Allows for easy and fast Prototyping. Supports both Convolution Neural Networks and Recurrent Neural Networks, as well as combination of the two. Runs seamlessly on CPU and GPU. Keras is among the libraries supported by Apple's coreML.

7.3 NumPy:

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

7.4 NiPype:

Nipype, an open-source, community-developed initiative under the umbrella of NiPy, is a Python project that provides a uniform interface to existing neuroimaging software and facilitates interaction between these packages within a single workflow. Nipype provides an environment that encourages interactive exploration of algorithms from different packages (e.g., ANTS, SPM, FSL, FreeSurfer, Camino, MRtrix, MNE, AFNI, Slicer), eases the design of workflows within and between packages, and reduces the learning curve necessary to use different packages. Nipype is creating a collaborative platform for neuroimaging software development in a high-level language and addressing limitations of existing pipeline systems.

7.5 FSL:

FSL is a comprehensive library of analysis tools for FMRI, MRI and DTI brain imaging data. It runs on Apple and PCs (both Linux, and Windows via a Virtual Machine), and is very easy to install. Most of the tools can be run both from the command line and as GUIs ("point-and-click" graphical user interfaces).

7.6 Pandas:

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data. Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

7.7 Nibabel

This package provides read +/- write access to some common medical and neuroimaging file formats, including: ANALYZE (plain, SPM99, SPM2 and later), GIFTI, NIfTI1, NIfTI2, MINC1, MINC2, MGH and ECAT as well as Philips PAR/REC. We can read and write FreeSurfer geometry, annotation and morphometry files. There is some very limited support for DICOM. NiBabel is the successor of PyNIfTI. The various image format classes give full or selective access to header (meta) information and access to the image data is made available via NumPy arrays.

7.8 Cv2:

OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day. Currently OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations. OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

7.9 Math:

It provides access to the mathematical functions defined by the C standard.

7.10 Sklearn:

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

8. Code

8.1 Image_preprocessing.py

```
import os # for doing directory operations
#import pandas as pd
import numpy as np
from nipype.interfaces import fsl #import matplotlib.pyplot as plt
file_path='/home/arpit/Desktop/Project'
data_dir = file_path+'/NiFTiSegmentationsEdited/'
patients = os.listdir(data_dir) #len(patients)
for patient in patients[:]:
    #print(patient)
    path = data_dir + patient
    input_file = path + '/' + patient+'_T2.nii.gz'
    print(input_file)
    #Preprocessing
    #Skull Stripping  /home/arpit/Desktop/Project/Skull _Stripped
    output_strip_file = "skull_stripping_"+patient+"_T2.nii.gz"
    skullstrip = fsl.BET(in_file=input_file,
out_file=file_path+'/Skull_Stripped/'+output_strip_file,mask=True)
    skullstrip.run()


    #Inhomogeneity Correction(Smoothing)
    output_smooth_file = "smooth_"+patient+"_T2.nii.gz"
    smooth = fsl.IsotropicSmooth(in_file=file_path+'/Skull_Stripped/'+output_strip_file,
out_file=file_path+'/Smooth/'+output_smooth_file,fwhm=4)
    smooth.run()
```

