BRAIN TUMOUR DETECTION

Done By:
MAYANK DASH (19BCE1663),
PRAKHAR RATHORE (19BCE1793)

ABSTRACT

Brain Tumor is one the most deadly disease whose detection and treatment is one of the toughest tasks. At present, treatment of patients requires clinical pictures examination and is becoming a critical field. It fuses a wide scope of imaging procedures some of them are Computed Tomography channels (CT examines), Magnetic Resonance Imaging (MRIs) and X-columns and so forth. Medical experts perform tumour segmentation on data obtained from magnetic resonance imaging (MRI) which is very time consuming. Brain tumour segmentation is a significant process to extract information from complex MRI of brain image. Segmentation assessment is done by human, which can involve human errors in the result. So we have choose this topic because it is one of the most difficult and trending arena whose precision is the most important thing than any think else. So we implemented the detection using 2 ways the first one is the openCV method done using python. It takes MRI image as input and performs basic pre-processing. We have also implemented another process using or a way without using any big standard library and using convolution and integrated with machine leaning this predicts, the presence of the tumour region in the brain. This 2 method help us to detect the basic region where the tumour is present and highlights the region where the tumour is present. So this in a good way and highly précised way for the detection of the tumour in the MRI images of the brain. Doing it technically reduces the chance of error in a very great manner.

INTRODUCTION AND MOTIVATION

In today’s scenario, brain tumour is dangerous and harmful to the human being that leads to death. As the time progresses brain tumour’s complexity increases and if not taken care during early stages the person might die. As a result, early detection of tumours is most crucial to save and prolong the patient’s lifetime. For this reason an enhanced and early scans and identification is great deal in this case. MRI or Magnetic resonance imaging is one of the most common technique used to detect any abnormality in brain is present or not. Before Magnetic resonance images were analysed by human (radiologist) to detect abnormalities in the brain. The human way of detecting the presence of tumour or any abnormalities is highly difficult and time consuming and if anything is misread by the human and calculated wrong due to some noise in the images or extra blurriness this might lead to wrong readings and might lead to the opposite outcome which is highly dangerous. For this reasons and many other simple reasons which might lead to greater wrong we proposed 2 approaches were in the first one we use openCV using which we convert or pre-process our image using various technique such as thresholding, morphological transformation, Gaussian blurring, histogram equalization and various other process to find the particular region. In the second one we have integrated the code with particular ai model so that we can see the result of presence of tumour or not using this model.

BRAIN TUMOUR DETECTION

Done By:
MAYANK DASH (19BCE1663),
PRAKHAR RATHORE (19BCE1793)

LITERATURE SURVEY:

Analyzing and processing of MRI brain tumor images are the most challenging and upcoming field. Magnetic resonance imaging (MRI) is an advanced medical imaging technique used to produce high-quality images of the parts contained in the human body and it is very important process for deciding the correct therapy. Many techniques have been proposed for classification of brain tumors in MR images such as fuzzy clustering means (FCM), support vector machine (SVM), artificial neural network (ANN), knowledge-based techniques, and expectation-maximization (EM) algorithm technique which are some of the popular techniques used for region-based segmentation and so to extract the important information from the medical imaging modalities.

BWT and SVM techniques image analysis for MRI-based brain tumor detection and classification. In this method, accuracy of 95% was achieved using skull stripping which eliminated all non-brain tissues for the detection purpose. The proposed segmentation of MRI brain images using K-means clustering algorithm along with morphological filtering for the detection of tumor images. The automated brain tumor classification of MRI images using support vector machine was proposed by Alfonse and Salem. The accuracy of a classifier was improved using fast Fourier transform for the extraction of features and minimal redundancy maximal relevance technique was used for reduction of features. The accuracy obtained from this proposed work was 98%.

The brain MRI image contains two regions which are to be separated for the extraction of brain tumor regions. One part of region contains the tumor abnormal cells, whereas the second region contains the normal brain cells. So we can see that there will be a change in the intensity in the parts of the image and edges can be detected in it and regions separated and tumor can be detected.

To manage and to address protocols of different images and nonlinearity of real data an effective classification based on contrast of enhanced MRI images, proposed an methodology which included extraction of textures features with wavelet transform and SVM with an accuracy of 83%. For the classification and brain tumor segmentation, they obtained an accuracy of 94% with this method. So we have used neural networks and the other openCV process to detect the tumor.

