**MRI Brain Tumor Image Detection using Neural Networks**

****

**Authors:**

Victor Kira, 1890910, Computer Science, undergraduate

Tariq Kharsa, 1763722, Computer Science, undergraduate

**Introduction:**

In this article, we will present the implementation for training a set of images with tumor and non-tumor classification. All algorithms and the training model were developed using Python Tensorflow. The images were first processed using libraries such as Numpy and Pillow. Then, they were divided into a training set and a validation set. After the model has been successfully created, it is then tested for accuracy to check whether or not it predicts the classification correctly.

**Objectives:**

**Image acquisition -** Create a dataset with tumor and non-tumor images and divide these images into training and testing sets proportionally. Using Pillow and other libraries, open the image from each folder and store in their corresponding variables.

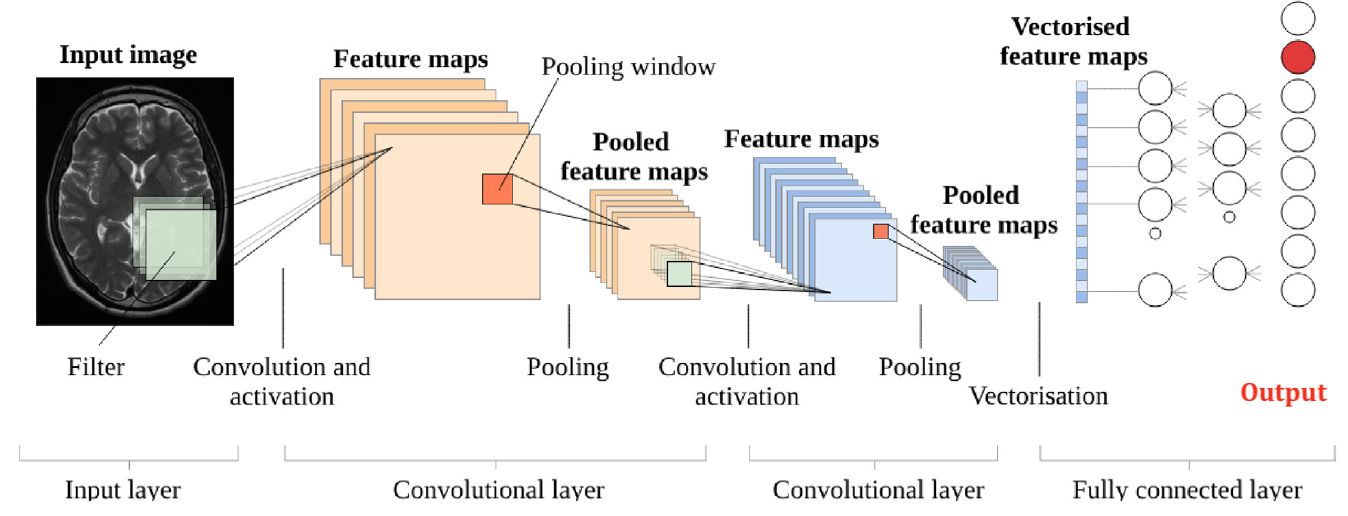
**Image manipulation -** Convert the image to a grayscale to allow for better accuracy. Resize the images into size 160 x 160 to fit the input shape for the training model. Convert the image from int [0, 255] to [0, 1] float-32 by dividing the image intensities by 255. Flatten the image into the input nodes for the neural network.

**Training the ML model -** Split the images into a training and validating set. Pass in the parameters and hidden layers to improve the model’s accuracy. Run epochs 10 times on the model for better results.

**Evaluating the ML model -** Using the testing set, evaluate the model and report its loss and accuracy. Just like the training set, the testing set should be pre-processed and converted into a numpy array.