8.2 functions.py

```
import os # for doing directory operations
import pandas as pd
import nibabel as nib
```

```
import numpy as np
import cv2
import math
#path of folders
#file_path='/home/arpit/Desktop/Project'
file_path = os.getcwd()
#folder name
data_dir = file_path+'/Smooth/'
#listing of all folder in data_dir
patients = os.listdir(data_dir)
#used in reshape to decrease fix size of slices eg:60->20 by chunk_size of 3
def chunks(l, n):
    """Yield successive n-sized chunks from l."""
    for i in range(0, len(l), n):
        yield l[i:i + n]

#used to merge slices by taking mean of them
def mean(l):
    return sum(l) / len(l)

def preprocess_data(patient,labels_df,img_px_size=50, hm_slices=20, visualize=False):
    #patient of type smooth_LGG-343_T2.nii.gz
    path = data_dir + patient
    #print(path)
     #convert smooth_LGG-343_T2.nii.gz to LGG-343
    s_patient=str(patient)
    split_patient=s_patient.split('_')
    #read label of LGG-343
    label = labels_df.get_value(split_patient[1], 'Grade')

    #load nifti file as numpy array
    img = nib.load(path)
```

```
    img_data =img.get_data()
    shape=img_data.shape
    #resize slices from 256*256 to 64*64
    slices = [cv2.resize(img_data[:,:,i],(img_px_size,img_px_size)) for i in range(0,shape[2])]

    #make slice no. = 20
    new_slices = []
    chunk_sizes = math.ceil(len(slices) / hm_slices)
    for slice_chunk in chunks(slices, chunk_sizes):
        slice_chunk = list(map(mean, zip(*slice_chunk)))
        new_slices.append(slice_chunk)
    #print(len(slices),len(new_slices))
    empty_slice=list(np.zeros([img_px_size,img_px_size]))

    for i in range(0,hm_slices-len(new_slices)):
        new_slices.append(empty_slice)

print(len(slices),len(new_slices))
     return new_slices,label
```

8.3 cnn.py

```
from keras.models import Sequential
from keras.layers import Conv3D
from keras.layers import MaxPooling3D
from keras.layers import Flatten
from keras.layers import Dense,Reshape,Dropout
from sklearn.model_selection import train_test_split
import os
import numpy as np
# Initialising the CNN
classifier = Sequential()
#classifier.add(Reshape((64,64,20,1), input_shape=(1,20,64,64)))
```

```
#classifier.add(Reshape((20,64,64,1), input_shape=(1,20,64,64)))

# Step 1 - Convolution
classifier.add(Conv3D(32, (3, 3, 3), input_shape = (64,64,20,1), padding='SAME'))

# Step 2 - Pooling
classifier.add(MaxPooling3D(pool_size = (2,2,2),padding='SAME'))
# Adding a second convolutional layer
classifier.add(Conv3D(64, (3, 3, 3), padding='SAME'))
classifier.add(MaxPooling3D(pool_size = (2, 2, 2),padding='SAME'))

# Step 3 - Flattening
classifier.add(Flatten())

# Step 4 - Full connection
classifier.add(Dense(units = 512, activation = 'relu'))
classifier.add(Dropout(0.25))
classifier.add(Dense(units = 32, activation = 'relu'))
classifier.add(Dropout(0.5))
classifier.add(Dense(units = 1, activation = 'sigmoid'))

# Compiling the CNN
from keras import optimizers
adam = optimizers.Adam(lr=0.0001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0,
amsgrad=False)
classifier.compile(optimizer = adam, loss = 'binary_crossentropy', metrics = ['accuracy'])
classifier.summary()

# Part 2 - Fitting the CNN to the images
final_img_path=os.getcwd()+"/final_img-64-64-20.npy"
final_img=np.load(final_img_path)
```

```
final_img=final_img.transpose(0,3,4,2,1)
final_labels_path=os.getcwd()+"/final_labels.npy"
final_labels=np.load(final_labels_path)


(X_train,X_test,y_train,y_test)=train_test_split(final_img,final_labels,test_size=0.2,random_
state=42)
#classifier.fit(X_train,Y_train,batch_size=30, epochs=1)
classifier.fit(X_train, y_train,
        batch_size=2,
        epochs=5,
        verbose=1,
        validation_data=(X_test, y_test),
        shuffle=True)


print(X_test)
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)

print( y_pred[0])
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

from keras.models import load_model
classifier.save('brain_model.h5')
```

8.4 final.py

```
import os # for doing directory operations
import pandas as pd
import numpy as np
from functions import *
```