PROPOSED WORK

We have created 2 models the first one using the openCV model integrated using python and the second model is the one which is made using the machine learning model. Both have their own way of execution and references.

The First Model using OpenCV:

We have implemented it using Pycharm. First we need to import cv2 and import images from the dataset and store the images in a directory of a Pycharm. First we need to convert the images it into gray scale images whether it may be coloured or grey-white images all should to be subjected to the conversion. Then the image converted to grayscale image is subjected to thresholding and value of the thresholding pixel is taken as 155. In thresholding each pixel value is compared

BRAIN TUMOUR DETECTION

Done By:
MAYANK DASH (19BCE1663),
PRAKHAR RATHORE (19BCE1793)

with the threshold value. If pixel value is smaller value then it is set to 0, otherwise to maximum value. So we have taken 155 as the threshold value and 255 is maximum value that can be assigned. Then we apply inverse thresholding to the image that was subjected to gray scale changes and assign pixel value 155 but the min value is set as 0. This works exactly reverse of thresholding. Then we use morphological operations on the image obtained after thresholding so that we can remove unwanted part from the thresholding image. Morphological operations are simple operations based on image shape. They are normally performed on binary images. They need two inputs, one is the original image and the second one is called a structuring element or kernel which decides the nature of the operation. We have performed both erosion and dilation. Erosion erodes away the boundaries of foreground objects. A pixel in the image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1 otherwise, it is eroded. Therefore, the pixels near the boundary will be discarded depending upon the size of the kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image. Thus we have used this to remove the small white noises. So the image we obtain after the function of erosion, we use that image to dilate to dilate it and store it another variable. Then the image we obtain after the subjection of morphological operations on it we again subject it to canny edge detectors. This detector is carried out to find out the outline of the tumor, this function automatically calculates the lower and upper threshold values. The image we obtain after subjecting it to canny detectors then image is then subjected to Contours. Contours are defined as the line joining all the points along the boundary of an image that are having the same intensity. After performing all this process we get an output image highlighted with region if a tumor is present otherwise no highlight we only get a gray scale MRI of brain.

The Second Model using ML:

This second model is performed or done using in python in Google colab. First we take a dataset, we download it from Kaggle and store it inside our particular Google drive, and we connect our Google drive with colab and store the images inside a list. We import some basic libraries such as tensor flow, keras, sklearn(for graphs), etc. and store images inside a list. We create some list such as yes, no, etc. Where we store the images containing the tumor and then in the no list we store the images that do con contain the tumor this case is done just to train our model for the given dataset. Then we first work on the yes list and resize and reshape all the images and convert it into the gray scale images. Then again we resize and reshape and convert into gray scale the images that are present in the no list. Then the list are converted into numpy arrays. Then we take a particular image form the array and convert it into the matrix form for the representation of particular areas colour configurations. After doing all this we try to train and test the image data for the complete dataset. We build a model in this case by importing various keras layers such as Dense, Dropout, Conv2D, Maxpooling2d, etc. We have used the dropout neural network because dropout works randomly setting the outgoing edges of hidden units to 0 this also prevents a model from over fitting. Then we apply convolution, batch normalization,

BRAIN TUMOUR DETECTION

Done By:
MAYANK DASH (19BCE1663),
PRAKHAR RATHORE (19BCE1793)

maxpooling, flattening, densing, dropout, etc. Then after getting the summary for the complete dataset their parameter, trainable parameters, etc. We do model compilation using RMSprop optimizer this is a gradient based optimizer used in neural networks. We use it to optimize the results and find the values using the epoch training data taken to be 75 for the case of learning rate. After the result of all the epochs we draw a graph showcasing training vs. validation accuracy and another graph showcasing the train vs. validation loss. After all the testing ,optimising and finding learning rate and accuracy we do testing and add the address of a particular image stored in Google drive and obtain it's output as yes or no. So in this manner we implement the project using 2 ways.

EXPERIMENTAL SETUP:

Dataset:

The dataset has been taken form the kaggel.

Link: https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection

This dataset contains 2 folders in which there are MRI images of the brain. The first folder contains the images which have tumour in it and named as yes, then second folder contains images which have no tumor in it named as yes. For testing a image just select any image from any of the folders and check the results.

SOFTWARE:

i) 1st Model:

This model uses OpenCV integrated using python and images stored in the directory of the Pycharm and to run the python code we have used Pycharm.

ii) 2nd Model:

This model uses neural networks for the purpose of detection and the software it uses is the Google colab and Google Drive. Google Drive is just used for storing the images which is then it is imported into list of created in python in colab.

