Brain Tumor Detection and Localization using Deep Learning

Title: “Association of genomic subtypes of lower grade gliomas with shape features automatically extracted by a deep learning algorithm”

Author: Mateusz Buda, Ashirbani Saha, Maciej A. Mazurowski

Year: 2019

Conference/Journal Name: Computers in Biology and Medicine

Kaggle dataset: https ://www.kaggle.com/ lgg mri segmentation

Subject: Applied Machine Learning (ITCS 5156)

Name: Sourabh Kumbhar (801254780)

Github: https://github.com/Sourabh29/Brain-Tumour-Detection

I. Introduction

1) Problem Statement:

Recent analysis identified distinct genomic subtypes of lower-grade glioma tumors which are associated with shape features. In this study, we explore a fully automatic way to quantify tumor imaging characteristics using deep learning-based segmentation and test whether these characteristics are predictive of tumor genomic subtypes [1]. Using the deep learning methodology namely the CNN based ResUnet architecture we will localize the tumor from the MRI and map out the same using the segmentation.

2) Motivation

To explore how deep learning methodologies can help in the health industry primarily for cancer/tumor detection and localization and implement them on available dataset to learn how the procedure works and how impactful it is. This would drastically reduce the cost of cancer diagnosis & help in early diagnosis of tumors which would essentially be a life saver.

3) Challenges

Primarily, for image-based machine learning or deep learning tasks, CNNs are a widely opted and famous choice for the same.

Although CNN offers advantages in object detection processing, it also has significant drawbacks:

(a) Both the training and detection processes take a long time, and the normalization method causes some discriminative details to be lost.

(b) As CNN's grow deeper, vanishing gradient tend to occur which negatively impact network performance.

(c) CNN is commonly used when the entire image must be classified as a class label. many tasks, on the other hand, require classification of each pixel of the image.[3]

(d) These operations are highly resource intensive and having good GPUs are mostly necessary to run complex operations or high number of iterations.

Given the above challenges with the CNN alongside its performance issues for tasks like these, it was important for better CNN based models to be implemented for these which do not degrade in performance with segmentation centric operations.

4) Summary of the solution (approach)

For tumor detection and localization, the method applied deep learning strategies based on CNN. To identify the tumors, we use the CNN based ResUnet model. More layers in classic neural networks imply a better network, but the first layer's weights will not be updated appropriately by back propagation due to the vanishing gradient problem. The error gradient is small because it is repeatedly multiplicated back to prior layers. as a result, the network's performance becomes saturated and begins to decrease significantly as it grows in layers.

Resnet addresses this problem with the identity matrix. The gradient is only multiplied by one when using the identity function for back propagation. The input is saved and no data is lost as a result of this. Resnet model is also immune to the vanishing gradient issue which is of a big concern with CNNs during the segmentation tasks.

