Brain Tumor Detection

CAPSTONE Project

Udacity AWS_ML nano degree

1-Project Overview:

- Early detection and classification of brain tumors is an important research domain in the field of medical imaging and accordingly helps in selecting the most convenient treatment method to save patients life. Due to the increasing amounts of electronic data in the healthcare, medical doctors and physicians are facing problems in analyzing the data using traditional diagnosing systems. But with the help of machine learning a model can be trained to aid doctors and experts in detecting deadly diseases in their early stages. In this a Convolutional Neural Network machine learning model will be built and traind on Brain MRI Images Dataset.

2-Problem Statement:

- The first step in in machine learning problem is the dataset, how to collect the dataset, is it enough to build and train a machine learning model with acceptable accuracy, what is the type of the collected dataset and is it normally distributed and covering all the possibilities. In our case luckily Kaggle provides a good MRI images for 3 types of brain tumor [’glioma’, ’meningioma’, ’pituitary’] and no tumor so finally 4 labels. Will get into more details about the dataset later in this proposal. The second step is preprocessing the data to prepare it to be feed to the machine learning model and get some insights for the data which will help during the training phase. Third step is to build a suitable machine learning model using the suitable library, in our problem its a classification computer vision problem so I will use CNN also will try first to use transfer learning and use ResNet50.

3-Evaluation Metrics

- I will use accuracy as our problem is a computer vision classification problem so accuracy will be a good representation of the model performance also will use the confusion matrix as in my opinion its accepted to false classify a non tumor MRI as a tumor but not the other way around and the confusion matrix will be the best way to test the model performance in this case.