MACHINE LEARNING MODEL:

i) 1st Model:

This model uses openCV integrated with python it does not uses any machine learning model but uses feature extraction techniques, smoothing, blurring, morphological, edge detection and contours techniques and various other technique were also used to achieve the particular result were in the input image if there is a presence of tumor a red mark comes surrounding the region.

ii) 2nd Model:

This model uses dropout neural network as this technique prevents model from over fitting and it works randomly setting outing edges of hidden units to 0 and etc. We have also used RMSprop optimizer which is an gradient based optimizer used in neural networks and used in knowing the learning rate. We also use the keras layers as batch normalization, Cov2D, flatten, etc. This a model and layers help to achieve the resultant and required answers and needs.

BRAIN TUMOUR DETECTION

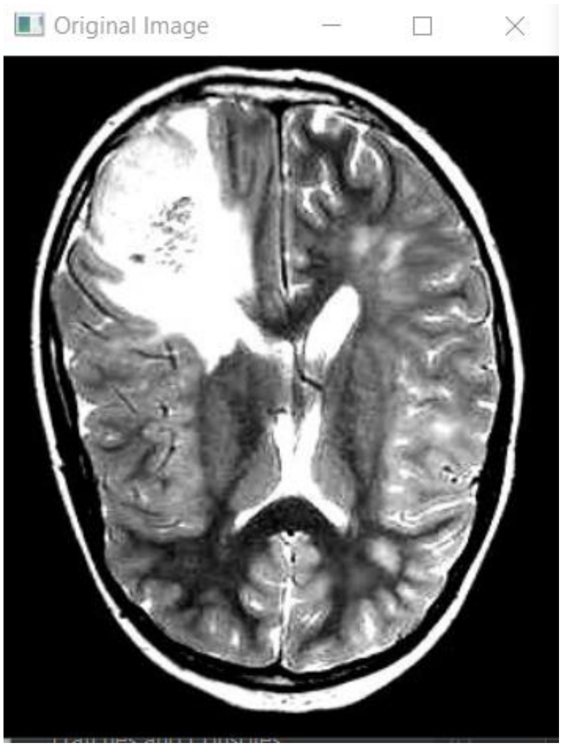
Done By:
MAYANK DASH (19BCE1663),
PRAKHAR RATHORE (19BCE1793)

SUMMARY:

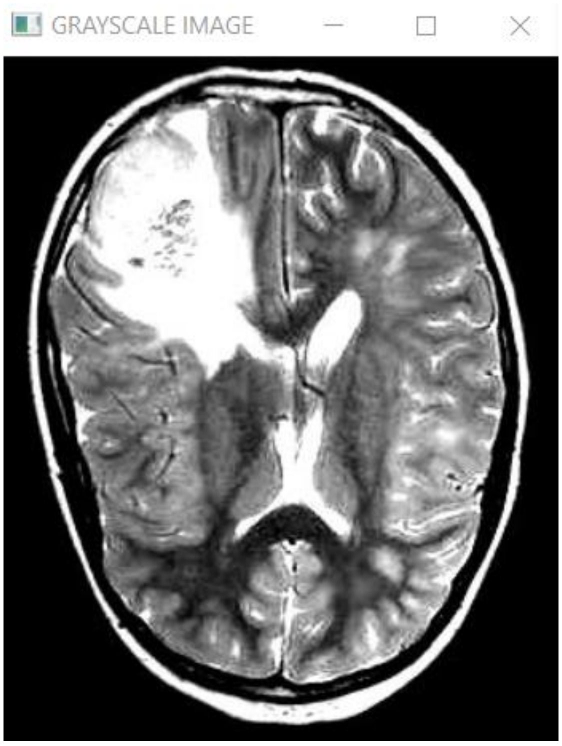
RESULTS OBTAINED:

i) 1st Model:

IMAGES: These all are the images after every intermediate steps and final steps.