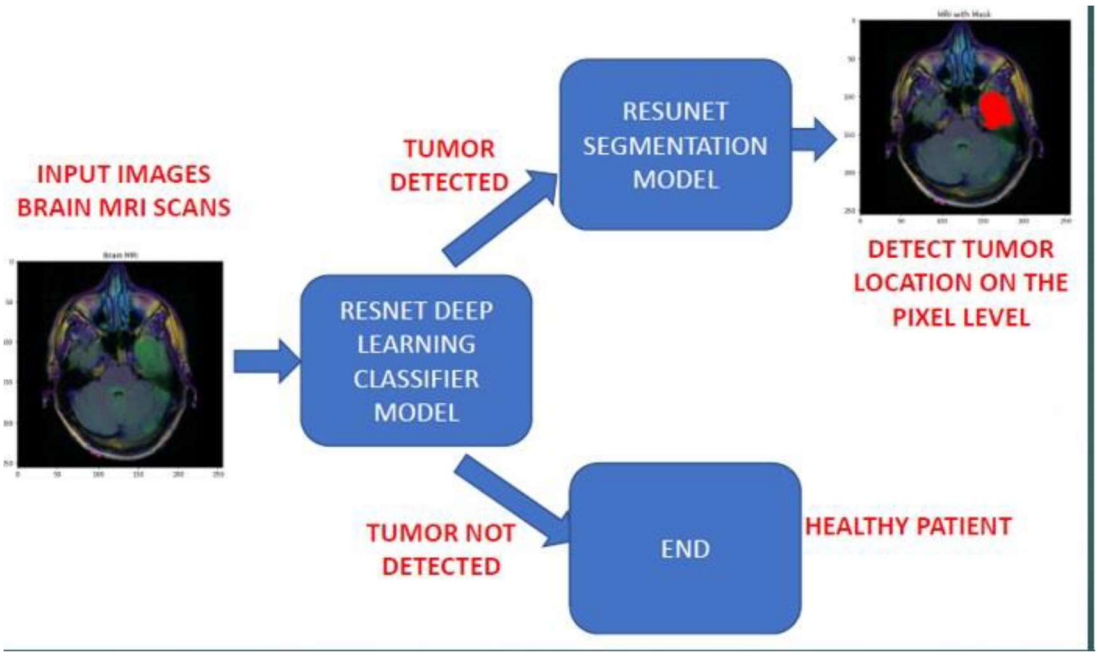


Fig: Basic layout of the working

II. Backgrounds/Related Work

1) Survey of the literature (2 other works)

The primary studies involved research about the performance assessment of various deep learning methodologies for tumor detections in various parts of the human anatomy.

Amongst all, two comparative studies were from papers 'Brain tumor detection from MRI images using deep learning techniques' published in the 'IOP Conference Series: Materials Science and Engineering' journal and 'Comparison of machine learning methods for classifying mediastinal lymph node metastasis of non-small cell lung cancer from 18F-FDG PET/CT images' published in 'EJNMMI Research' journal which discussed the CNN and its working in image analysis.

2) Summary of approaches, pros and cons

CNN is regarded as one of the most effective methods for analyzing picture datasets. The CNN makes the forecast by shrinking the image without losing the required necessary information to make the prediction.[3]

The approaches discussed in the above papers primarily deal with the ANN and CNN models which have been widely used for a variety of application in the deep learning domain. In the first paper [3], the experiments were primarily based on the ANN and the CNN algorithms where the CNN accuracy outperformed the ANN accuracy.

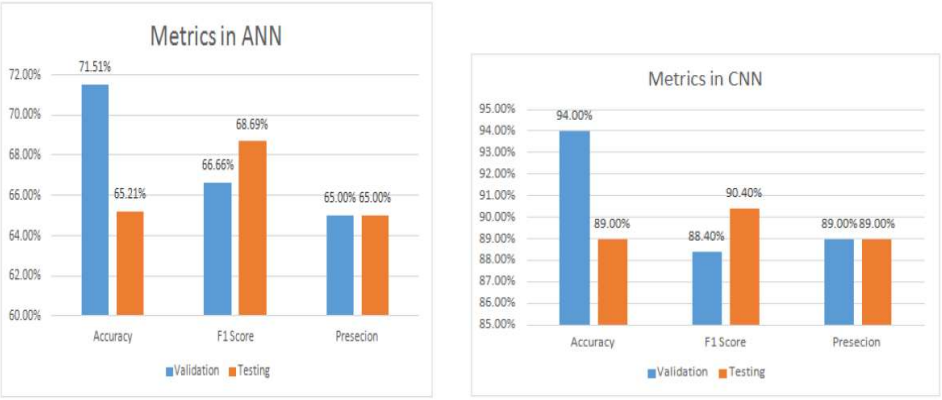


Fig [3]: Concluding Metrics

ANN model generated here produces 65.21% of testing accuracy whereas the CNN model generates an accuracy of 89%. The paper concluded that Optimization techniques will be utilized in the future to determine the number of layers and filters that can be used in a model.[3]

