**MRI Brain Tumour Classifier Using CNN – Project Report**

**Problem Statement –** In this project I am trying to solve a problem related to the medical field , It is a multiclass classification problem, where I have pictures of MRI (Magnetic Resonance Imaging) of human brain , the pictures are from different angles and I need to make an AI model which will take the MRI picture as input and will classify it.

**About the data set-**

We have the following 4 categories in our data set

* Glioma
* Meningioma
* Pituitary
* No Tumor

The first three categories are types of brain tumors and the 4th category is No Tumor i.e the MRI reflects no brain tumor

Let us have a look at each type of brain tumor

**Glioma** - Gliomas are a type of tumor that arises from **glial cells**, which are the supportive cells in the nervous system. Gliomas include several subtypes based on the specific type of glial cell involved, such as astrocytomas, oligodendrogliomas, and ependymomas.

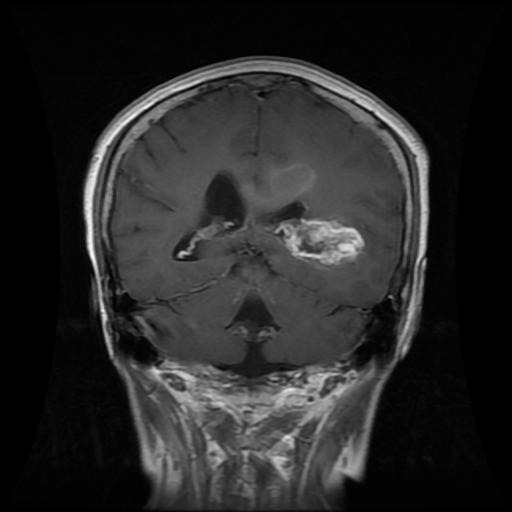
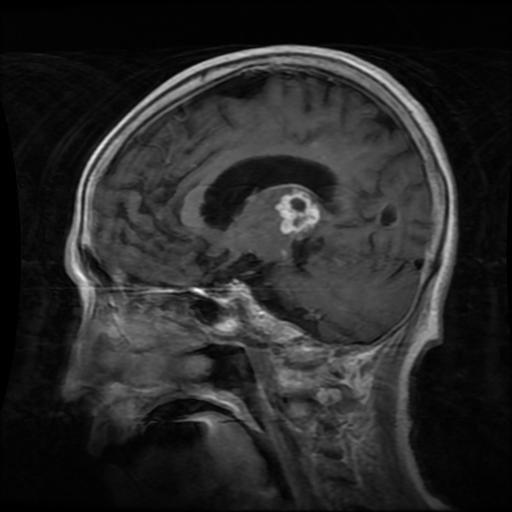
**Location**

Gliomas can occur in various parts of the brain and spinal cord since glial cells are distributed throughout the central nervous system. They are commonly found in the cerebral hemispheres (the main part of the brain), but they can also develop in the brainstem, cerebellum, and spinal cord.

**Shape and appearance**

Gliomas often have irregular and infiltrative shapes. They tend to grow by infiltrating surrounding brain tissue, making their boundaries less distinct compared to other tumour types.

They can appear as diffuse masses with varying degrees of contrast enhancement on imaging studies. Higher-grade gliomas, like glioblastomas, may show necrotic (dead) areas within the tumour, leading to a heterogeneous appearance

** **

**Meningioma- These** are tumours that arise from the **meninges**, the protective membranes covering the brain and spinal cord. They are the most common type of primary brain tumour, accounting for about 30% of all brain tumours. Most meningiomas are benign (non-cancerous), but some can be atypical or malignant (cancerous).

