Semantic Segmentation of Brain Tumor using Deep Neural Network – UNet

Anudeep Balagam
abalagam@buffalo.edu
50442091

1. Introduction

A crucial topic in the domains of computer vision and image processing is semantic segmentation. I think I could use this technology to diagnose and segment brain tumors. Early diagnosis and treatment planning may be able to save the patient`s life. This project`s main objective is to apply the U-Net neural network model to a collection of FLAIR pictures from the Cancer Imaging Archive, where this sort of image eliminates signal from the cerebrospinal fluid to produce the final images. An effective convolutional neural network for biological picture segmentation is the UNet.

2. Data:

The dataset I used is in Kaggle: https://www.kaggle.com/datasets/mateuszbuda/lgg-mri-segmentation .

The collection includes manual FLAIR abnormality segmentation masks and 3929 brain MRI images. These images correspond to 110 patients included in The Cancer Genome Atlas (TCGA) lower-grade glioma collection with at least fluid-attenuated inversion recovery (FLAIR) sequence and genomic cluster data which we will not using for this project.
Also, some brain tumors are visible on brain MRIs while others are not. Each of the brain MRI images in the dataset is a 256 x 256 RGB image and each mask is a 256 x 256 Boolean matrix with each index indicating 1 or 0.

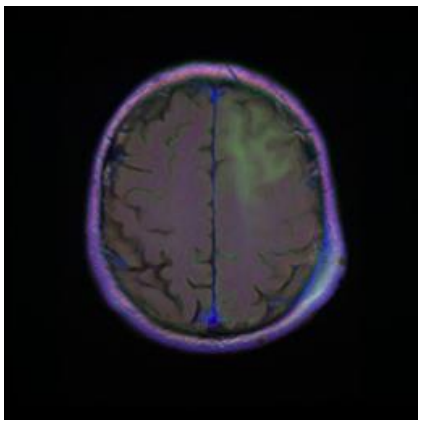


Figure: 1(a)

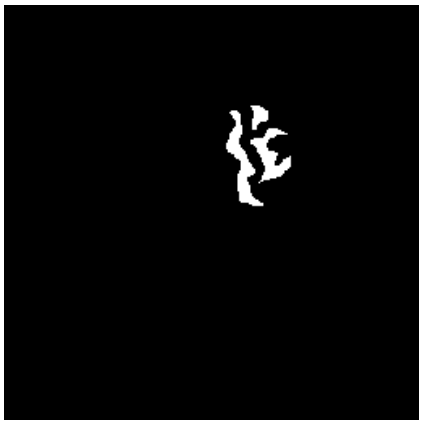


Figure: 1(b)

Figure: 1(a) is a brain MR image and Figure:2(b) is the mask where the white part represents the brain tumor that will be indicated as 1 in the Boolean matrix.

3. Data Pre-Processing and Augmentation:

For the pre-processing step the RGB images are converted into gray scale images so as to eliminate the complexity, and also gray scale images have only one channel that is between 0 to 255. The below figures: 2(a) and 2(b) show the RGB image and gray scale image respectively.

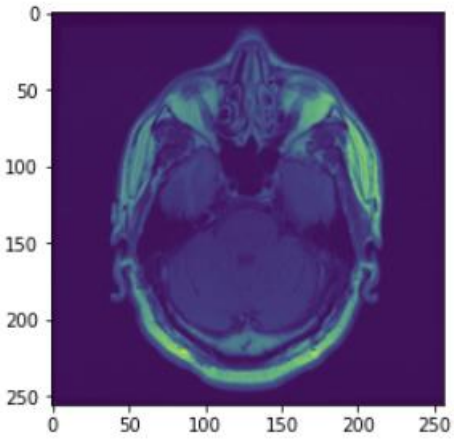


Figure: 2(a)