In the second paper [4], a wide range of methodologies such as Adaboost, ANN and CNN with various types were explored. In the concluding results, even though the CNN accuracy was about 85%, it was not prominently better than the other approaches for the segmentation-based operations. Even though critical diagnostic characteristics such as SUV and tumor size were not used in this investigation, CNN's performance was not significantly different from that of the best traditional approaches. Furthermore, because CNN does not employ hand-crafted features as input, it eliminates the need for tumor segmentation and feature selection, making the entire process considerably easier and less prone to user bias.[4]

3) Relation to the used approach

For tumor detection and localization, the method applied deep learning strategies based on CNN. To identify the tumors, we use the CNN based ResUnet model. More layers in classic neural networks imply a better network, but the first layer's weights will not be updated appropriately by back propagation due to the vanishing gradient problem. The error gradient is small because it is repeatedly multiplicated back to prior layers.

As a result, the network's performance becomes saturated and begins to decrease significantly as it grows in layers. Res net addresses this problem with the identity matrix. The gradient is only multiplied by one when using the identity function for back propagation. The input is saved and no data is lost as a result of this.

ResNet employs a skip connection, which means that an original input is also appended to the convolution block's output. This aids in the resolution of the vanishing gradient problem by providing an alternate path for the gradient to flow through. They also employ the identity function, which aids the higher layer in performing as well as the lower layer, if not better.

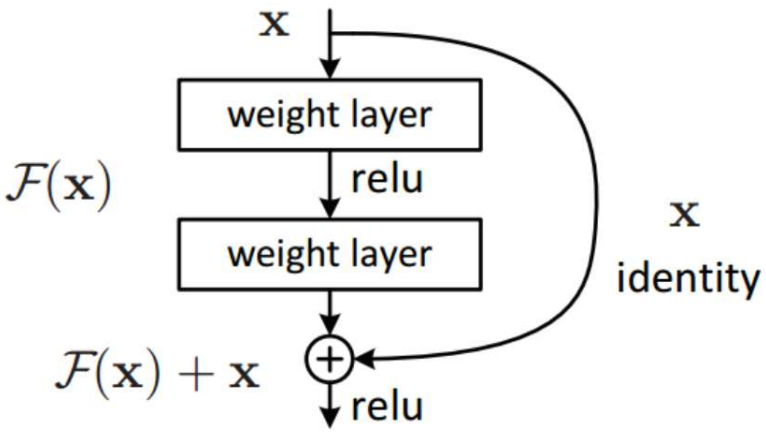


Fig: Skip connection Residual Block

III. Method

1) Detailed description of Method/algorithms with architecture.

CNNs are comparable to neural networks in that they have a number of neutrons with different weights and biases that may be learned. A number of inputs are given to each neuron, a weighted sum is performed, an activation function is applied, and an output is given. The network has a loss function that is used to reduce weight error. A machine perceives an image as a pixel matrix with a resolution of h x w x d, where h denotes height, w denotes width, and d denotes dimension. d is determined by the color scale, which ranges from 3 for RGB to 1 for grayscale. CNN converts an image into a vector, which is commonly utilized in classification issues.

U-Net:

In U-Net, however, an image is turned to a vector, which is then translated back to an image using the same mapping. This reduces distortion by keeping the image's original structure. When the entire image needs to be classed as a class label, CNN is frequently employed. However, many tasks necessitate classifying each visual pixel. The U-net and Res-Net solve this problem.[1]