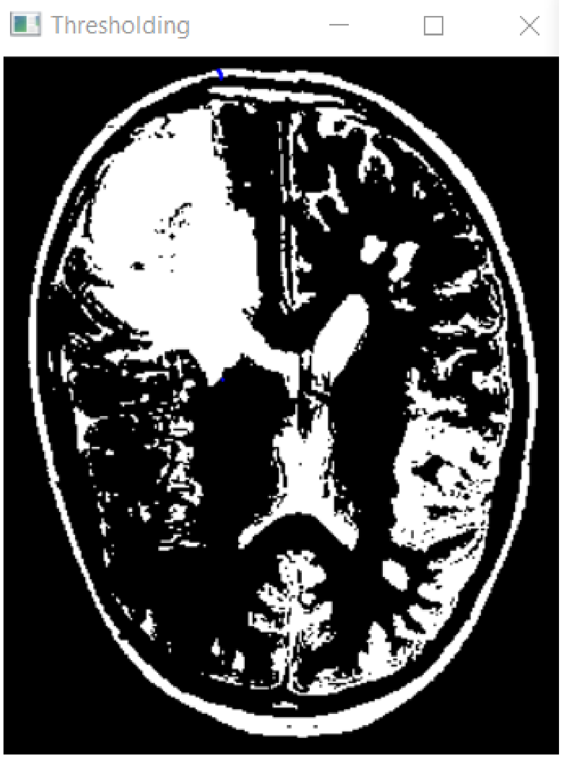
After importing from the Pycharm directory this is the original image on which we need to perform all the operations. This is an original MRI image from the dataset from the yes folder.



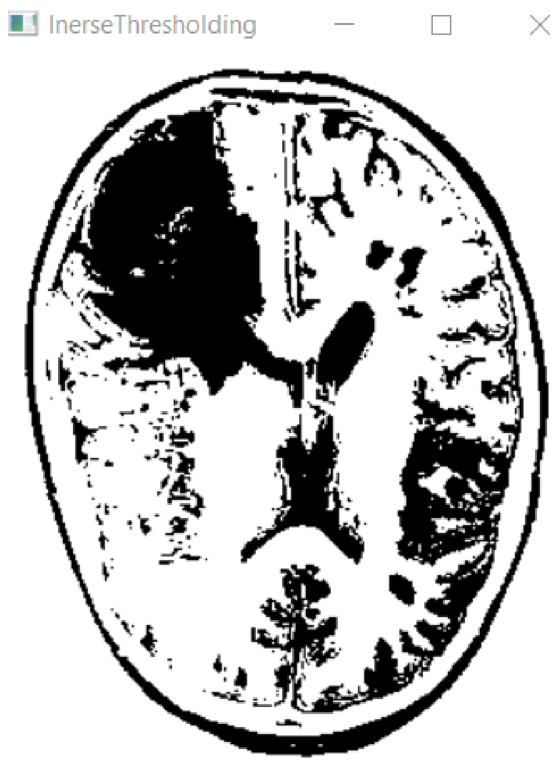
This is the conversion of any image or the original image into grayscale image.

BRAIN TUMOUR DETECTION

Done By:
MAYANK DASH (19BCE1663),
PRAKHAR RATHORE (19BCE1793)



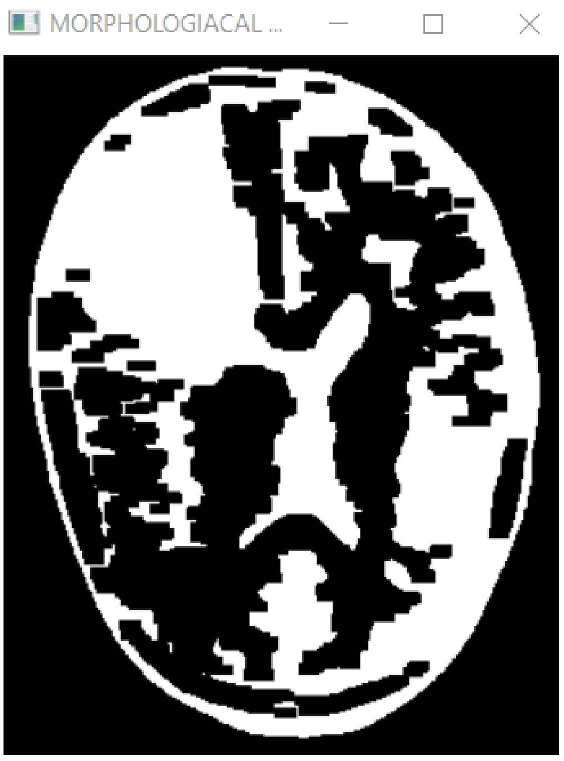
Here we have performed thresholding on the gray scale converted image taking pixel value to 155 where maximum value is taken to 255. If more than 155 value is taken as 255 or else 0.