**Algorithms:**

**Convolution Neural Network Diagram:**

****

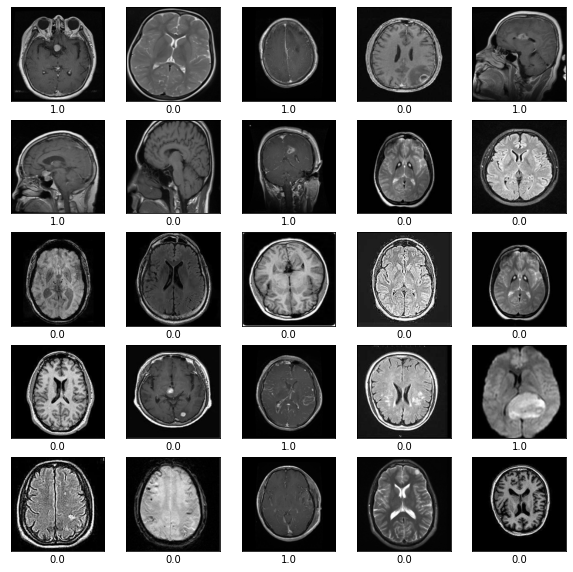
**Overall Algorithm:**

1. Open folder Train\_Tumor and Train\_None, resize images to 160x160 and append the images within the two folders into a list called d\_train, which be used for training the ML model
2. Open folder Test\_Tumor and Test\_None, resize images to 160x160 and and append the images within the two folders into a list called d\_test, which be used for evaluating the ML model
3. Slice the d\_train into 2 parts, validate set (x\_val,y\_val) and train set (x\_train,y\_train)
4. Store validate and train set itself as a float-32 array
5. Display the first 25 images of the x\_train to ensure the proper image transformation has been done and that x\_train set has been shuffled
6. Loop through the sets and dividing the each image intensities by 255
7. Flatten the image using the builtin function Flatten from keras.layers
8. Pass the two sets in the parameters and input layers which creates the ML model using:
   1. model = Sequential([Flatten(input\_shape = [160, 160]), Dense(300, activation = "relu" )
   2. Create hidden layers for the convolution neural network
   3. Add an output layer with 10 nodes and a softmax activation function
9. Call epochs 10 times to train the ML model using:
   1. model.fit(x\_train, y\_train, epochs = 10, validation\_data = (x\_val, y\_val)).
10. Use model.evaluate(x\_test, y\_test) function to evaluate the model and gain the result.

**Sample for the MRI images with labels:**

**0 - non-tumor**

**1 - tumor**

****

**First Implementation:**

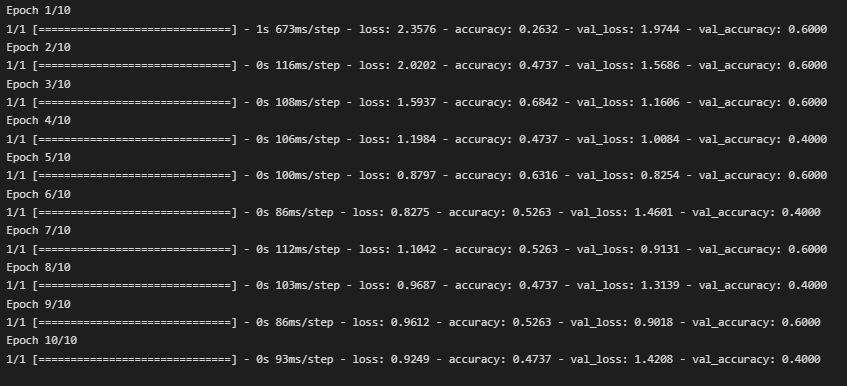
Image acquisition-

During the first implementation, our group was only able to acquire 12 MRI nontumor images and 12 MRI tumor images, making a total of 24 MRI images for our entire dataset. The way the images were acquired was through browsing through the internet for MRI images that are from normal patients and MRI images that are from patients with brain tumors. Then pick out singular images one by one and add it into the dataset. The method was inefficient but due to the lack of MRI images shared to the public, this was the method we resorted to during the first implementation.

Training the ML model-

Due to the lack of dataset given, the time taken for the machine learning model to train was extremely short. Each Epoch took less than 1 second to run. Epoch is when the dataset is passed forward and backward through the neural network once. As shown in image 1, the accuracy during the training went as low as 0.26 (26%) and topped out at 0.6316 (63.16%). For the validate accuracy during the model training, the number fluctuates between 0.4 (40%) and 0.6 (60%).

(Image 1)



Evaluating the ML model (results)-

After training the ML model, evaluation with a set of test cases was done. Image 2 displays the output for the evaluation. The results, as expected, did not have high accuracy, as the accuracy was 0.5 (50%) and loss being >1.

(Image 2)



**Second Implementation:**

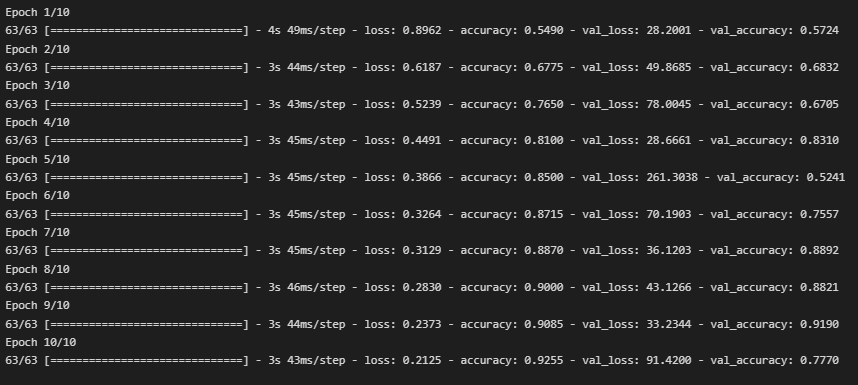
Image acquisition-

During the second implementation, we were able to acquire a much larger dataset of around 1500 MRI images of nonTumor patients and 1500 MRI images of patients who suffered from a tumor. Making a total of around 3000 images for the dataset. This time the MRI images are acquired from a website called debugger cafe which dedicates on sharing Projects related to Machine Learning and Deep Learning as well as their datasets. There, our group was able to find machine learning projects that also required the use of MRI images of patients and used a portion of MRI images from their dataset to train our MRI model.