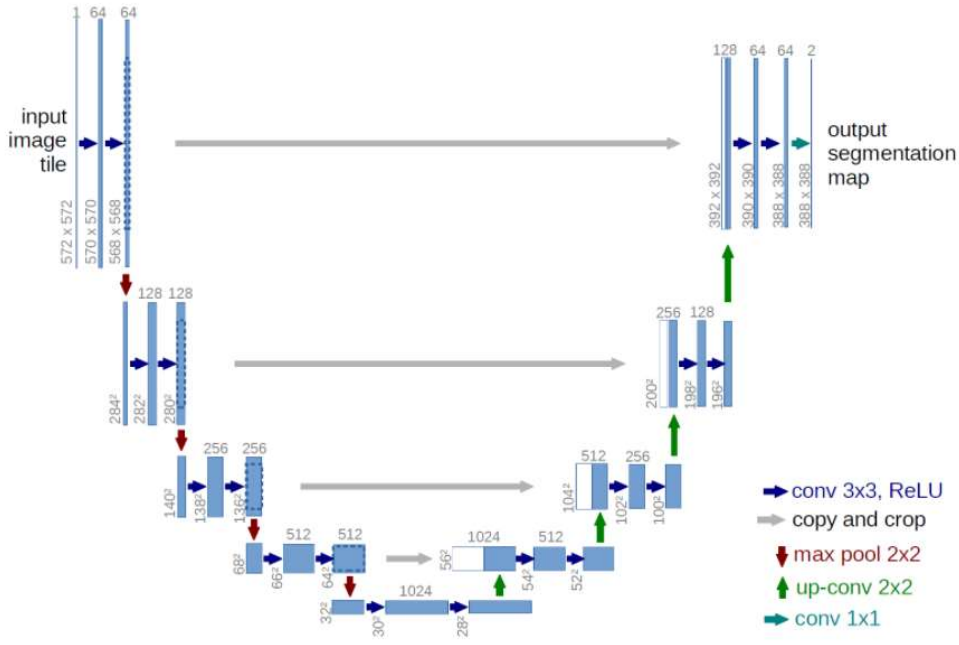


Fig: U-net Architecture [6]

Convolution Operation, Max Pooling, ReLU Activation, Concatenation, and Up Sampling Layers make up U-Net, which is divided into three sections: contraction, bottleneck, and expansion. There are four contraction blocks in the contractions section.

Each contraction block receives an input before applying two 3X3 convolution ReLu layers and a 2X2 max pooling. At each pooling layer, the number of feature mappings doubles. Two 3X3 Conv layers plus a 2X2 up convolution layer make up the bottleneck layer. The expansion component is made up of many expansion blocks, each of which sends the data to two 3X3 Conv layers and a 2X2 up sampling layer, which reduces the number of feature channels by half. A concatenation with the suitably clipped feature map from the contracting path is also included.

Finally, a 1X1 Conv layer is utilized to make the number of feature maps equal to the desired number of segments in the output. For each pixel of the image, U-net employs a loss function. Individual cells inside the segmentation map can now be easily identified. Each pixel is given a SoftMax value, which is then followed by a loss function. This changes the challenge from segmentation to classification, requiring us to assign each pixel to one of the classes.[6]

Image Segmentation:

The goal of image segmentation is to understand and extract information from images at the pixel-level. Image Segmentation can be used for object recognition and localization which offers tremendous value in many applications such as medical imaging and self-driving cars etc.

The end result of image segmentation is to train a neural network to produce pixel-wise mask of the image. Modern image segmentation techniques are based on deep learning approach which makes use of common architectures such as CNN, FCNs (Fully Convolution Networks) and Deep Encoders-Decoders. I have used ResUNet architecture to solve the current task.

In traditional CNN for image classification problems, we had to convert the image into a vector and possibly add a classification head at the end. However, in case of Unet, we convert (encode) the image into a vector followed by up sampling (decode) it back again into an image. In case of Unet, the input and output have the same size so the size of the image is preserved.

For classical CNNs: they are generally used when the entire image is needed to be classified as a class label. For Unet: pixel level classification is performed. U-net formulates a loss function for every pixel in the input image. Softmax function is applied to every pixel which makes the segmentation problem works as a classification problem where classification is performed on every pixel of the image.

ResUNet:

More layers in classic neural networks imply a better network, but due to the vanishing gradient problem, the first layer's weights will not be updated appropriately by back-propagation. The error gradient is modest because it is back-propagated to prior layers by repeated multiplication. As a result, as the network grows in layers, its performance becomes saturated and begins to decline significantly. The identity matrix is used by Res-Net to tackle this problem. When using the identity function for back-

propagation, the gradient is only multiplied by one. This ensures that the input is preserved and that no data is lost.