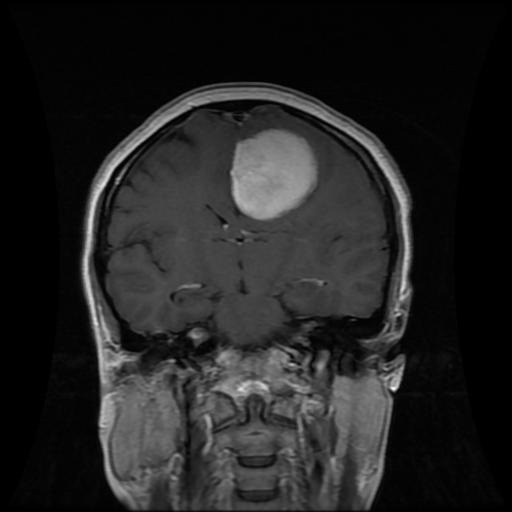
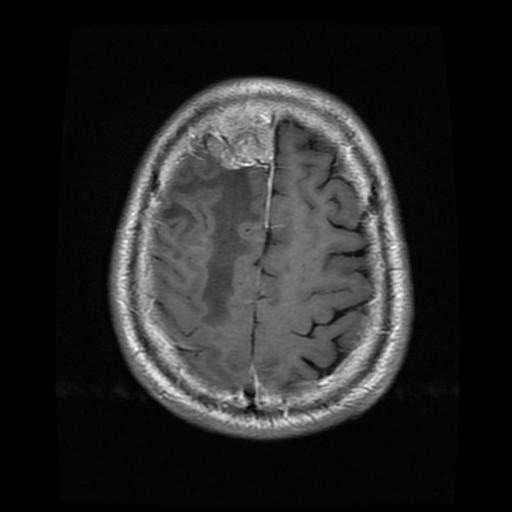
**Intracranial Locations**: Meningiomas can develop anywhere along the meninges within the cranial cavity. Common sites include:

* **Convexity Meningiomas**: On the outer surface of the brain.
* **Parasagittal Meningiomas**: Along the top part of the brain near the midline.
* **Sphenoid Wing Meningiomas**: At the base of the skull near the sphenoid bone.
* **Olfactory Groove Meningiomas**: Near the olfactory nerves.
* **Posterior Fossa Meningiomas**: In the back of the skull near the cerebellum.

**Spinal Locations**: Meningiomas can also arise from the meninges surrounding the spinal cord.

**Shape and appearance**

Meningiomas are typically well-defined, round, or lobulated (multilobed). They grow outward from the meninges, creating a clear and often smooth contour against the brain or spinal cord.

**Pituitary**- **Pituitary adenomas** are benign (non-cancerous) tumours that arise from the pituitary gland, which is a small gland located at the base of the brain. The pituitary gland is often referred to as the "master gland" because it regulates various hormones that control important body functions. Pituitary adenomas are relatively common, accounting for about 10-15% of all intracranial tumours.

**Location**

Pituitary adenomas are located in the **sella turcica**, a bony structure at the base of the brain. This small cavity houses the pituitary gland

**Shape and appearance**

Pituitary adenomas are usually well-circumscribed and have a rounded or oval shape. They tend to expand within the confines of the sella turcica.

**Data set Details**

The data set was in two parts Training data and Testing data each part was having 4 categories which I have previously mentioned .

In training data each of the four categories were having about 1500 samples , with slight data imbalance , at max of 200 images.

In testing data each of the four categories were having about 400 samples , minimal data imbalance.

The above images in the report have been picked up from the original data set itself

**Data Augmentation**

The need of data augmentation –

* To increase the number of samples in the training data set, deep learning models are data hungry
* To mitigate the problem of overfitting, so that the model doesn’t pick up some trend in the data set which isn’t actually a trend
* To solve the problem of imbalance in the training data set, which would have caused the problem of bias

So I applied data augmentation technique and post augmentation I was having about 2.5k images in each category, i.e a total of 10k images in the training data set.

**ImageDataGenerator**

*train\_datagen=ImageDataGenerator(rotation\_range=5,*

*rescale=1/255,*

*shear\_range=0.1,*

*horizontal\_flip=False,*

*vertical\_flip=False,*

*width\_shift\_range=0.1,*

*height\_shift\_range=0.1,*

*fill\_mode='nearest')*

This is the image data generator for training data, do notice the fact that the **parameter settings are very mild**, for example the

*rotation\_range=5*, i.e the image will be rotated within the range +- 5 degree

*rescale=1/255*, the images were RGB so the entries in the pixels varied between 0 to 255 , this basically

scales(normalizes) the the pixels of the images generated by the image data generator , so the

generated images will have pixel values between 0 to 1, where (0,0,0) represents black and (1,1,1)

represents white

*shear\_range=0.1* , Applies a slant or skew effect, distorting the image's angles and proportions.