Training the ML model-

Due to the large dataset, the time taken for the machine learning model to train was a few times longer than the First implementation. Each Epoch took around 3 to 4 seconds to run. As shown in image 3, the accuracy during the training ranges from 0.5490 (54.9%) to a staggering 0.9255 (92.55%). For the validate accuracy during the model training, similar to accuracy, the number ranges from 0.5241 (52.41%) to 0.9190 (91.90%).

(Image 3)



Evaluating the ML model (results)-

After training the ML model, evaluation with a set of test cases was done. Image 4 displays the output for the evaluation. The results display a high accuracy of 0.8550 (85.50%) and loss being <1 at 0.370.

(Image 4)

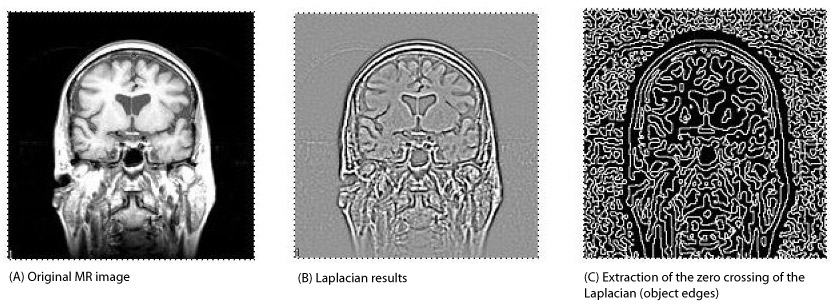


**Discussion:**

The results from this research/implementation showed two things: 1) The importance of data collection 2) The efficiency and potential of Machine Learning in the field of Medical Imaging.

As the Teacher Assistant once stated in class, the data collected and used for the dataset are at times more essential than the programming code itself. The result clearly supports this statement as the code for the ML model remained the same for both implementations, while the dataset was the one variable that changed. And yet the second implementation outperformed the first implementation by a 0.3550 accuracy difference.

The efficiency and potential of Machine Learning in the field of Medical Imaging is evident as only 3000 MRI images were needed to achieve an 85% accuracy for this research. Addition to that, it took the ML model merely minutes for it to train and reach this accuracy. With this projection, around 10000 MRI images might be able to result in a 95% or even higher accuracy for this research, but this requires further research and testing. Yes, indeed there are some other side factors that may contribute to this result. One instant, could be the test cases we used for this research, because different test cases may result in a different degree of accuracy for the ML model. Also, the fact that we are merely doing a simple image classification between identifying MRI images with brain tumor and without brain tumor. This may also contribute to this high degree of accuracy. Still, the dramatic increase in accuracy simply by increasing the dataset that we have seen in this research shows the potential of Machine Learning in the field of Medical Imaging. According to MRI statistics, there are 30 million MRI scans performed every year. Imagine even a slight portion of that 30 million is dedicated to Machine learning. This possibility of an ever-growing dataset serves as proof of the unlimited potential that Machine learning has in the Medical Field.

In regards to improvement that could have been done for the training model, the images could have undergone specific transformation such as Laplacian convolution to capture the important features of the MRI image before flattening the image and using them for input nodes. The transformation is depicted below:

**Conclusion:**

In conclusion, we presented the implementation of Machine learning on MRI images by training a set of images with tumor and non-tumor classification. The data were collected through internet research. All algorithms and the training model were developed using Python Tensorflow. The images were first processed using libraries such as Numpy and Pillow. Then, they were divided into a training set and a validation set. After the model has been successfully created, it is then tested for accuracy to check whether or not it predicts the classification correctly. The results from this implementation showed the importance of data collection and the potential of Machine Learning in the field of Medical Imaging.

**Reference:**

* Rath, Sovit Ranjan RathSovit Ranjan, and Name \*. “Brain MRI Classification Using Pytorch EFFICIENTNETB0.” *DebuggerCafe*, 23 Jan. 2022, https://debuggercafe.com/brain-mri-classification-using-pytorch-efficientnetb0/.
* Saha, S. (2022) *A comprehensive guide to Convolutional Neural Networks - the eli5 way*, *Medium*. Towards Data Science. Available at: [A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science](https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53) .
* panelAlexander SelvikvågLundervoldabPersonEnvelopeArvidLundervoldacdEnvelope *et al.* (2018) *An overview of deep learning in medical imaging focusing on MRI*, *Zeitschrift für Medizinische Physik*. Urban & Fischer. Available at: [An overview of deep learning in medical imaging focusing on MRI - ScienceDirect](https://www.sciencedirect.com/science/article/pii/S0939388918301181) .
* ​​Saha, S. (2022) *A comprehensive guide to Convolutional Neural Networks - the eli5 way*, *Medium*. Towards Data Science. Available at: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> .

**Appendix:**

* Training and Testing datasets under v\_data folder
* Training and Testing datasets under g\_data folder
* A Python notebook for the small dataset code with detail comments
* A Python notebook for the large dataset code with detail comments
* A readme file for instructions on how to run the code

**GitHub:**

GitHub link: <https://github.com/Tariqkharsa/Medical-Image-MRI-CNN-/tree/main>