ResUNet architecture combines UNet backbone architecture with residual blocks to overcome the vanishing gradients problems present in deep architectures. Unet architecture is based on Fully Convolutional Networks and modified in a way that it performs well on segmentation tasks. Resunet consists of three parts:

- Encoder or contracting path
- Bottleneck
- Decoder or expansive path

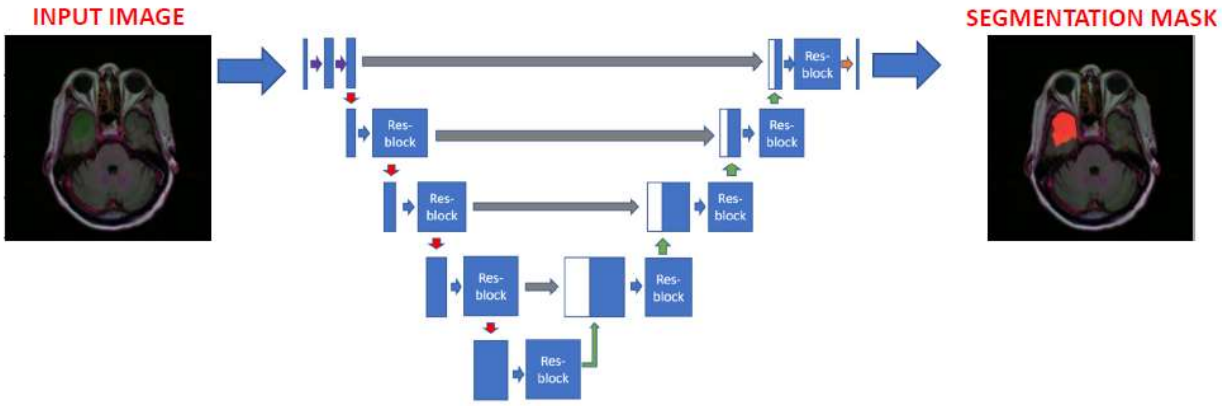


Fig: Resnet Architecture

ENCODER (CONTRACTION PATH)

The contraction path consists of several contraction blocks, each block takes an input that passes through res-blocks followed by 2x2 max pooling. Feature maps after each block doubles, which helps the model learn complex features effectively.

First block consists of 3x3 convolution layer + Relu + Batch-Normalization, remaining three blocks consist of Res-blocks followed by Max-pooling 2x2.

BOTTLENECK

The bottleneck block, serves as a connection between contraction path and expansion path. The block takes the input and then passes through a res-block followed by 2x2 up-sampling convolution layers. It is in-between the contracting and expanding path. It consists of Res-block followed by up sampling conv layer 2x2.

DECODER (EXPANSION PATH)

The expansion or decoder section of this architecture is a significant benefit. Each block concatenates the previous layer's up-sampled input with the matching output features from the res-blocks in the contraction path. This is then sent through the resblock once again before being passed through 2x2 up-sampling convolution layers.

This ensures that features learned when contracting are applied to the image reconstruction. Finally, the output from the res-block is transmitted through a 1x1 convolution layer in the last layer of the expansion route to produce the desired output with the same size as the input.

3 blocks following bottleneck consist of Res-blocks followed by up-sampling conv layer 2x2. Final block consists of Res-block followed by 1x1 conv layer.

Masking:

The goal of image segmentation is to understand the image at the pixel level. It associates each pixel with a certain class. The output produce by image segmentation model is called a “mask” of the image. Masks can be represented by associating pixel values with their coordinates. For example, if we have a black image of shape (2,2), this can be represented as:

[[0, 0],
[0, 0]]

if our output mask is as follows:

[[255, 0],
[0,255]]

To represent this mask, we have to first flatten the image into a 1-d array. This would result in something like [255,0,0,255] for mask. Then, we can use the index to create the mask. Finally, we would have something like [1,0,0,1] as our mask.