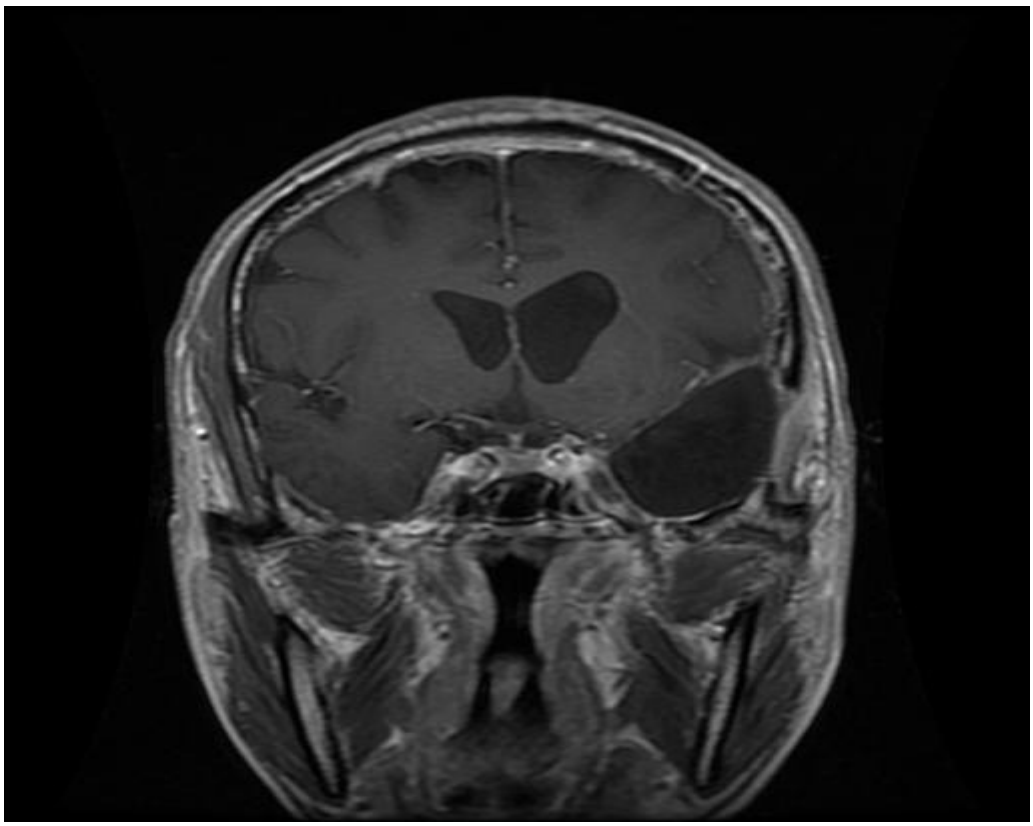
4-Data Exploration

- **To download the dataset click** here

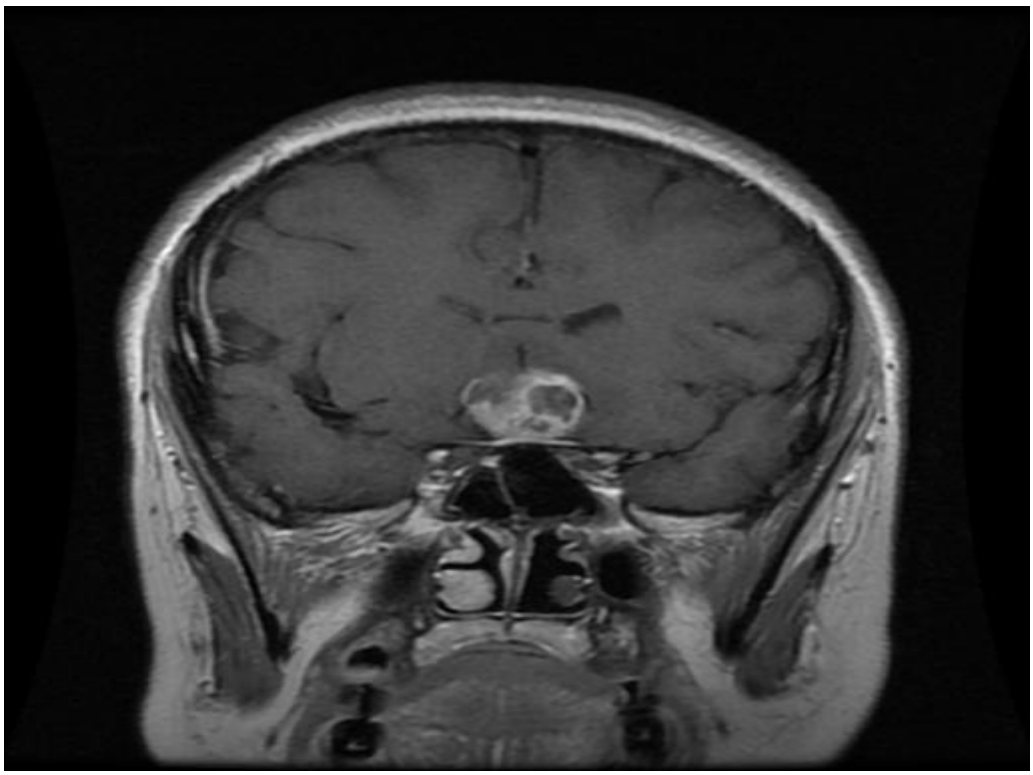
- **total number of training images is 5712 and divided on each type as follows:**
  - number of images of tumor type “glioma” is 1321 number of images of tumor type “pituitary” is 1457 number of images of tumor type “notumor” is 1595 number of images of tumor type “meningioma” is 1339
- **total number of testing images is 1311 and divided on each type as follows:**
  - number of images of tumor type “glioma” is 300 number of images of tumor type “pituitary” is 300 number of images of tumor type “notumor” is 405 number of images of tumor type “meningioma” is 306
- **the images shape is 512*512 black and white.**