*fill\_mode='nearest'* , Fills the remaining pixels with the nearby pixel values

**The reason that all the parameters are set to a very minimum value is because if you say generate flipped images and extremely shifted , rotated sheared images in case of MRI images , note that MRIs are taken under the guidance of a professional , and they are taken in a certain monitored position, that is the images in the MRIs are not very tilted ,** **shifted , flipped, etc , they are very fixed ,because of the fixed position of the head in the MRI scanner , yeah obviously shape of the head , a little bit of zooming variation and mild tilt depending on the patient is there in the actual MRI images , and if you generated images with extreme tilts and angles you will just confuse your model during training because it will start observing non-essential trends and will not perform at its best. Thus the data augmentation is very mild**

*test\_datagen=ImageDataGenerator(rescale=1/255)*  This is my test Image Data Generator , so I am not applying any thing in here other than scaling the test images that too because I will be training my model on scaled images for better performance and convergence speed

**Applying Augmentation using flow\_from\_directory**

*# Base directory for saving augmented images*

*save\_base\_dir=r'C:\IIT B\Brain Tumor CNN\Augmented Data\Training'*

*glioma\_generator=train\_datagen.flow\_from\_directory(r'C:\IIT B\Brain Tumor CNN\Data\Training',*

*target\_size=(256,256), # we are resizing the image as well*

*batch\_size=1, # from each image 1 new image will be generated*

*class\_mode='categorical',*

*classes=['glioma'],*

*save\_to\_dir=os.path.join(save\_base\_dir,'glioma'),# save the augmented images to the 'glioma' subdir*

*save\_prefix='glioma') # the image genrated will have the prefix 'glioma '*

*# Iterate through the generator*

*for i in range(0,2500): # I want 2.5k images in glioma*

*glioma\_images,glioma\_labels=next(glioma\_generator)*

In this I have specified the directory where I will be saving the augmented images , this code is specifically for ‘glioma’ class of the training data set , you will have to write code for each category of the training data set

Target\_size is the size of the images which will be generated , I chose this specific size because I wanted my images to be of good size so that I can extract features well , batch size combined with the number of times which you are running the loop will determine the total number of images you will have in this category post augmentation ,

So note that before running this you will have to make a folder ‘glioma’ inside the *r'C:\IIT B\Brain Tumor CNN\Data\Training'* location and then you will save the augmented images inside the directory ‘glioma’ using save\_to\_dir

**The following is the code for ‘glioma’ class of testing data, I didn’t generate extra images I just rescaled them**

*save\_base\_dir=r'C:\IIT B\Brain Tumor CNN\Augmented Data\Testing'*

*glioma\_generator=test\_datagen.flow\_from\_directory(r'C:\IIT B\Brain Tumor CNN\Data\Testing',*

*target\_size=(256,256), # we are resizing the image as well*

*batch\_size=1, # from each image 1 new image will be generated*

*class\_mode='categorical',*

*classes=['glioma'],*

*save\_to\_dir=os.path.join(save\_base\_dir,'glioma'),# save the augmented images to the 'glioma' subdir*

*save\_prefix='glioma') # the image genrated will have the prefix 'glioma '*

*# since we are not adding additional pictures in the training data we will run the loop = number of images of this category in the training data set*

*for i in range(0,300):*

*glioma\_images,glioma\_labels=next(glioma\_generator)*

**By the end of all this I had new Training and Testing data , all the four categories in the Training data were having about 2k images each , each image was of shape (250,250,3) with slight modifications which we set in ImageDataGenerator class**

**I used this data to train my model**

**LOADING DATA FOR TRAINING THE MODEL**

# let us get the training data set

train\_ds=keras.utils.image\_dataset\_from\_directory(directory="C:\IIT B\Brain Tumor CNN\Augmented Data\Training",

labels='inferred' ,

label\_mode='categorical' ,

batch\_size=32,

image\_size=(256,256)

)

**It is pretty clear that I have used keras.utils.image\_dataset\_from\_directory to inflow the images from the linked directory on my device , *labels=’inferred’* means it will pick up the folder name as labels , *label\_mode=’categorical’* ,now since I am picking up label names from the subfolder of the training data set and they are ‘glioma’, ’meningioma’ etc i.e non numerical so *label\_mode=’categorical’* and the names of the labels will be one hot encoded example ‘glioma’=[1 0 0 0] , ’meningioma’ =[0 1 0 0] etc , we also have *label\_mode=’int’,* would have used this if my labelling of categories in the training data would have been numerical , and in this case I would have used ‘sparse\_categorical\_crossentropy’ as my loss function,**

**Also the batch\_size= 64 tells the batch size in which the images will be forward propagated into the architecture or you can even say that it sets the flow of images from the directory , it is the same batch size which is used for deciding the nature of Gradient Descent algo**