IV. Experiments

1) Explanation of experimental setup

I used the Kaggle environment itself for the experimental setup. Since the data file is extensive and is already available on Kaggle, you can simply import it in your notebook while designing the same over Kaggle. Also, Kaggle provides 36-hours of free GPU access which was necessary for this setup to boost up the processing speed while training of the models. We are using TensorFlow and keras along with NumPy, pandas, matplotlib, etc.

We do the necessary imports followed by data visualizations to understand the data. We then do the train and test data splits and import the ResNet50 model from the TensorFlow library.

```
from tensorflow.keras.applications.resnet50 import ResNet50
clf_model = ResNet50(weights='imagenet', include_top=False, input_tensor=Input(shape=(256,256,3)))
clf_model.summary()
```

I have then trained the classifier and the segmentation models for 30 iterations(epochs) over the dataset.

```
h = model.fit(train_generator,
              steps_per_epoch= train_generator.n // train_generator.batch_size,
              epochs = 30,
              validation_data= valid_generator,
              validation_steps= valid_generator.n // valid_generator.batch_size,
              callbacks=[checkpointer, earlystopping])
```

```
h = seg_model.fit(train_data,
                  epochs = 30,
                  validation_data = val_data,
                  callbacks = [checkpointer, earlystopping, reduce_lr]
                 )
```

For this project, I have used the Focal Tversky loss function which has good results for U-net architectures for segmentation operations. The original paper had implemented dice co-efficient loss function but the above method semes to be better for U-net architectures. The dice co-efficient gave an accuracy of 82% where as the accuracy by Tversky function is much higher.

For additional details about the loss function and as my source for its code used while implementation, I am providing the github link for the same: https://github.com/nabsabraham/focal-tversky-unet

Finally, the results are displayed which show the original MR images with the AI predicted tumor masks.

2) Test results of the proposed method.

The ResUNet architecture is extremely effective at the image segmentation and localization of the tumors. More iterations you allow to train the model for, its results can get better until a certain threshold.

For the Classifier model, I trained it with 30 epochs and received an overall accuracy of 94.57% and the segmentation accuracy was 90.48% (higher than the dice co-efficient method used in paper giving accuracy of 82%).

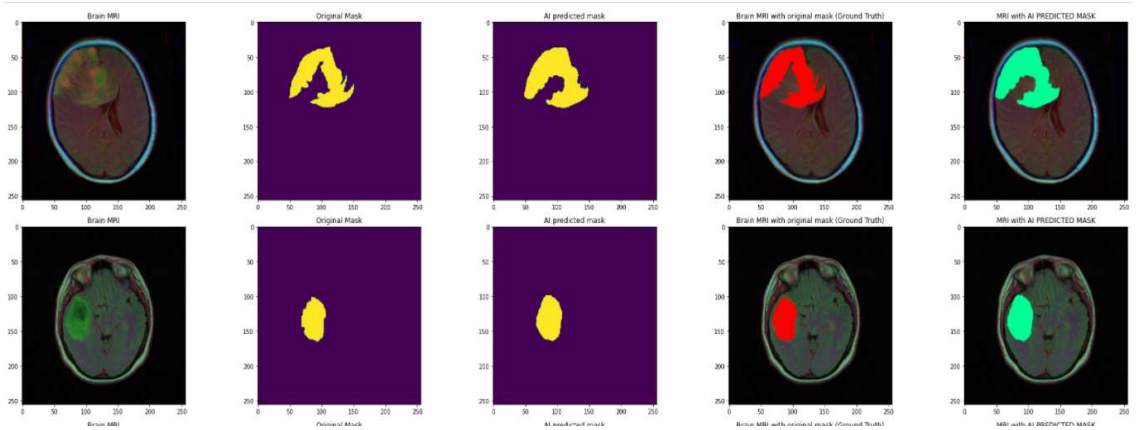
```
_, acc = model.evaluate(test_generator)
print("Test accuracy : {} %".format(acc*100))
```