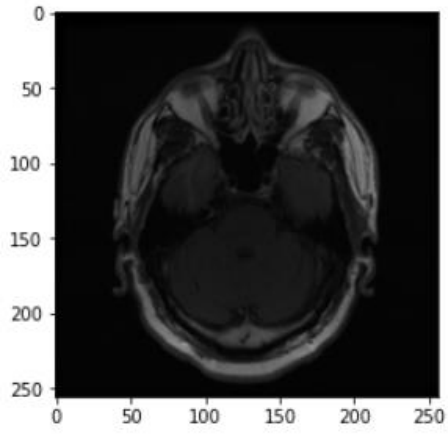


Figure: 2(b)

To increase accuracy and avoid overfitting, neural networks need many images and training masks. The 110 patients in our data correlate with 3929 images and labels. Applying an image processing methodology to data augmentation, in which transformations are performed on an existing training set to produce additional images and training labels, is one way of dealing with this problem.

Operations	Parameters
Horizontal Flip	True
Vertical Flip	True
Rotation	20 degrees
Shift	0.02 vertical and horizontal direction
Shear	5

4. Method

a. UNet:

The method I used to detect brain tumors is U-Net Network and referred to the paper named "U-Net: Convolutional Networks for Biomedical Image Segmentation" by Olaf Ronneberger, Philipp Fischer, and Thomas Brox.

UNet, was developed from the conventional convolutional neural network, was created, and used for the first time in 2015 to process images used in biomedicine. A standard convolutional neural network focuses on classifying images, with an input of an image and an output of a single label. However, in biomedical applications, it is necessary to identify both the presence of a disease and the location of the abnormality, hence UNet is dedicated to solving this problem. UNet localizes the borders by doing classification on every pixel by which input, and output can share the same size.

b. Architecture:

Figure: 3 represent the architecture of UNet as appears to be in the shape of a "U." The design is symmetrical and is divided into two main sections: the left section is known as the contracting path and is made up of the basic convolutional process; the right section is known as the expansive path and is made up of transposed 2D convolutional layers it acts as upsampling technic.