5-Data Visualization

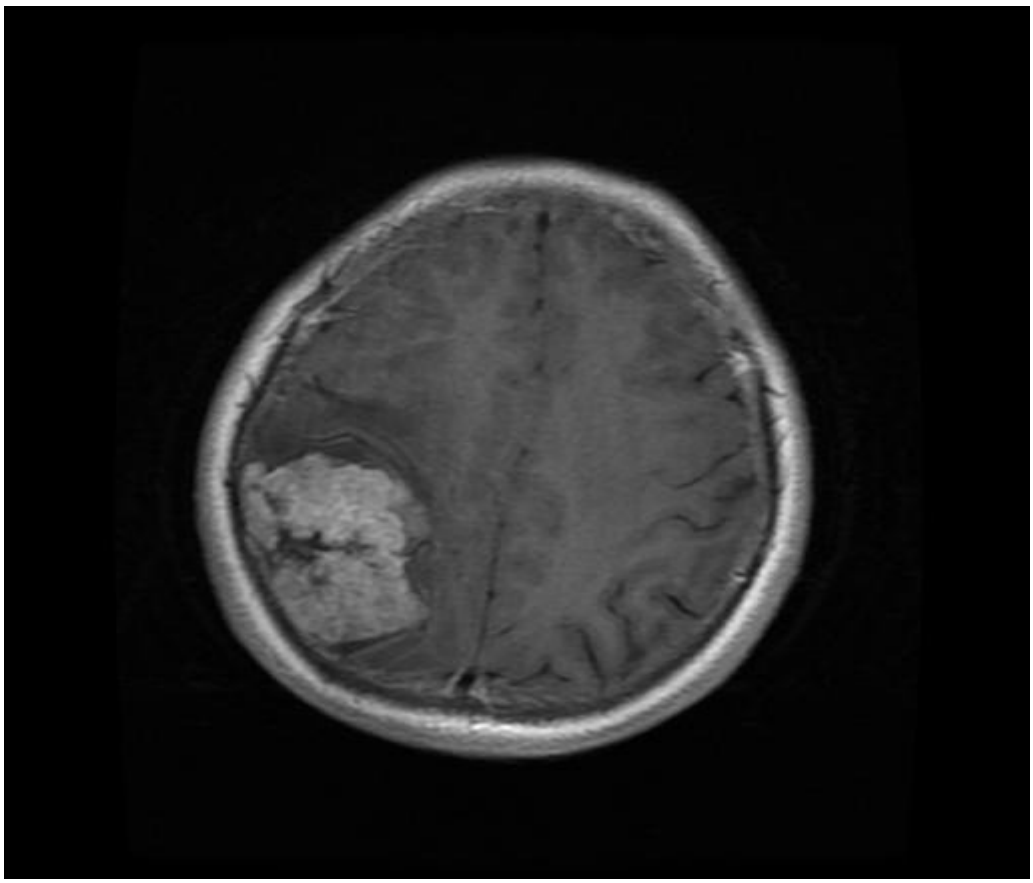
- sample image for tumor type “glioma”



- sample image for tumor type “pituitary”



- sample image for tumor type “meningioma”



6-Benchmark Model:

- brain tumor classification is performed by using Fuzzy C Means (FCM) based segmentation has been introduced in the following research paper which will be used as a benchmark to achieve the results `click here`.
- In this paper they used the accuracy

> In this work, efficient automatic brain tumor detection is performed by using convolution neural network. Simulation is performed by using python language. The accuracy is calculated and compared with the all other state of arts methods. The training accuracy, validation accuracy and validation loss are calculated to find the efficiency of proposed brain tumor classification scheme. In the existing technique, the Support Vector Machine (SVM) based classification is performed for brain tumor detection. It needs feature extraction output. Based on feature value, the classification output is generated and accuracy is calculated. The computation time is high and accuracy is low in SVM based tumor and non-tumor detection.

- They reached 95%.