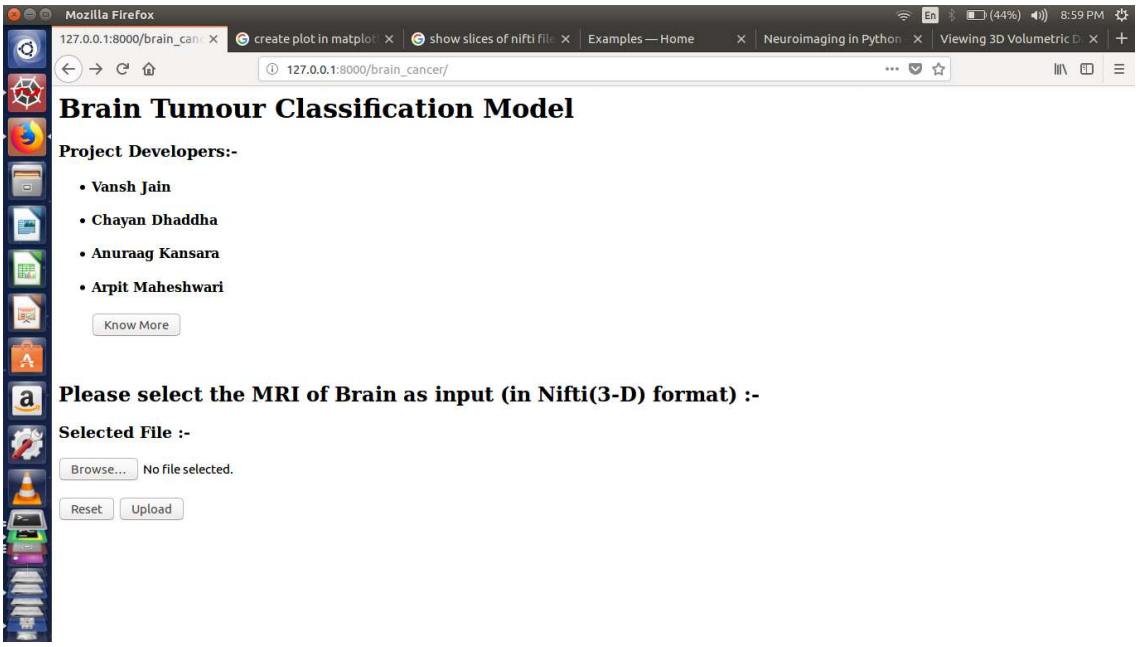
```
#from nipype.interfaces import fsl
#import matplotlib.pyplot as plt
#used to reshape and resize
IMG_PX_SIZE = 64
NUM_SLICES = 20

file_path = os.getcwd()
#folder name
data_dir = file_path+'/Smooth/'
#listing of all folder in data_dir
patients = os.listdir(data_dir)
   #get labels from xlsx file
labels_df=pd.ExcelFile(file_path+"/TCIA_LGG_cases_159.xlsx")
labels_df=labels_df.parse("Sheet1",index_col=0)
labels_df=labels_df.drop(['1p/19q','Type'],axis=1)
#print(labels_df)
final_img=[]
final_labels=[]
for patient in patients[:]:
   print("hiii ",patient)
   img_data,label=preprocess_data(patient,labels_df,img_px_size=IMG_PX_SIZE,
hm_slices=NUM_SLICES)
   #print(img_data,label)
   """if label==2:
      file_out = file_path + "/final_data/benign/"
   elif label==3:
      file_out = file_path + "/final_data/malignant/" """
   final_img.append([img_data])
   final_labels.append(label-2)
   print(np.array(img_data).shape)
np.save(file_path+'/final_img-{}-{}-
{}.npy'.format(IMG_PX_SIZE,IMG_PX_SIZE,NUM_SLICES),final_img)np.save(file_path
+'/final_labels.npy',final_labels)   print(np.array(final_img).shape)
```