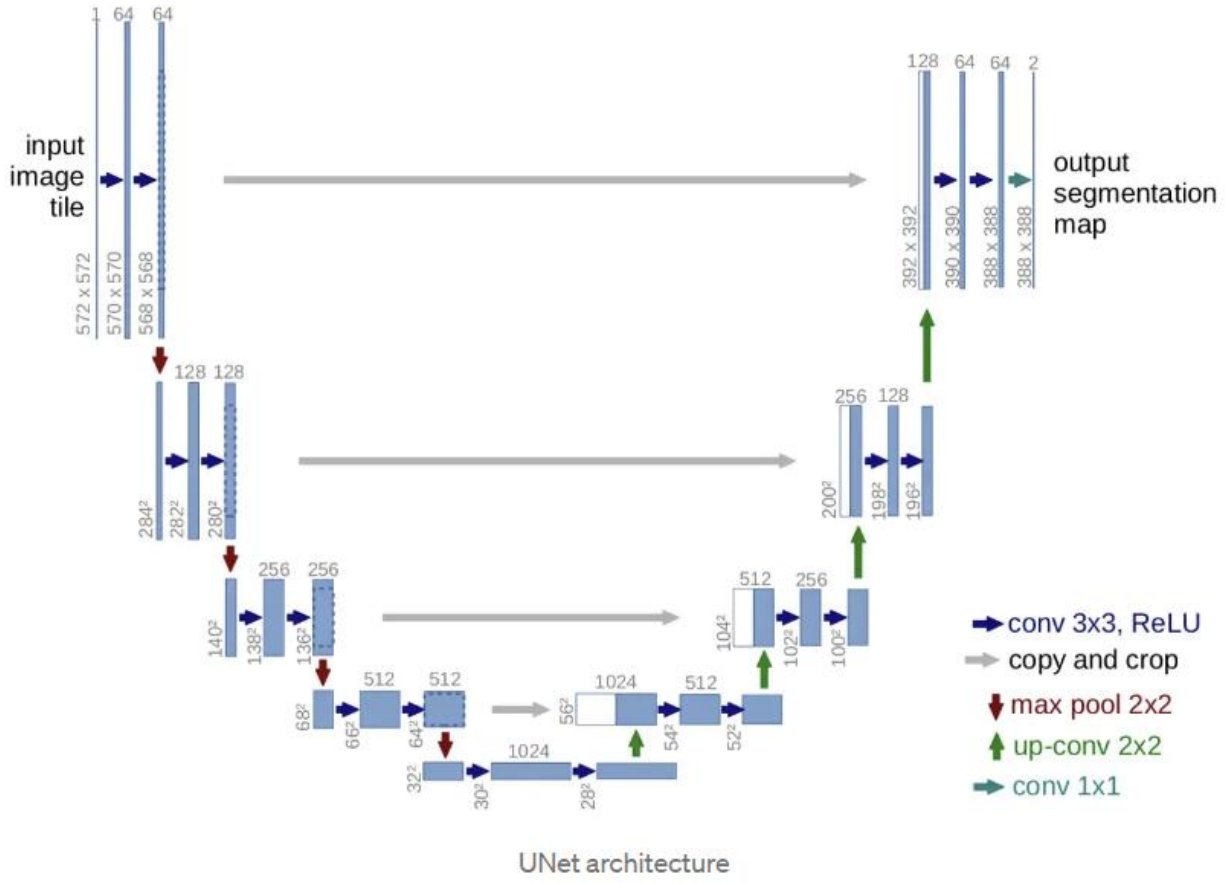


Figure: 3

The architecture is divided into two parts. The left side of the architecture represents the contracting path, and the right side represents the expansive approach. The number of features would double every time the contracting path was used, and this process is known as downsampling. The expansive path after the downsampling would result in an upsampling that would cut the number of features in half.

c. Code of UNet

```
def build_model(input_layer, start_neurons):
    conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(input_layer)
    conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(conv1)
    pool1 = MaxPooling2D((2, 2))(conv1)
    pool1 = Dropout(0.25)(pool1)

    conv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(pool1)
    conv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(conv2)
    pool2 = MaxPooling2D((2, 2))(conv2)
    pool2 = Dropout(0.5)(pool2)

    conv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(pool2)
    conv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(conv3)
    pool3 = MaxPooling2D((2, 2))(conv3)
    pool3 = Dropout(0.5)(pool3)

    conv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(pool3)
    conv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(conv4)
    pool4 = MaxPooling2D((2, 2))(conv4)
    pool4 = Dropout(0.5)(pool4)

    # Middle
    convm = Conv2D(start_neurons * 16, (3, 3), activation="relu", padding="same")(pool4)
    convm = Conv2D(start_neurons * 16, (3, 3), activation="relu", padding="same")(convm)

    deconv4 = Conv2DTranspose(start_neurons * 8, (3, 3), strides=(2, 2), padding="same")(convm)
    uconv4 = concatenate([deconv4, conv4])
    uconv4 = Dropout(0.5)(uconv4)
    uconv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(uconv4)
    uconv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(uconv4)

    deconv3 = Conv2DTranspose(start_neurons * 4, (3, 3), strides=(2, 2), padding="same")(uconv4)
    uconv3 = concatenate([deconv3, conv3])
    uconv3 = Dropout(0.5)(uconv3)
    uconv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(uconv3)
    uconv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(uconv3)

    deconv2 = Conv2DTranspose(start_neurons * 2, (3, 3), strides=(2, 2), padding="same")(uconv3)
    uconv2 = concatenate([deconv2, conv2])
    uconv2 = Dropout(0.5)(uconv2)
    uconv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(uconv2)
    uconv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(uconv2)

    deconv1 = Conv2DTranspose(start_neurons * 1, (3, 3), strides=(2, 2), padding="same")(uconv2)
    uconv1 = concatenate([deconv1, conv1])
    uconv1 = Dropout(0.5)(uconv1)
    uconv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(uconv1)
    uconv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(uconv1)

    output_layer = Conv2D(1, (1,1), padding="same", activation="sigmoid")(uconv1)

    return output_layer

input_layer = Input((img_size_target, img_size_target, 1))
output_layer = build_model(input_layer, 16)
```