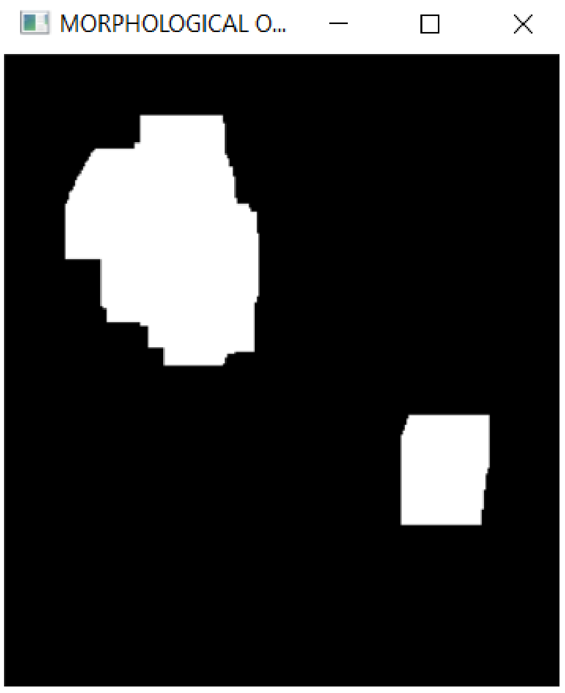
Next step after conversion of the image into thresholding we need to perform image into inverse thresholding where value of white and black is changed but pixel value is taken as 155 and black region appears and vice versa from the previous image. Its reverse of thresholding.

BRAIN TUMOUR DETECTION

Done By:
MAYANK DASH (19BCE1663),
PRAKHAR RATHORE (19BCE1793)



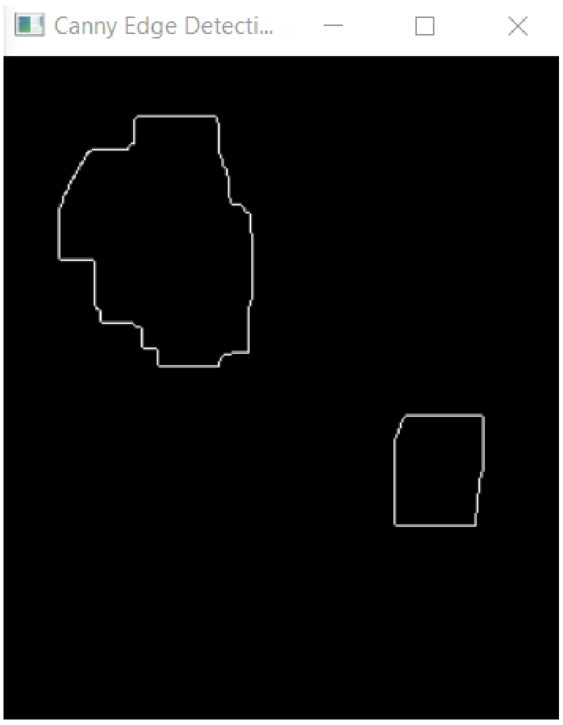
Here he have performed morphological operation on the image so the left out or unwanted parts in case of thersholding could be removed here.



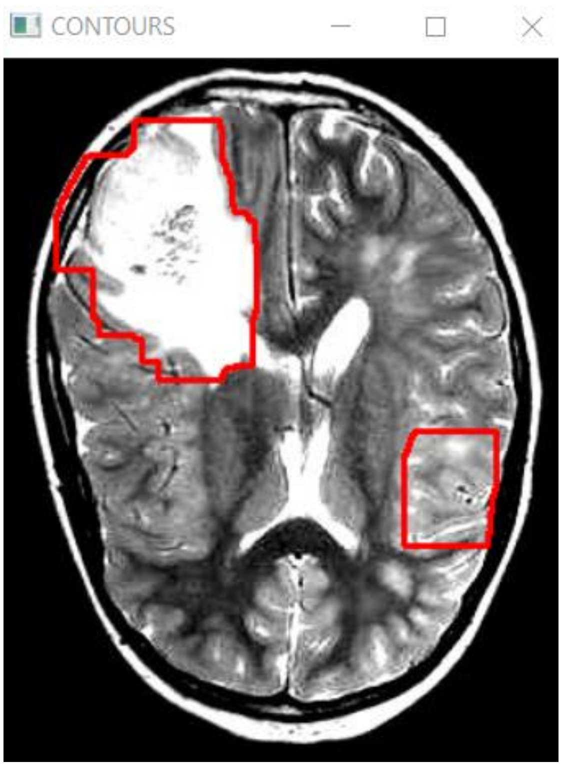
In this image morphological operations are performed after the previous step, this operations are erosion and dilation and we can see some part of brain on a backdrop complete white.

