Data Mining Final Project: Brain Tumor Classification

Sanjay Jars

DSC 550

Bellevue University

Abstract:

Images are used in various fields to make the problem easier to understand. Image processing techniques are most widely used in medical imaging to identify the affected area through an X-ray, computed tomography scan(CT scan), MRI scan(Magnetic resonance images). These images used to detect, identify, and locate the infections, abnormal growths from the human body. Heart diseases, Cancer, Brain tumor, Blood clotting, these are some of the abnormalities that can be found by medical imaging techniques. We can use different machine learning techniques to classify different types of brain tumors by using MRI. The convolutional neural network (CNN) is a class of deep learning neural networks that are highly effective with image classifications.

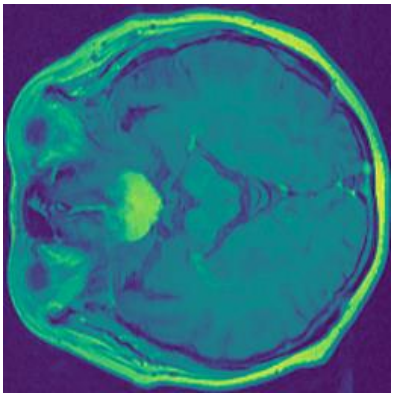
Introduction**:**

With every year, the number of patients with a brain tumor is increasing. There are two classes of brain tumors, primary and secondary tumors. Primary tumors have several types; one of the frequently found is a meningioma type. It is very challenging to locate, detect, and select the infected tumor portion in the brain from the MRI (Magnetic resonance images). This tedious and time-consuming job requires radiologists and medical field experts. The accuracy of this task is mainly subject to the experience and expertise of the person performing this task. So, if we use a machine learning model to perform this task, it will help to overcome the shortcomings of the

person involved in performing this task. So, I think if we can automate this process of classifying the tumors by using machine learning algorithms, it will improve the accuracy of the results and cost due to the expertise required. If we correctly classify the tumors, the specific treatment to that type of tumor can be applied. The accurate data about type and position assists in planning the surgical process for its extraction. An estimated 700,000 Americans are living with a brain tumor. Out of all brain tumors, approximately 69.8% of tumors are benign and 30.2% are malignant. An estimated 87,240 people will receive a primary brain tumor diagnosis in 2020.

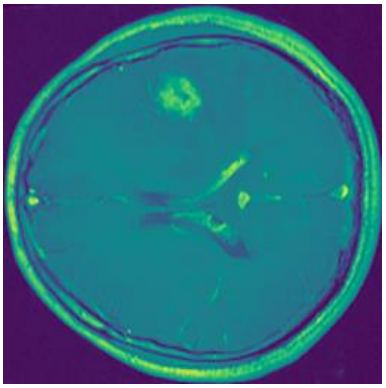
Types of brain tumors: For this implementation, I have considered following brain tumor type.

1. *Meningioma*: The meningioma type of tumors seen near the top-outer part of the brain. Meningioma is slowly growing noncancerous tumors that cause seizures and visual problems. This type of tumors accounting for 37.6% of all tumors, and 53.3% of all non-malignant tumors.

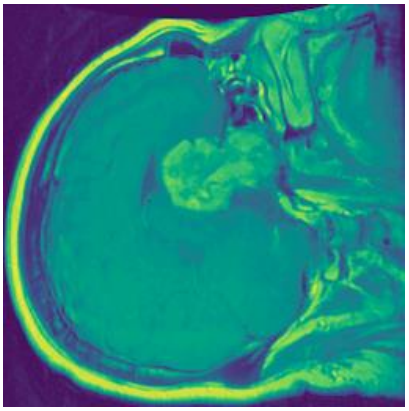


2. *Glioma*: Glioma is an abnormal growth in glial cells present around the neurons in the brain. Gliomas (such as glioblastoma, ependymomas, astrocytomas, and

oligodendrogliomas), which make up 81% of malignant brain tumors in adults.