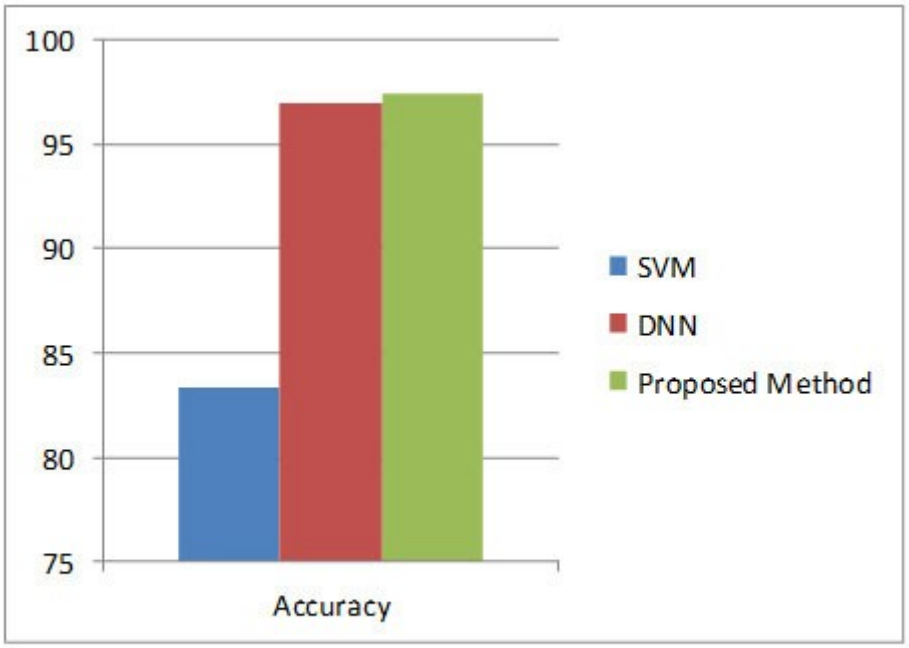


Figure 3: Accuracy of brain tumor classification

- In Our approach we reached 89% which can be enhanced this will be address in a later section.

7-Data Preprocessing

- I will create a data loader that will load the images from the folders with their corresponding label using pytorch and resize the images to be 224*224 as its the input layer of the ResNet50 model and finally convert it to tensor.

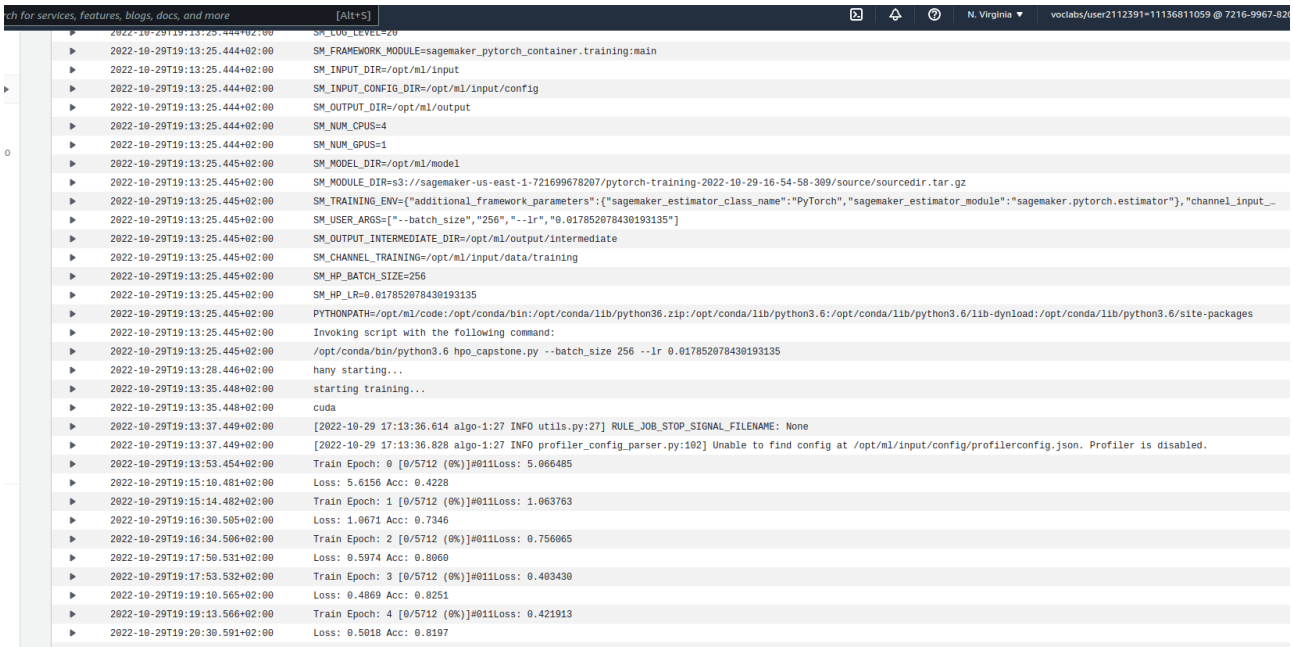
8-Implementation

- A convolutional neural network will be used and will use a ResNet50 pretrained model using pytorch.
- Sage maker notebook instance will be used with “ml.P2.xlarge” as instance type.
- Hyperparameters tuning job will be used to try to reach the best model usign the following hyperparameters:
  - Learning Rate
  - Batch Size