**You will have to make a image\_dataset\_from\_directory for the testing data set also !!**

**And by the end of this process you will have train\_ds and validation\_ds, which you will feed to your architecture**

**THREE IMPORTANT THINGS TILL NOW**

1-ImageDataGenerator

2-flow\_from\_directory

3-image\_dataset\_from\_directory

**CNN ARCHITECTURE**

*model=Sequential()*

*model.add(Conv2D(64,kernel\_size=(5,5),activation='relu',input\_shape=(256,256,3)))*

*model.add(MaxPooling2D(pool\_size=(3,3)))*

*model.add(Conv2D(128,kernel\_size=(5,5),activation='relu'))*

*model.add(MaxPooling2D(pool\_size=(3,3)))*

*model.add(Conv2D(128,kernel\_size=(4,4),activation='relu'))*

*model.add(MaxPooling2D(pool\_size=(3,3)))*

*model.add(Conv2D(256,kernel\_size=(4,4),activation='relu'))*

*model.add(MaxPooling2D(pool\_size=(3,3)))*

*model.add(Flatten())*

*model.add(Dense(512,activation='relu'))*

*model.add(Dropout(0.3))*

*model.add(Dense(256,activation='relu'))*

*model.add(Dropout(0.3))*

*model.add(Dense(128,activation='relu'))*

*model.add(Dropout(0.3))*

*model.add(Dense(64,activation='relu'))*

*model.add(Dropout(0.3))*

*model.add(Dense(4,activation='softmax'))*

*model.summary()*

This is a 9 layered CNN architecture with 4 CNN layers and 5 Dense layers and you can clearly see the number of filters and the number of nodes in the architecture it has about a total of 1M+ trainable parameters .

Reached this specific architecture after a lot of tuning , tried BatchNormalisation as well , surprisingly model was not performing well with batch normalization , tried adding more layers and a lot of things , tried weight initialisation as well (went for HeUniform()) which is a switch from the default GloratUniform() etc . Also tried to take inspiration from architecture of similar problems available online , figured out that 2 attached cone shaped architecture works really well and then adopted the 2 attached cone shaped architecture and fine tuned it to get the maximum accuracy on the validation data set .

**# Early Stopping**

*call= EarlyStopping(monitor='val\_accuracy',*

*restore\_best\_weights=True,*

*patience=25,*

*min\_delta=0.001,*

*verbose=1,*

*mode='auto'*

*)*

Implemented EarlyStopping mechanism from keras.callbacks to prevent overfitting and to restore best weights from the training epochs , also tuned the early stopping class for the best implementation.

*model.compile(loss='categorical\_crossentropy',optimizer='adam',metrics=['accuracy'])*

*history=model.fit(train\_ds,epochs=60,validation\_data=validation\_ds,verbose=1,callbacks=call)*

This is my further implementation !!

*Restoring model weights from the end of the best epoch: 57.*

This is what EarlyStoppings’ restore\_best\_weights=True parameter did

**OUTPUT**

Predicted output , do note the fact that for every image you will get four probabilistic outputs because in the output layer there are 4 nodes with softmax activation function , and the final output will be the one with the highest probability

In my training\testing data , the ordering of the files was the following –

1-Glioma

2-Meningioma

3-No tumor

4-Pituitary

So the output given by the first output node will be the probability of the sample belonging to Glioma

and the output given by the second output node will be the probability of the category Meningioma etc.

Thus for each sample the output will be a 1 D tensor – 4D vector then using the argmax of numpy you will extract the index of the biggest value in that array 0,1,2,3 .

So you will have a new array where you will have predictions for the whole testing data set and that array will be of the form

[[0,1 ,2 3 ,0…]] where 0 is the predicted category of the first sample and so on

NOW to verify you will have to extract the actual category of the testing data set and then since we have OHEd the category names , we will have to use argmax again to convert the OHEd category names into numerical labelling

Then from it you can plot the confusion matrix and also calculate various metrics like Accuracy ,F1, weighted F1 and precision/ recall for each category and overall precision and recall .

**Special Point**

Do note the fact that we are working on a problem related to the medical field , and in my case a specific type of error carried more significance , i.e error where my model classifies patients with tumour as ‘no tumour’, this is very fatal , thus you need to check this error as well , if my model classifies glioma as meningioma it is also not good but not equally fatal as the previous case!

Thus I calculated precision of the ‘no tumour’ category as well =

Precision Of No Tumour = (True Positive)/(True Positive + False Positive)

You can calculate this using confusion matrix , False Positive will be the samples which are classified as ‘no tumour ’ by the model but are actually tumour samples