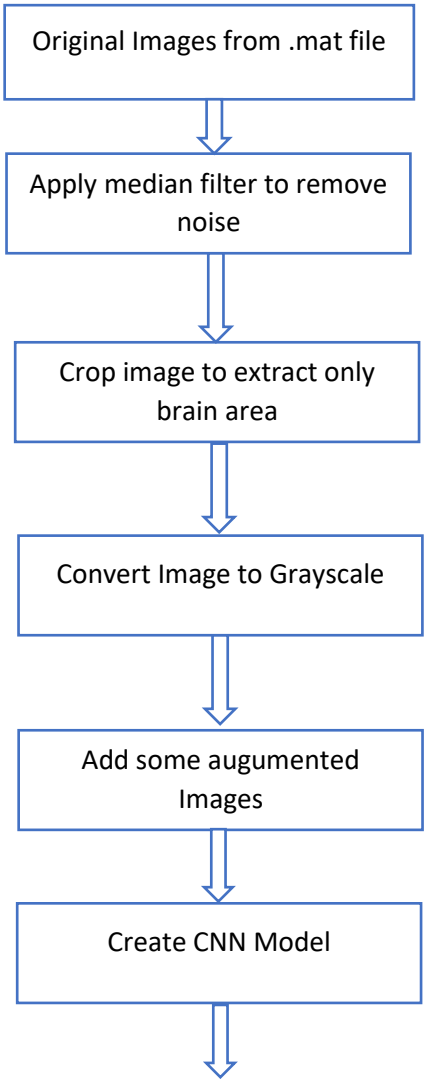


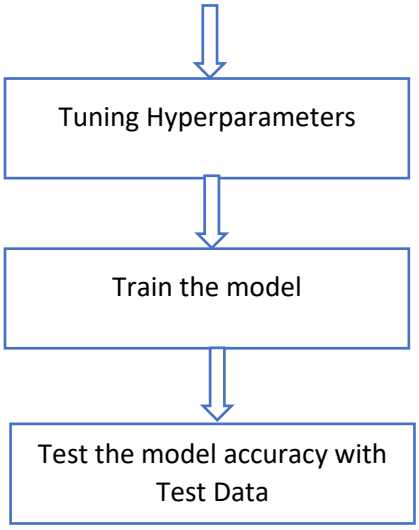
3. *Pituitary*: Pituitary tumors grow in pituitary glands that affect body functions. Some pituitary tumors result in too many of the hormones that regulate important functions of your body. Some pituitary tumors can cause your pituitary gland to produce lower levels of hormones.



Approach:

I will be following an approach comprising image preprocessing like noise removal, cropping the image, extraction, augmentation, and classification of the MRI images. Convolution neural networks are a breakthrough in image recognition. They're most commonly used to analyze visual imagery and are frequently working behind the scenes in image classification. In my approach, I used vector with size 65536 for each image, rather than using 3-dimensional arrays for images. As per my observation, it is taking less time for training with a vector. I have transformed RGB images to grayscale. This image transformation changes 3 RGB channels to only one channel, reduces the memory footprint, and training time.





Methodology:

1. Dataset:

The data set is available at the web site https://figshare.com/articles/brain_tumor_dataset/1512427. The brain tumor dataset contains 3064 T1-weighted contrast-enhanced images from 233 patients. The dataset includes the above discussed three kinds of tumor images. The data set comprises the following.

- 708 Meningioma images,
- 1426 Glioma images,
- 930 pituitary tumor images.

The dataset contains files in Matlab data format(.mat files). Every file contains a structure with the following fields.