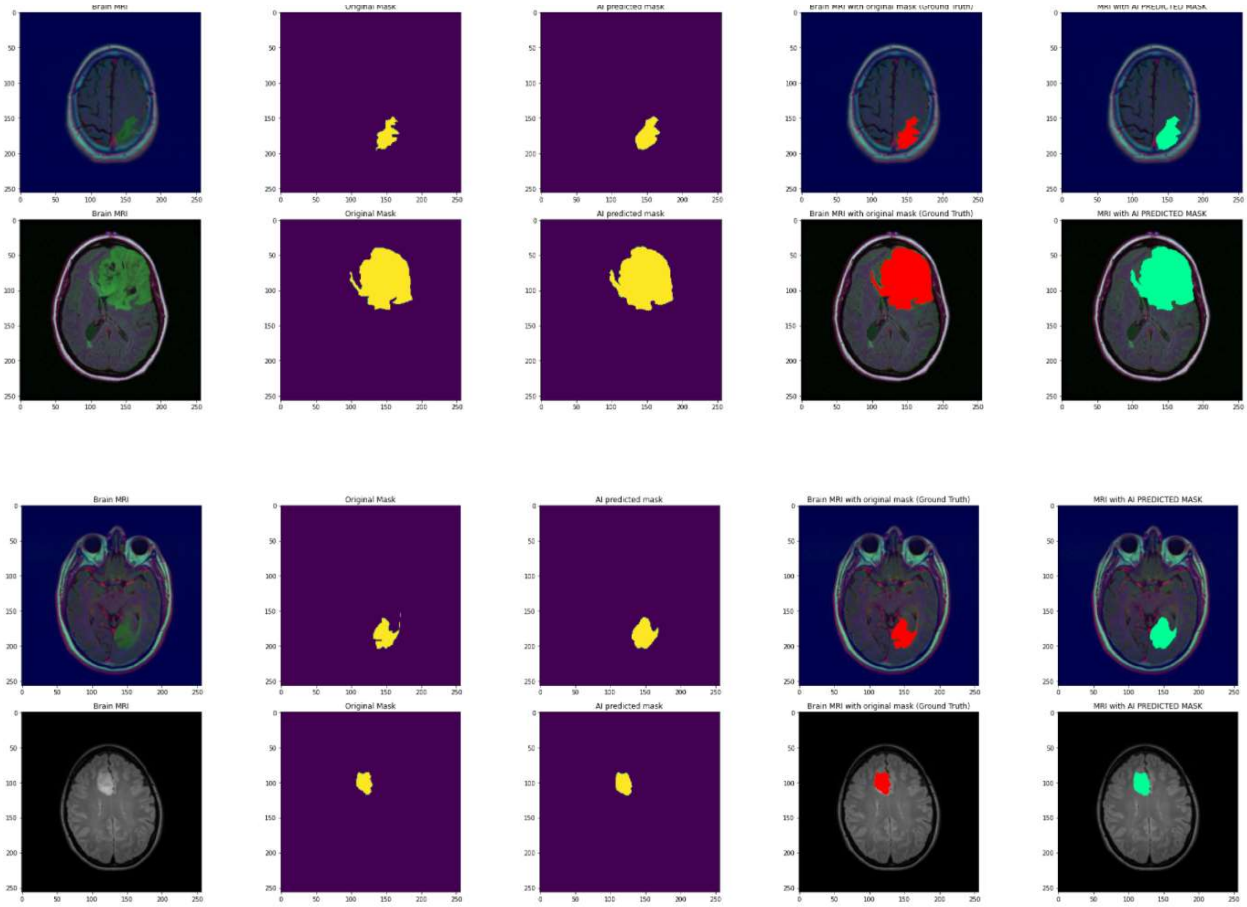
```
37/37 [==============================] - 2s 56ms/step - loss: 0.2665 - accuracy: 0.9458
Test accuracy : 94.57626938819885 %
```

```
test_ids = list(X_test.image_path)
test_mask = list(X_test.mask_path)
test_data = DataGenerator(test_ids, test_mask)
_, tv = seg_model.evaluate(test_data)
print("Segmentation tversky is {:.2f}%".format(tv*100))
```

```
6/6 [==============================] - 1s 88ms/step - loss: 0.1709 - tversky: 0.9048
Segmentation tversky is 90.48%
```