BRAIN TUMOUR DETECTION

Done By:
MAYANK DASH (19BCE1663),
PRAKHAR RATHORE (19BCE1793)



Here we can see the canny edge detectors performing and identifying the region with 2 different intensities it one of the edge detecting process. So we can see that 2 edges are highlighted which may represent the tumor.

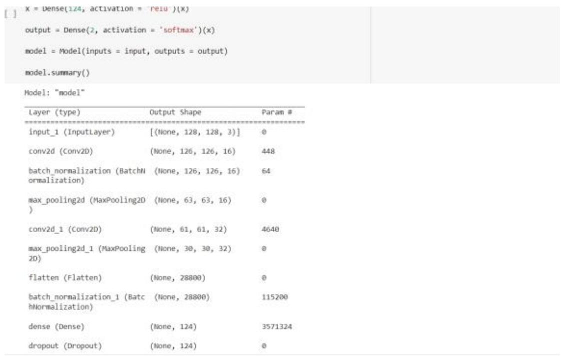


Here at end we use contours which highlights the edge from previous image and shows us the presence of tumor using a red highlighter which confirms presence of tumor. If there were no edge identified during edge canny detectors there will not have been any highlight or red mark showcasing tumor.

2ND MODEL:

BRAIN TUMOUR DETECTION

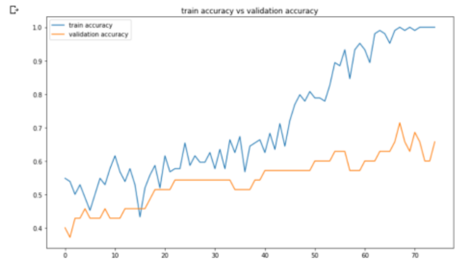
Done By:
MAYANK DASH (19BCE1663),
PRAKHAR RATHORE (19BCE1793)



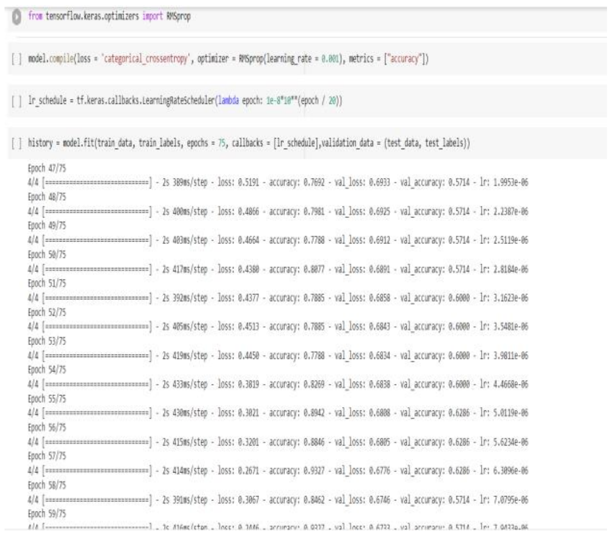
```
[ ] x = Dense(124, activation = 'relu')(x)

    output = Dense(2, activation = 'softmax')(x)

    model = Model(inputs = input, outputs = output)

    model.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 128, 128, 3)]     0

 conv2d (Conv2D)             (None, 126, 126, 16)      448

 batch_normalization (BatchN  (None, 126, 126, 16)     64
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 63, 63, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 61, 61, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 30, 30, 32)       0
 2D)

 flatten (Flatten)           (None, 28800)             0

 batch_normalization_1 (Batc  (None, 28800)            115200
 hNormalization)

 dense (Dense)               (None, 124)               3571324

 dropout (Dropout)           (None, 124)               0
```

After performing featurisation on the image using keras layers such as convolution, flattening, densing etc. we get this.

```
plt.figure(figsize = (12,7))

plt.plot(history.history['accuracy'], label = 'train accuracy')
plt.plot(history.history['val_accuracy'], label = 'validation accuracy')
plt.title("train accuracy vs validation accuracy")
plt.legend()
plt.show()
```



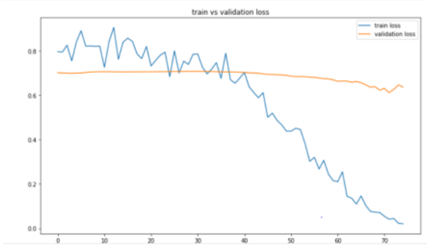
Train accuracy and validation accuracy of graphs for each epoch. It’s a gain graph for each epoch.