a. cjdata.label: 1 for meningioma, 2 for glioma, 3 for pituitary tumor

b. cjdata.PID: patient ID

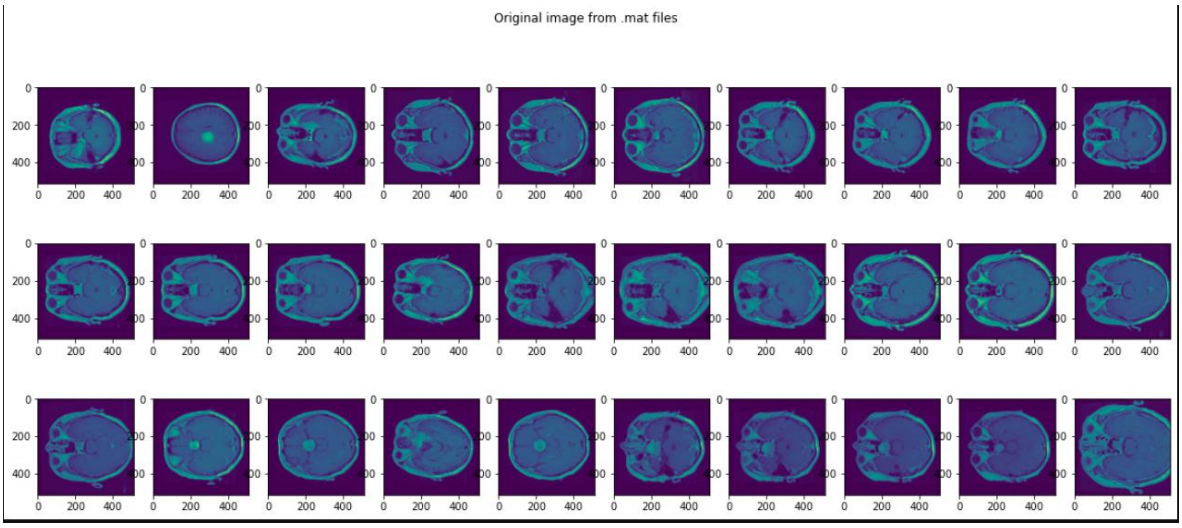
c. cjdata.image: image data

d. cjdata.tumorBorder: a vector storing the coordinates of discrete points on the tumor border.

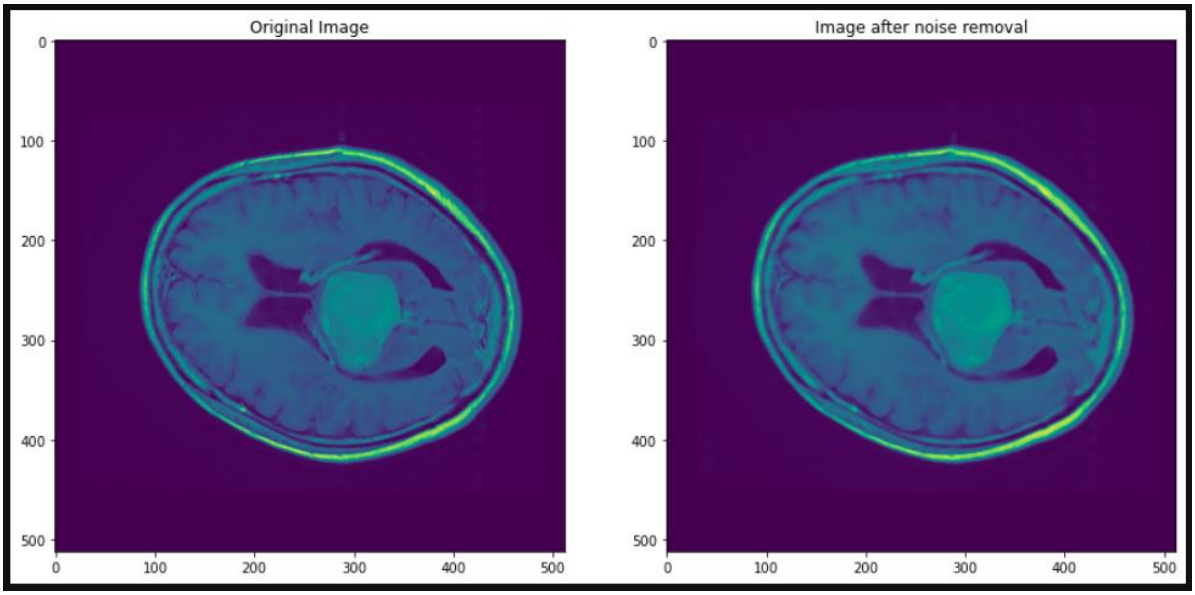
e. For example, [x1, y1, x2, y2,...] in which x1, y1 are planar coordinates on tumor border. It was generated by manually delineating the tumor border. We can use it to create a binary image of the tumor mask.