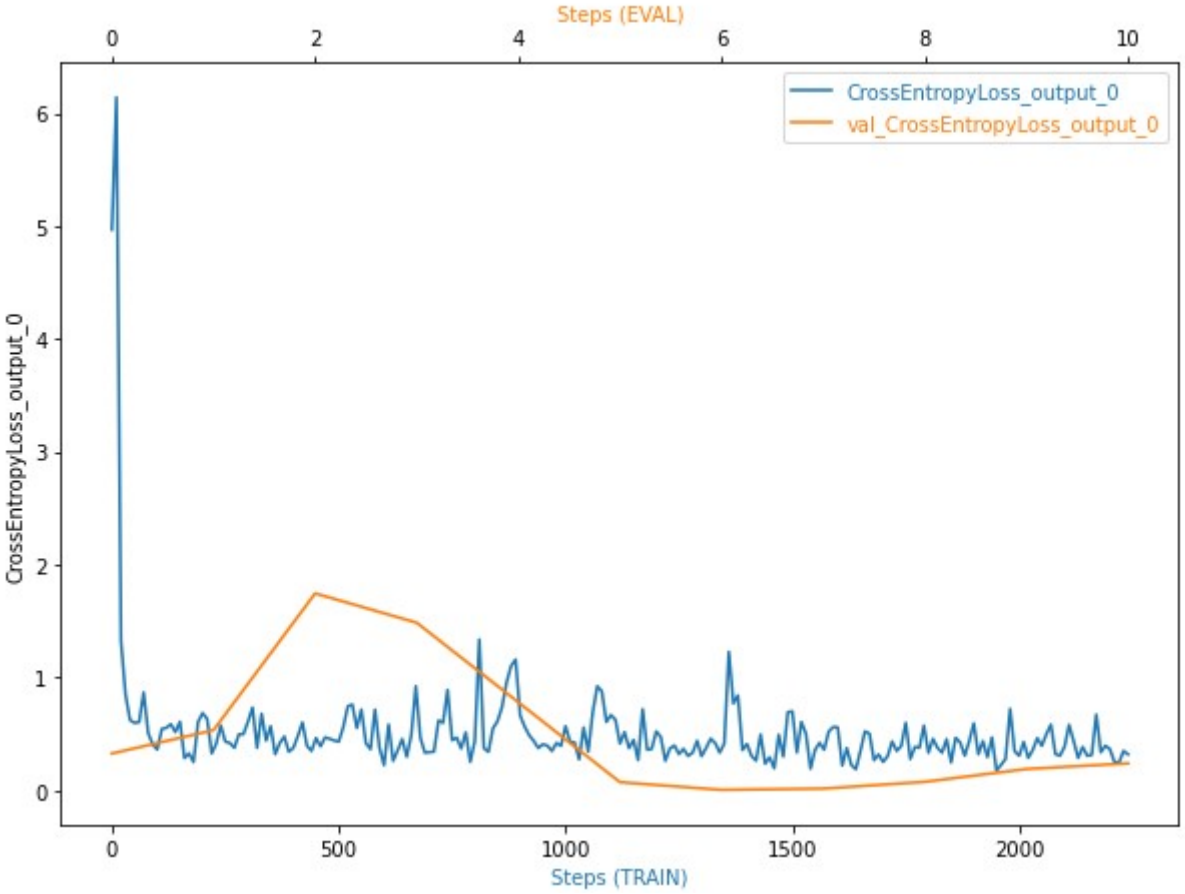
- Then Training job will be created using the best Hyperparameters from the previous step.