Upon further plotting the predictions over the dataset. Following are some of the results:





Based on the results generated, It Is evident that the U-net based architectures are far superior than the traditional CNN models and accurately segment the tumor locations based on the provided MR images. Although some masks are a bit blunt, the model can be further evaluated with increased epochs to see if it betters the results.

V. Contributions and Conclusion

This was a very complex project and it gave me an enhanced sense of how to apply machine learning for the health industry. Learning about the different parameters concerning the medical fields and tumors in particular was extremely mind boggling but interesting at the same time.

The algorithm is deterministic, which implies that it will always do the same assessment given the same image. Finally, using a computer algorithm is both economical and quick. In terms of the mean Tversky, our segmentation system performed at 90 percent, putting it on par with skilled human readers.

Even though I faced many time-consuming challenges such as operational error or matrix size related errors or learning about how multiple iterations can help deep learning models to improvise further, I have learnt a lot in this project.

In terms of contributions, I have done the following:

- Created the notebook, studied dependencies and core solution to develop the project. https://www.kaggle.com/mateuszbuda/brain-segmentation-pytorch

- Implemented the Tversky loss function over the dice co-efficient. https://github.com/nabsabraham/focal-tversky-unet

- Used the ResNet50 model and played around with the number of iterations (epochs) to see how I can achieve a good accuracy with minimal number of iterations. For me 30 seemed to be a good count.

- Performed advanced visualizations to give a presentable result overview for multiple aspects of the project.

In future works for this project, I would like to explore advanced U-net architectures such as AlexNet which have an extremely high accuracy rate for the segmentation tasks, but due to its requirements of powerful computational environment it is time consuming. I would also like to work towards creating a model which can detect any kind of abnormality or tumors in the body.

VI. References & Citation

[1] Maciej A Mazurowski, Mateusz Buda, Ashirbani Saha, Maciej A. Mazurowski: 'Association of genomic subtypes of lower-grade gliomas with shape features automatically extracted by a deep learning algorithm'. *Computers in Biology and Medicine,* 9 Jun 2019.

[2] Maciej A. Mazurowski, Kal Clark1, Nicholas M, Czarnek, Parisa Shamsesfandabadi, Katherine B. Peters, Ashirbani Saha: "Radiogenomics of lower-grade glioma: algorithmically-assessed tumor shape is associated with tumor genomic subtypes and patient outcomes in a multi-institutional study with The Cancer Genome Atlas data". *Journal of Neuro-Oncology*, May 2017.

[3] P Gokila Brindha, M Kavinraj, P Manivasakam and P Prasanth :"Brain tumor detection from MRI images using deep learning techniques*". IOP Conference Series: Materials Science and Engineering,* 2020.

[4] Hongkai Wang, Zongwei Zhou, Yingci Li, Zhonghua Chen, Peiou Lu, Wenzhi Wang, Wanyu Liu and Lijuan Yu: "Comparison of machine learning methods for classifying mediastinal lymph node metastasis of non-small cell lung cancer from 18F-FDG PET/CT images". *EJNMMI Research,* 2017

[5] Siyuan Lu, Shui-Hua Wang, Yu-Dong Zhang a,b : "Detecting pathological brain via ResNet and randomized neural networks". *CellPress Heliyon 6, 2020*

[6] https://aditi-mittal.medium.com/introduction-to-u-net-and-res-net-for-image-segmentation-9afcb432ee2f