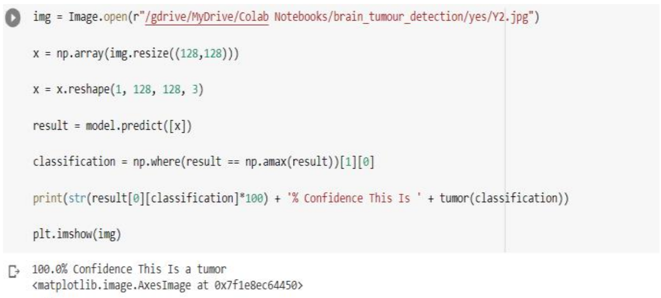
```
from tensorflow.keras.optimizers import RMSprop

[ ] model.compile(loss = 'categorical_crossentropy', optimizer = RMSprop(learning_rate = 0.001), metrics = ["accuracy"])

[ ] lr_schedule = tf.keras.callbacks.LearningRateScheduler(lambda epoch: 1e-8*10**(epoch / 20))

[ ] history = model.fit(train_data, train_labels, epochs = 75, callbacks = [lr_schedule],validation_data = (test_data, test_labels))

Epoch 47/75
4/4 [==============================] - 2s 389ms/step - loss: 0.5191 - accuracy: 0.7692 - val_loss: 0.6933 - val_accuracy: 0.5714 - lr: 1.9953e-06
Epoch 48/75
4/4 [==============================] - 2s 400ms/step - loss: 0.4866 - accuracy: 0.7981 - val_loss: 0.6925 - val_accuracy: 0.5714 - lr: 2.2387e-06
Epoch 49/75
4/4 [==============================] - 2s 403ms/step - loss: 0.4664 - accuracy: 0.7788 - val_loss: 0.6912 - val_accuracy: 0.5714 - lr: 2.5119e-06
Epoch 50/75
4/4 [==============================] - 2s 417ms/step - loss: 0.4380 - accuracy: 0.8077 - val_loss: 0.6891 - val_accuracy: 0.5714 - lr: 2.8184e-06
Epoch 51/75
4/4 [==============================] - 2s 392ms/step - loss: 0.4377 - accuracy: 0.7885 - val_loss: 0.6858 - val_accuracy: 0.6000 - lr: 3.1623e-06
Epoch 52/75
4/4 [==============================] - 2s 405ms/step - loss: 0.4513 - accuracy: 0.7885 - val_loss: 0.6843 - val_accuracy: 0.6000 - lr: 3.5481e-06
Epoch 53/75
4/4 [==============================] - 2s 419ms/step - loss: 0.4450 - accuracy: 0.7788 - val_loss: 0.6834 - val_accuracy: 0.6000 - lr: 3.9811e-06
Epoch 54/75
4/4 [==============================] - 2s 433ms/step - loss: 0.3819 - accuracy: 0.8269 - val_loss: 0.6838 - val_accuracy: 0.6000 - lr: 4.4668e-06
Epoch 55/75
4/4 [==============================] - 2s 430ms/step - loss: 0.3021 - accuracy: 0.8942 - val_loss: 0.6808 - val_accuracy: 0.6286 - lr: 5.0119e-06
Epoch 56/75
4/4 [==============================] - 2s 415ms/step - loss: 0.3201 - accuracy: 0.8846 - val_loss: 0.6805 - val_accuracy: 0.6286 - lr: 5.6234e-06
Epoch 57/75
4/4 [==============================] - 2s 414ms/step - loss: 0.2671 - accuracy: 0.9327 - val_loss: 0.6776 - val_accuracy: 0.6286 - lr: 6.3096e-06
Epoch 58/75
4/4 [==============================] - 2s 391ms/step - loss: 0.3067 - accuracy: 0.8462 - val_loss: 0.6746 - val_accuracy: 0.5714 - lr: 7.0795e-06
Epoch 59/75
4/4 [==============================] - 2s 416ms/step - loss: 0.3446 - accuracy: 0.9327 - val_loss: 0.6733 - val_accuracy: 0.5714 - lr: 7.9433e-06
```