f. cjdata.tumorMask: a binary image with 1s indicating tumor region

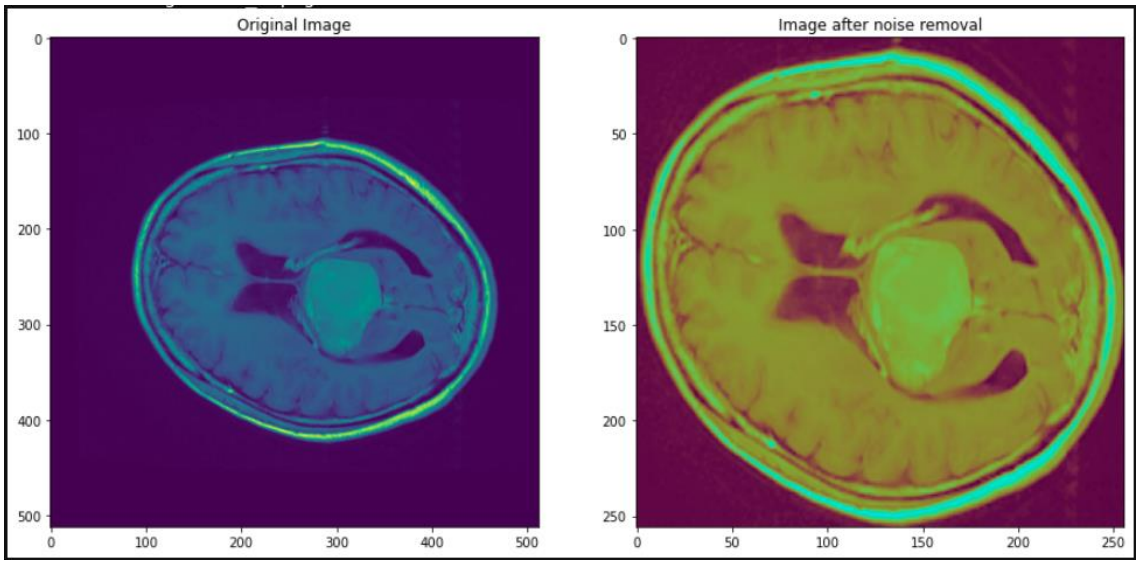
2. Reading .mat Files**:** The image and the read from the .mat file into a NumPy array by using the h5py python module. The following image displays the original images loaded from .mat files.



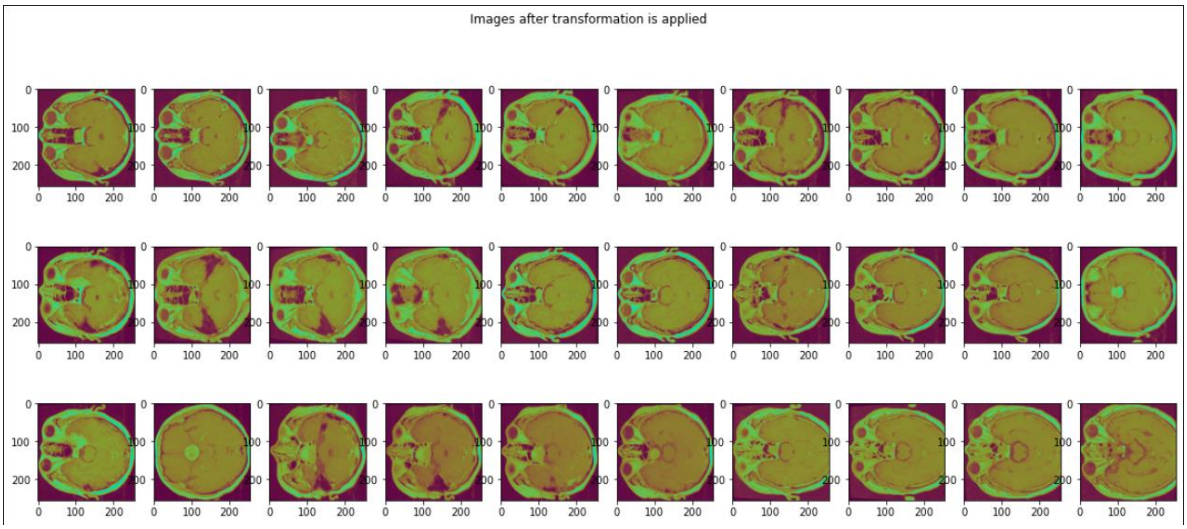
3. Remove noise: We applied the median filter to reduce the noise available. Please refer to [1] for particular. The following images show the transformation results.