9.0 Screenshots

```
_________________________________________________________________
Train on 138 samples, validate on 32 samples
Epoch 1/10
138/138 [==============================] - 316s 2s/step - loss: 0.9881 - acc: 0.5652 - val_loss: 0.7063 - val_acc: 0.6500
Epoch 2/10
138/138 [==============================] - 202s 1s/step - loss: 0.7051 - acc: 0.6087 - val_loss: 0.6633 - val_acc: 0.6500
Epoch 3/10
138/138 [==============================] - 210s 2s/step - loss: 0.6970 - acc: 0.6232 - val_loss: 0.6792 - val_acc: 0.6500
Epoch 4/10
138/138 [==============================] - 225s 2s/step - loss: 0.6460 - acc: 0.6739 - val_loss: 0.6749 - val_acc: 0.6500
Epoch 5/10
138/138 [==============================] - 200s 1s/step - loss: 0.6367 - acc: 0.6159 - val_loss: 0.6876 - val_acc: 0.6500
Epoch 6/10
138/138 [==============================] - 200s 1s/step - loss: 0.6343 - acc: 0.6594 - val_loss: 0.6840 - val_acc: 0.6500
Epoch 7/10
138/138 [==============================] - 202s 1s/step - loss: 0.5834 - acc: 0.6812 - val_loss: 0.7404 - val_acc: 0.6500
Epoch 8/10
138/138 [==============================] - 199s 1s/step - loss: 0.5776 - acc: 0.6957 - val_loss: 0.7408 - val_acc: 0.6800
Epoch 9/10
138/138 [==============================] - 199s 1s/step - loss: 0.4883 - acc: 0.7754 - val_loss: 1.0127 - val_acc: 0.6700
Epoch 10/10
138/138 [==============================] - 199s 1s/step - loss: 0.4940 - acc: 0.7391 - val_loss: 0.6538 - val_acc: 0.6800
```

Layer (type)	Output Shape	Param #
conv3d_8 (Conv3D)	(None, 64, 64, 20, 32)	896
max_pooling3d_9 (MaxPooling3	(None, 32, 32, 10, 32)	0
conv3d_9 (Conv3D)	(None, 32, 32, 10, 64)	55360
max_pooling3d_10 (MaxPooling	(None, 16, 16, 5, 64)	0
flatten_5 (Flatten)	(None, 81920)	0
dense_13 (Dense)	(None, 512)	41943552
dropout_9 (Dropout)	(None, 512)	0
dense_14 (Dense)	(None, 32)	16416
dropout_10 (Dropout)	(None, 32)	0
dense_15 (Dense)	(None, 1)	33

Total params: 42,016,257
Trainable params: 42,016,257
Non-trainable params: 0


Mozilla Firefox
8:59 PM
create plot in matplot
show slices of nifti file
Examples — Home
Neuroimaging in Python
Viewing 3D Volumetric D
127.0.0.1:8000/brain_cancer/
Brain Tumour Classification Model
Project Developers:-
Vansh Jain
Chayan Dhaddha
Anuraag Kansara
Arpit Maheshwari
Know More
Please select the MRI of Brain as input (in Nifti(3-D) format) :-
Selected File :-
Browse...
No file selected.
Reset
Upload

10. Conclusion

In conclusion, it is clear now that the methods used can improve the classification of different grades of Glioma in patients. These methods can be integrated in one methodology that is capable of significantly improve the classification problem at hand.

Using a balancing training data prevents bias during classifier training. Feature ranking has not a direct effect on improving the classification task, though it can reduce the search time in the feature selection step. Feature selection hits two birds with one stone. First, it reduces the future training process by selecting an optimal subset of features. Second, it improves the accuracy by eliminating irrelevant and redundant features.

This project uses some specific methods of image pre-processing on Brain MRI scans. The processed images are fed into a Convolutional Neural Network which extracts features from them. These features are used as an input to an Artificial Neural Network which classifies the grade of disease.

11. References

- Deep Learning A-Z course on Udemy.com
- Coursera Deep Learning Specialization by deeplearning.ai
- Cancer Imaging Archive, www.cancerimagingarchivearchive.net
- Magnetic Resonance Contrast Prediction Using Deep Learning, Cagan Alkan, John Cocjin, Andrew Weitz.
- Neurohacking in R course on Coursera.com
- Kaggle, pythonprogramming.net

ACKNOWLEDGEMENT

The satisfaction that accompanies with the successful completion of this seminar would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success. I am grateful to my seminar guide **Mr. Alok Singh Gahlot** for the guidance, inspiration and constructive suggestions that was helpful in the preparation of this study.

Last but not the least I wish to avail myself of this opportunity to express a sense of gratitude and love to my friends and my beloved parents for their strength and support.

I have tried my level best to make this seminar report error free, but I regret for errors, if any.