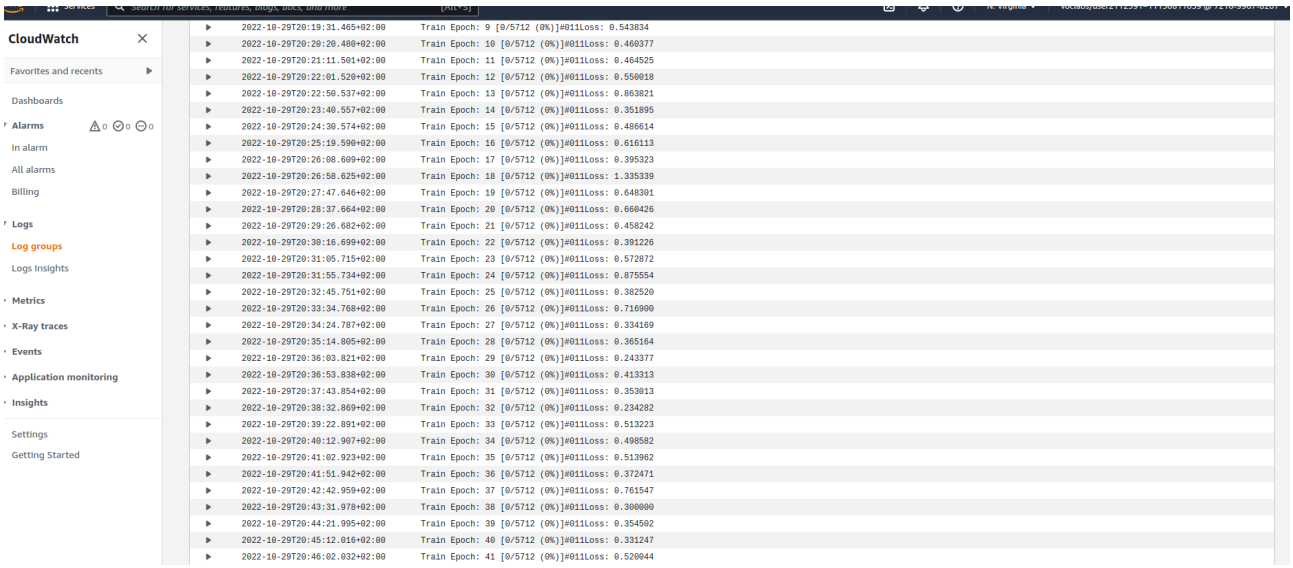
9-Model Evaluation and Validation

- The best hyperparameters found was
  - Learning Rate = “0.0142154184555958”
  - Batch Size = “128”



- The training is done using the best hyperparametrs and the results came as follows:


Steps (EVAL)
CrossEntropyLoss_output_0
val_CrossEntropyLoss_output_0
CrossEntropyLoss_output_0
Steps (TRAIN)