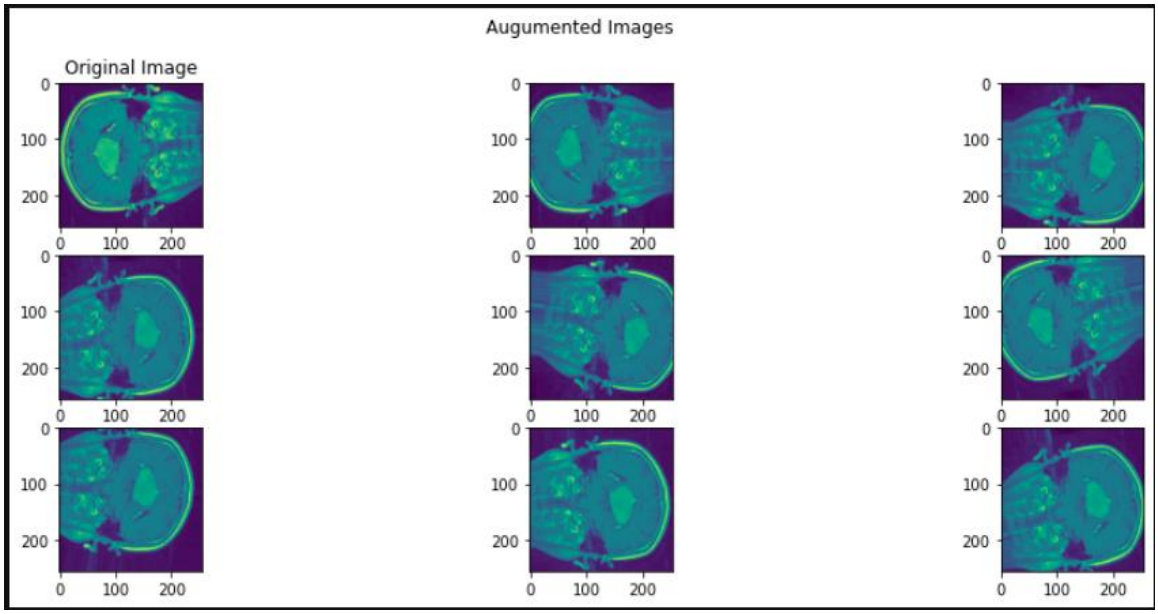
4. Crop Image: In this step, images resized to 256 X 256 pixels by using the opencv-python module. Please refer to [2] for finding out extreme points of images. The image cropped to get the most prominent contour in the picture by eliminating additional border areas. The following image shows the transformation effect. It confirms that we got rid of a lot of unnecessary data from the image.



5. Convert Images to Grayscale: All the images are converted to grayscale to reduce the memory footprint. This transformation will also improve training and prediction performance. By changing to grayscale reduces image size by 3-folds, as we are converting 3 RGB channels to only one channel. Because of this transformation, we can use a vector for an image instead of a 3-dimensional array.



6. Save Images: Transformed images saved on to disk as .png files for future usage.
7. Load Images: All transformed images from the above step loaded into the NumPy array. The labels converted into one-hot encoding. Meningioma class is coded as [1,0,0], Glioma class is coded as [0,1,0] and Pituitary class is code as [0,0,1].
8. Augmented Images: To avoid the overfitting of the model, I am adding some augmented images to the dataset. I used the ImageDataGenerator module from Keras to generate these images. With ImageDataGenerator, we can create new images from existing images by doing some transformations. We can apply alterations like rotating images by some angle, or shifting image by some pixels, or flipping across verticle or horizontal axis, etc. The following snapshot shows the original image with multiple augmented images.



9. Reshape the Data**:** The image data and label data array reshaped to vectors. After reshaping, the image dataset shape is (3164, 65536), and the label dataset shape is (3164, 3).