After applying RMSprop model to the data which has been subjected to different features such as convolution, flattening, batch normalization etc. we are optimizing it and epochs are shown for 75 training data.

```
[ ] plt.figure(figsize = (12,7))

    plt.plot(history.history['loss'], label = 'train loss')
    plt.plot(history.history['val_loss'], label = 'validation loss')
    plt.legend()
    plt.title("train vs validation loss")
    plt.show()
```

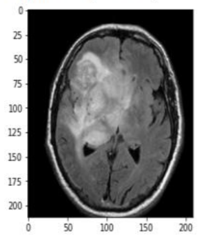


This graph represents a the figure for each train and valid loss for each epochs. It’s a loss graph for each epochs.



```
img = Image.open(r"/gdrive/MyDrive/Colab Notebooks/brain_tumour_detection/yes/Y2.jpg")

x = np.array(img.resize((128,128)))

x = x.reshape(1, 128, 128, 3)

result = model.predict([x])

classification = np.where(result == np.amax(result))[1][0]

print(str(result[0][classification]*100) + '% Confidence This Is ' + tumor(classification))

plt.imshow(img)
```

```
100.0% Confidence This Is a tumor
<matplotlib.image.AxesImage at 0x7f1e8ec64450>
```

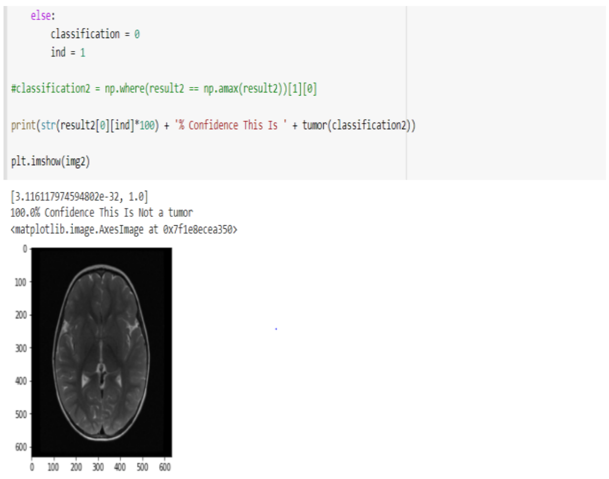


BRAIN TUMOUR DETECTION

Done By:
MAYANK DASH (19BCE1663),
PRAKHAR RATHORE (19BCE1793)

We have taken a image from yes folder in dataset and test the model or predict its accuracy we can see that it show 100% there is a tumor present in the MRI image. So this model predicts exactly correct for this image.

‘



```
    else:
        classification = 0
        ind = 1

#classification2 = np.where(result2 == np.amax(result2))[1][0]

print(str(result2[0][ind]*100) + '% Confidence This Is ' + tumor(classification2))

plt.imshow(img2)
```

```
[3.116117974594802e-32, 1.0]
100.0% Confidence This Is Not a tumor
<matplotlib.image.AxesImage at 0x7f1e8ecea350>
```

As we have taken a MRI image from No folder and try to predict its result, we can see that it predicts 100% there is no tumor in the MRI image, so we can see here this model clearly gives a correct result as we have tested for both yes and no.

CONCLUSION:

As you can see that we have implemented 2 models using openCV and ML model in both these model we can see that all the MRI images that we are taking for prediction or testing we are getting correct accurate results for mostly every image. This models are fast and remove human inclusion in it which makes chances of error very less and show an easy way for recognizing the tumor region as the first model directly highlights the region which has tumor and makes it easy for doctors to put in their views and operate on it and model only says whether the tumor is present or not in the given particular image and its results are quite accurate. In total these feature or computational feature removes human error and slowness from the process of detection and makes it a very good predictor which can help many people and patients around the world.

REFRENCE:

1) https://www.researchgate.net/publication/319623148_Identification_of_Brain_Tumor_using_Image_Processing_Techniques#:~:text=MRI%20is%20mainly%20used%20to,compared%20with%20other%20imaging%20technologies.&text=MRI%20images%20can%20be%20processed,using%20various%20image%20segmentation%20techniques.

2) https://ieeexplore.ieee.org/document/9076542

3)

https://www.ripublication.com/irph/ijisaspl2019/ijisav11n1spl_05.pdf

4)

https://braininformatics.springeropen.com/articles/10.1007/s40708-017-0075-5

5)

https://www.hindawi.com/journals/ijbi/2017/9749108/

BRAIN TUMOUR DETECTION

Done By:
MAYANK DASH (19BCE1663),
PRAKHAR RATHORE (19BCE1793)