CloudWatch
Favorites and recents
Dashboards
Alarms
In alarm
All alarms
Billing
Logs
Log groups
Logs Insights
Metrics
X-Ray traces
Events
Application monitoring
Insights
Settings
Getting Started
2022-10-29T20:19:31.465+02:00 Train Epoch: 9 [0/5712 (0%)]#011Loss: 0.543834
2022-10-29T20:20:20.480+02:00 Train Epoch: 10 [0/5712 (0%)]#011Loss: 0.460377
2022-10-29T20:21:11.501+02:00 Train Epoch: 11 [0/5712 (0%)]#011Loss: 0.464525
2022-10-29T20:22:01.520+02:00 Train Epoch: 12 [0/5712 (0%)]#011Loss: 0.550018
2022-10-29T20:22:50.537+02:00 Train Epoch: 13 [0/5712 (0%)]#011Loss: 0.863821
2022-10-29T20:23:40.557+02:00 Train Epoch: 14 [0/5712 (0%)]#011Loss: 0.351895
2022-10-29T20:24:30.574+02:00 Train Epoch: 15 [0/5712 (0%)]#011Loss: 0.486614
2022-10-29T20:25:19.590+02:00 Train Epoch: 16 [0/5712 (0%)]#011Loss: 0.616113
2022-10-29T20:26:08.609+02:00 Train Epoch: 17 [0/5712 (0%)]#011Loss: 0.395323
2022-10-29T20:26:58.625+02:00 Train Epoch: 18 [0/5712 (0%)]#011Loss: 1.335339
2022-10-29T20:27:47.646+02:00 Train Epoch: 19 [0/5712 (0%)]#011Loss: 0.648301
2022-10-29T20:28:37.664+02:00 Train Epoch: 20 [0/5712 (0%)]#011Loss: 0.660426
2022-10-29T20:29:26.682+02:00 Train Epoch: 21 [0/5712 (0%)]#011Loss: 0.458242
2022-10-29T20:30:16.699+02:00 Train Epoch: 22 [0/5712 (0%)]#011Loss: 0.391226
2022-10-29T20:31:05.715+02:00 Train Epoch: 23 [0/5712 (0%)]#011Loss: 0.572872
2022-10-29T20:31:55.734+02:00 Train Epoch: 24 [0/5712 (0%)]#011Loss: 0.875554
2022-10-29T20:32:45.751+02:00 Train Epoch: 25 [0/5712 (0%)]#011Loss: 0.382520
2022-10-29T20:33:34.768+02:00 Train Epoch: 26 [0/5712 (0%)]#011Loss: 0.716900
2022-10-29T20:34:24.787+02:00 Train Epoch: 27 [0/5712 (0%)]#011Loss: 0.334169
2022-10-29T20:35:14.805+02:00 Train Epoch: 28 [0/5712 (0%)]#011Loss: 0.365164
2022-10-29T20:36:03.821+02:00 Train Epoch: 29 [0/5712 (0%)]#011Loss: 0.243377
2022-10-29T20:36:53.838+02:00 Train Epoch: 30 [0/5712 (0%)]#011Loss: 0.413313
2022-10-29T20:37:43.854+02:00 Train Epoch: 31 [0/5712 (0%)]#011Loss: 0.353013
2022-10-29T20:38:32.869+02:00 Train Epoch: 32 [0/5712 (0%)]#011Loss: 0.234282
2022-10-29T20:39:22.891+02:00 Train Epoch: 33 [0/5712 (0%)]#011Loss: 0.513223
2022-10-29T20:40:12.907+02:00 Train Epoch: 34 [0/5712 (0%)]#011Loss: 0.498582
2022-10-29T20:41:02.923+02:00 Train Epoch: 35 [0/5712 (0%)]#011Loss: 0.513962
2022-10-29T20:41:51.942+02:00 Train Epoch: 36 [0/5712 (0%)]#011Loss: 0.372471
2022-10-29T20:42:42.959+02:00 Train Epoch: 37 [0/5712 (0%)]#011Loss: 0.761547
2022-10-29T20:43:31.978+02:00 Train Epoch: 38 [0/5712 (0%)]#011Loss: 0.300000
2022-10-29T20:44:21.995+02:00 Train Epoch: 39 [0/5712 (0%)]#011Loss: 0.354502
2022-10-29T20:45:12.016+02:00 Train Epoch: 40 [0/5712 (0%)]#011Loss: 0.331247
2022-10-29T20:46:02.032+02:00 Train Epoch: 41 [0/5712 (0%)]#011Loss: 0.520044

12-Project refinement and improvement

- We can improve the performance using the following:
    - Adding scheduler learning rate which will decrease the lr every step this step can be 10 epochs for eaxmple.
    - Using other types of pretrained model as DenseNet-121 can be used to enhance the accuracy.
    - Increasing number of epochs and using early stopping to only save the best model.

11-Justificatiion

- The problem was building a suitable model that classifies the brain Tumor, A data is used from Kaggle, The data set was good but may be more data could do some enhancement on the results.
- A CNN model using transfer learning is used, I used ResNet50 I guess other types of pretrained model as DenseNet-121 can be used to enhance the accuracy.
- AWS Sagemaker notebook instance will be used of the type “ml.P2.Xlarge” and hyperparameters tuning will be used to get the best model.