10. Split the Data in Training and Test Dataset**:** Dataset divided into two parts, 80 percent of data for training and 20 percent of data a test dataset. By doing this, we can sure test is not seen by the model in the training phase.

11. Create the CNN model**:** I am using Keras sequential model for my CNN model. The CNN model created with the Input as image data in batches with each image shape as 65536. I am using grayscale images, so the size of each image is smaller compared to using RGB. A similar picture with RGB will need an array of size 196, 608. The first layer used in the network is the Batch normalization layer. The batch normalization is a technique for training deep neural networks that normalize the input to a layer for every mini-batch. The batch normalization has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks[3]. The second and fourth layers of the model are the activation layer with activation type as ReLU. ReLU stands for the

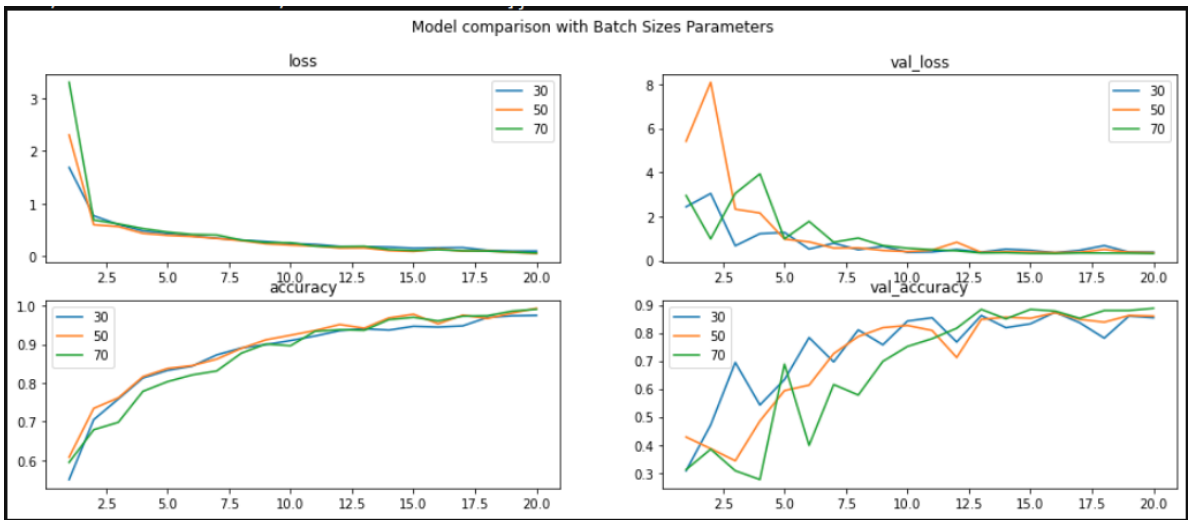
Rectified Linear Unit. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time. That means that the neurons will only be deactivated if the output of the linear transformation is less than 0[4]. The third layer is a densely connected layer with size as 100 with activation as ReLU. For the final layer, I am using the Dense layer with activation type as "softmax" to obtain the probabilities of all brain tumor classes.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bn0 (BatchNormalization)     (None, 65536)             262144
_________________________________________________________________
activation (Activation)      (None, 65536)             0
_________________________________________________________________
dense (Dense)                (None, 100)               6553700
_________________________________________________________________
activation_1 (Activation)    (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 303
=================================================================
Total params: 6,816,147
Trainable params: 6,685,075
Non-trainable params: 131,072
```

12. Hyperparameters Tuning**:** For selecting the best parameters, I tried to compare different parameter performance with the above model. I used 0.2 as a validation data ratio for cross-validating the model.

a. *Batch Size***:** I used the following batch sizes for comparison, 30, 50, and 70 with epoch as 20. The below plot shows the performance comparison of the model with different

batch sizes. I have selected batch size 70 for further tuning.



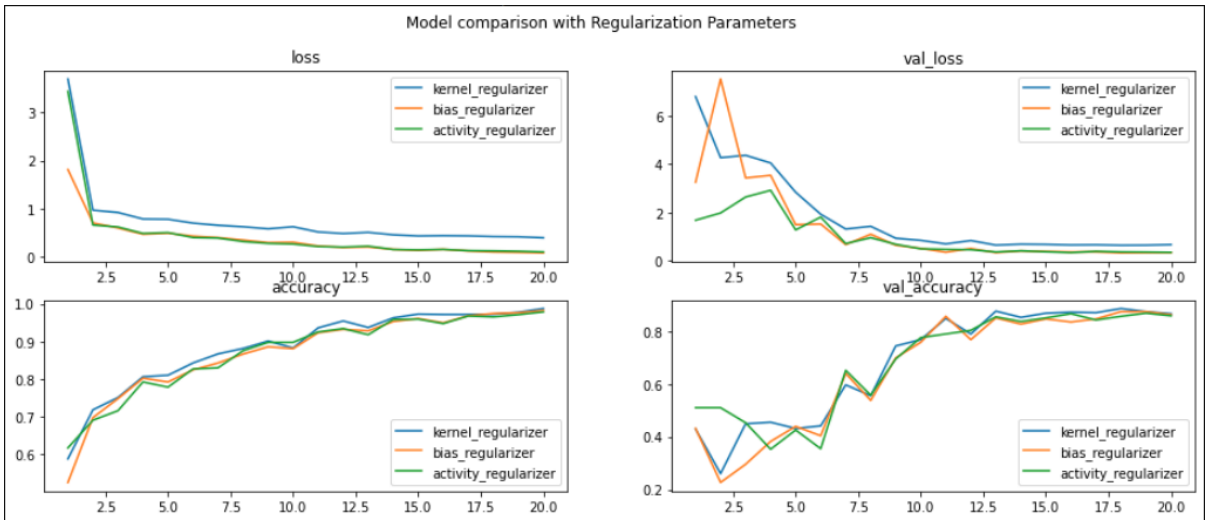
b. *Layer weight regularizers*: I ran model with 3 different regularizer values

kernel_regularizer: l1=1e-5 and l2=1e-4

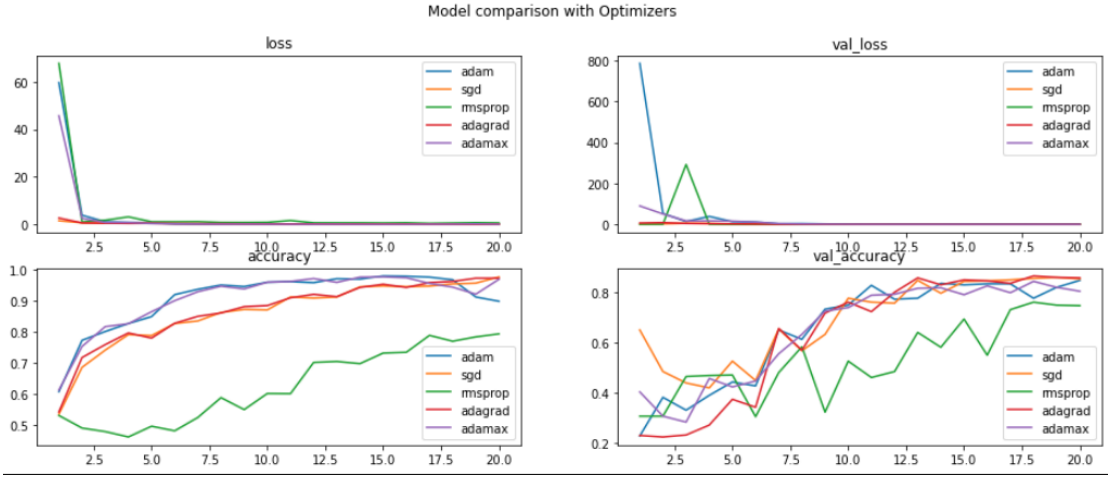
bias_regularizer: l2 = 1e-4

activity_regularizer: l2=1e-5

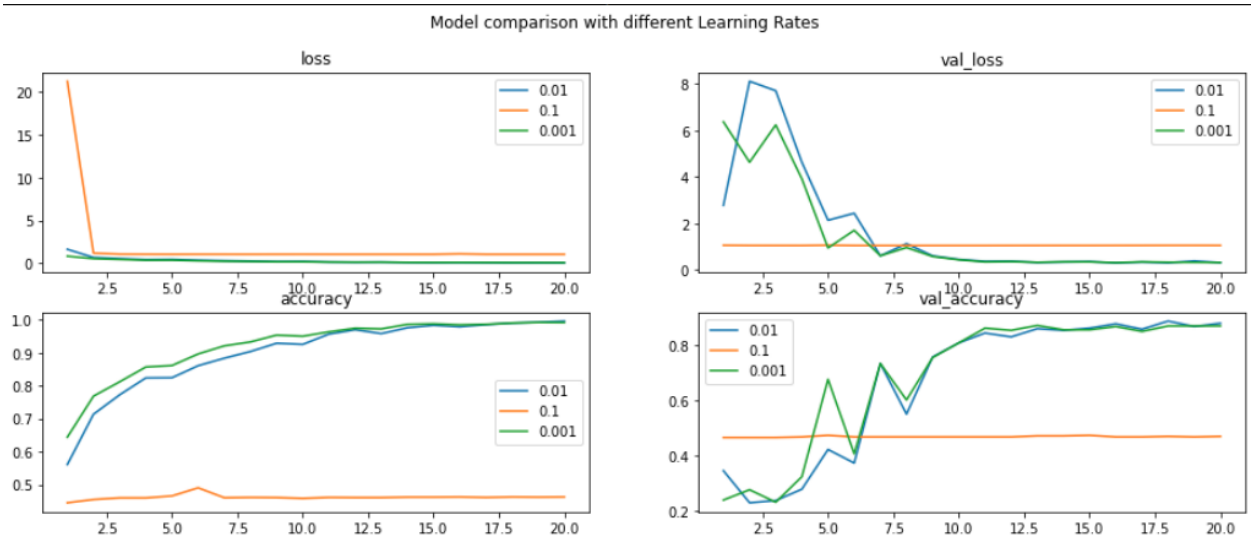
I have observed with bias regularizer and activity regularizer the training and validations loss is reduced.



c. *Optimizers*: I ran the model with the following optimizer types. The following plot displays the "Adagrad" optimizer is consistent. I used the Adagrad optimizer for further training of the model.



d. *Learning Rate*: I ran the model with three different learning rates 0.01, 0.1, and 0.001. The following plot shows that the learning rate of 0.1 performed worst, and 0.001 is slightly better than 0.01.



13. Train the model: After selecting hyperparameters, I have trained the model with the training dataset. The following hyperparameters used for training the model.

Optimizer: Adagrad, bias regularizer: l2=1e-4, learning rate:0.001 and batch size as 70. The model trained for 45 epochs with training data with the above parameters. The results of the training are as follows.

Loss: 0.0409, accuracy:0.9975 for training data. Loss:0.3085 and val_accuracy: 0.889 for validation data.

14. Test the Model**:** The model tested with 20 percent of the test data set that was not used in the training of the model. The model showed 0.34 loss, and approximately 88 percent accuracy similar to validation data. The model shows little more loss with test data, which shows the model is overfitted.

Conclusion**:** By using the Convolution neural network, we have created a model to classify brain tumors from MRI images. The accuracy of the model is approximately 88 percent for test data. The precision of the model can be further improved by using more image processing techniques like image segmentation. The overfitting of the model can be reduced by using more data along with some more image processing techniques like segmentation.

References:

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.median_filter.html

[2] https://www.pyimagesearch.com/2016/04/11/finding-extreme-points-in-contours-with-opencv/

[3] https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

[4] https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/#:~:text=The%20ReLU%20function%20is%20another,neurons%20at%20the%20same%20time.

[5] Brain Tumor Detection Using K-Means Clustering Algorithm: V.Surudhi, K.Sanjana, R.Saravanan, G.Santhosh, S.Kirubha https://pdfs.semanticscholar.org/b70e/4a5455a4531ca650272474d49d29e5e1da5c.pdf

[6] The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks and Image Classification: Anne Bonner: https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb

[7] QUICK BRAIN TUMOR FACTS: National Brain Tumor Society: https://braintumor.org/brain-tumor-information/brain-tumor-facts/