5. Experiments

a. Experiment 1:

Applying Dice loss function on the training sample for the UNet. The [0,1] value range of the dice coefficient is a predefined similarity measure function that is typically used to compare two samples. Dice Loss is 1 - Dicecoefficient, hence it will be less if two samples are more similar.

$$Dice_Loss(A, B) = 1 - \frac{2|A \cap B| + Smooth}{|A| |+ |B| + Smooth}$$

Dicecoefficient is 2 * area of overlap divided by the total number of pixels in both images.

b. Experiment 2:

Applying Binary Cross Entropy(BCE) on the training samples for the same UNet. BCE compares each of the predicted probabilities to actual class output which can be either 0 or 1. The result we need is a semantic segmentation the shape of the tumor, BCE is suitable for this as it helps is identifying whether the tumor is in certain pixel.

$$BCE_Loss = \sum_x -(T_x \log(P_x) + (1 - T_x) \log(1 - P_x))$$

c. Experiment 3:

Applying IOU Loss Function on the training samples for the Neural Network. IoU stands for Intersection over Union – a metric used to evaluate Neural Network algorithms by estimating how well a predicted mask or bounding box matches the ground truth.

$$IOU_Loss(A, B) = 1 - \frac{A \cap B + Smooth}{A \cup B + Smooth}$$

Smooth is a value defined by us, by default it is 1. A is the predicted value and B is the true value of the mask.

d. Model Performance in Semantic Segmentation:

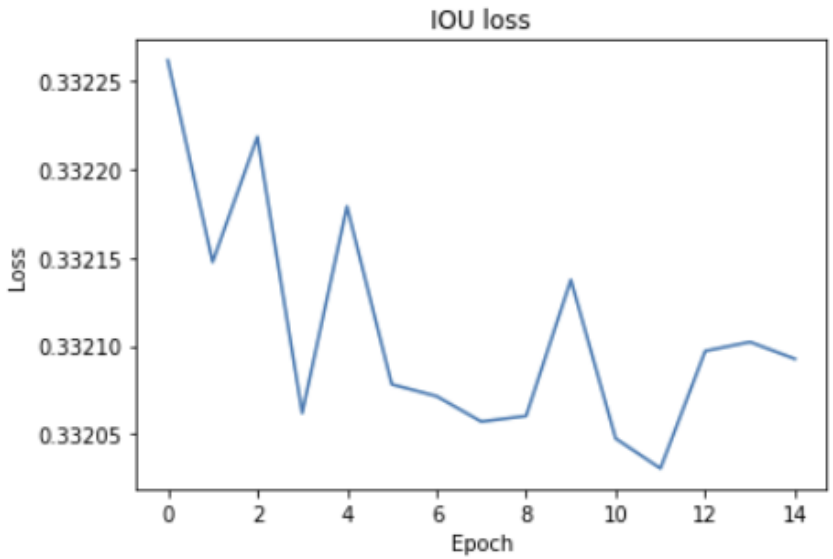
We use Mean Intersection over Union (Miou) to determine the model performance in semantic segmentation. This is calculated using the below formula:

$$MIoU = \frac{1}{k+1} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}}$$

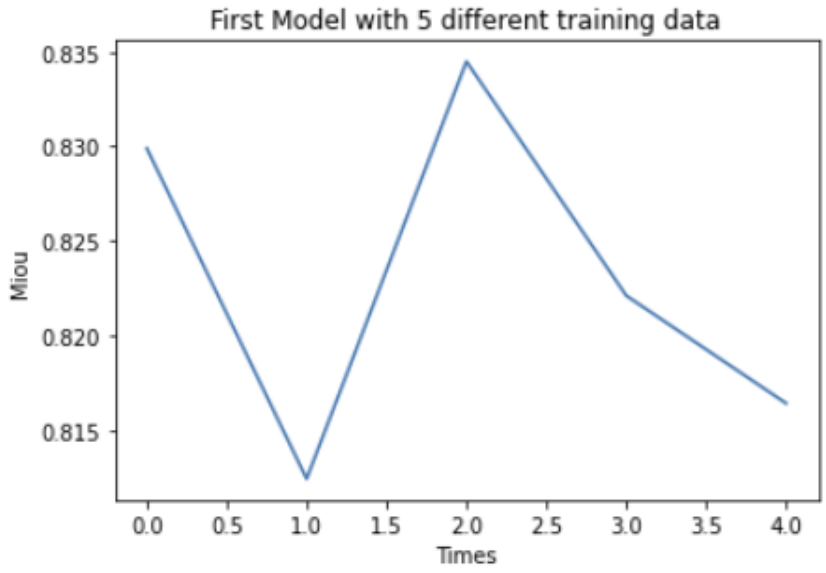
This formula averages the IOUs for all the categories.

6. Graphical Result:

a. Experiment 1 Result:

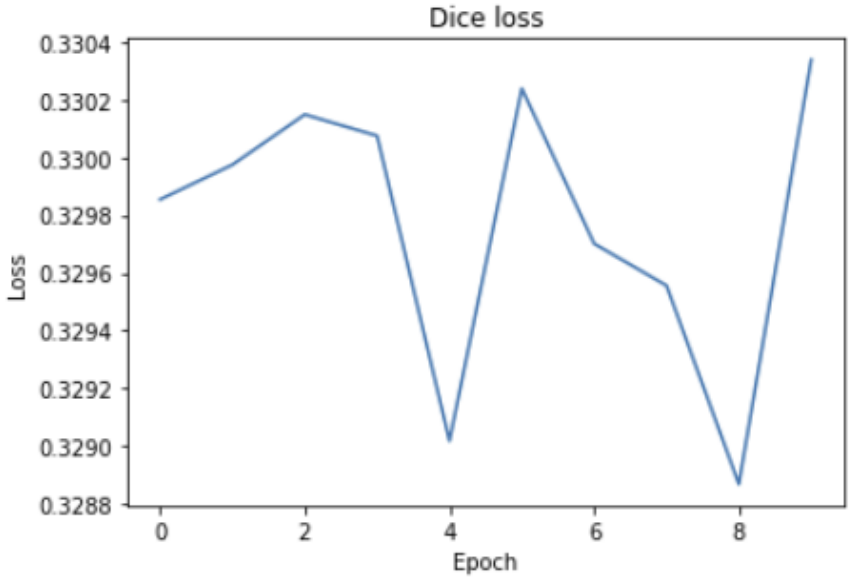


The IOU loss graph shows a drop in 15 epoch and starting loss value is less. I think it is the features of our dataset that most areas in the mask are black and even some mask that without tumor is all black and it will make the loss value very small.

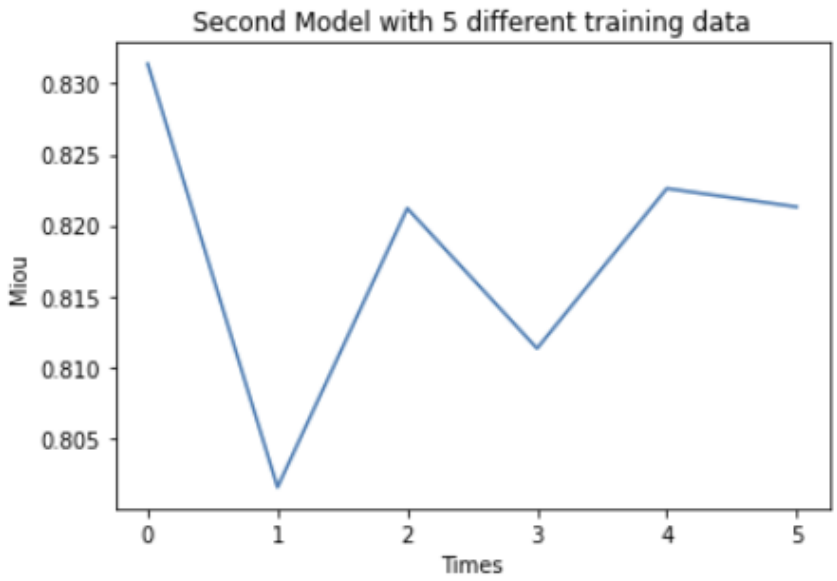


The Mean intersection over union(Miou) score of the model IOU is between 81% to 83% and the accuracy of this model is 82.8%

b. Experiment 2 Result:

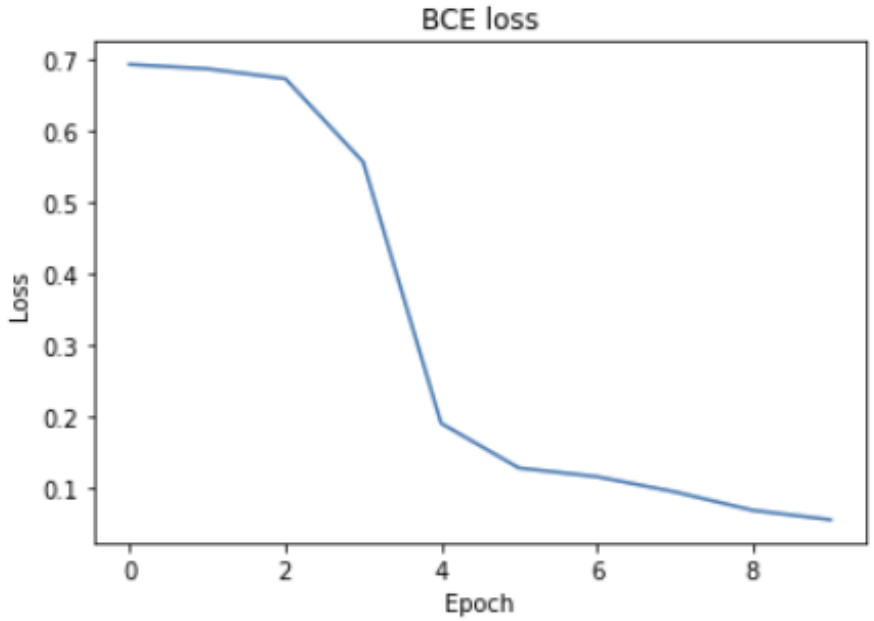


The Dice loss graph shows a drop in 3rd epoch and starting loss value is less. We can also see that the loss rice after 4th epoch but there is a decrease in the loss after 6th epoch.

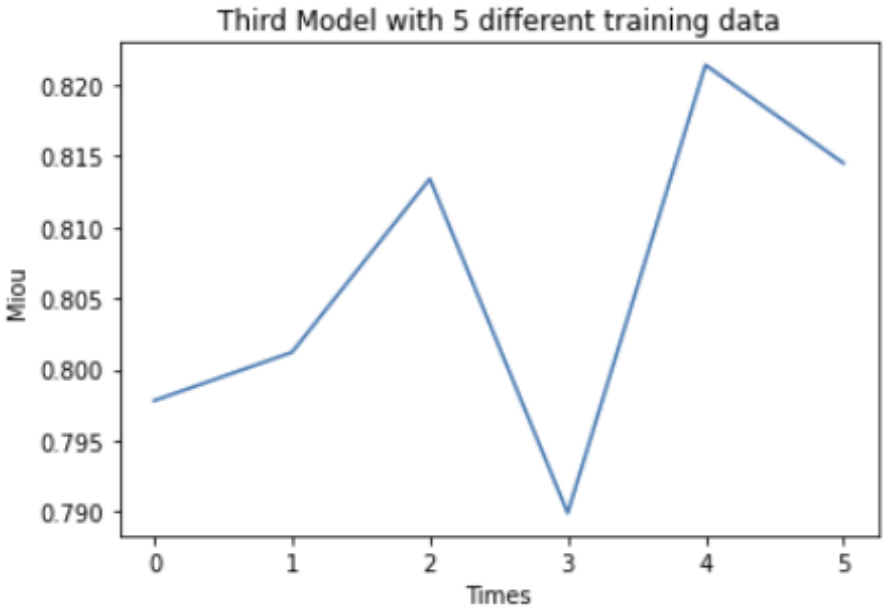


The Mean intersection over union(Miou) score of the model IOU is between 80% to 83% and the accuracy of this model is 82.1%

c. Experiment 3 Result:



The BCE loss graph shows a drop after 3rd epoch and starting loss value is high. The loss function shows a downward trend, indicating the model has a potential to rise.



The Mean intersection over union(Miou) score of the model IOU is between 79% to 82% and the accuracy of this model is 80.2%

7. Final Solution:

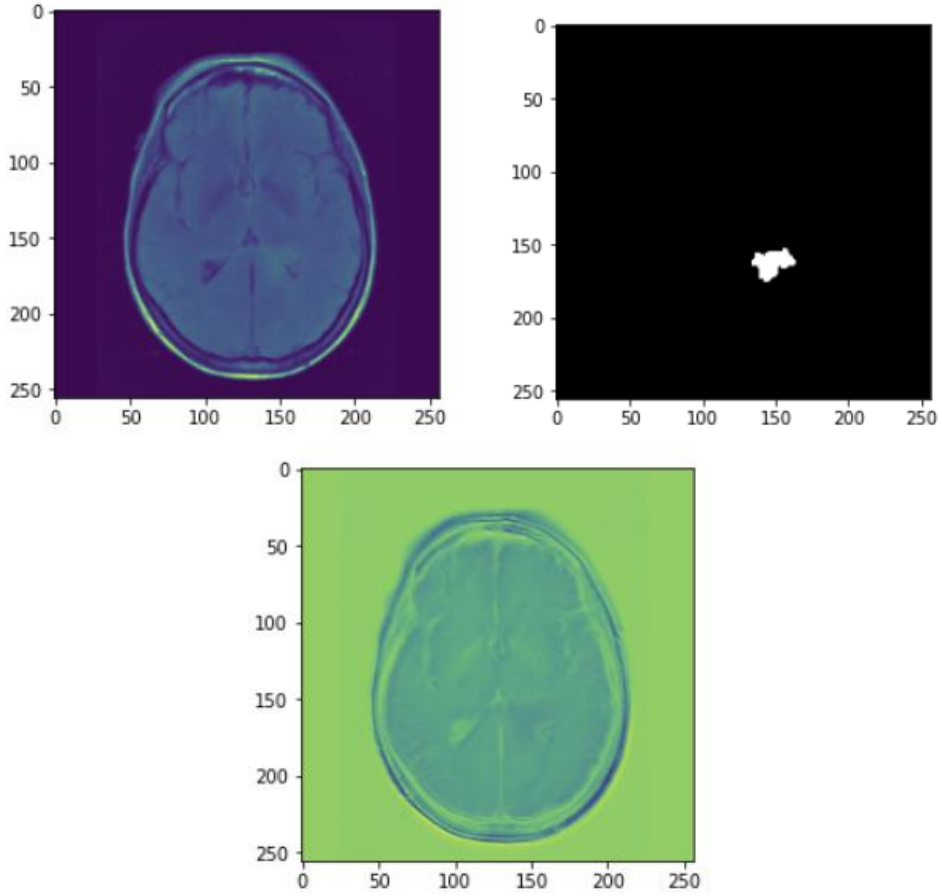
a. Code of the solution:

From the results we can see that in the brain tumor MRI segmentation the performance of Dice loss and IOU loss are similar around 82%, and binary cross entropy loss is 80.2% under the same model structure, training data and testing data.
So according to our experiments IOU model is a good choice to implement.

```
from keras.callbacks import EarlyStopping
train_generator = get_augmented(X_train,Y_train)

model1 = iou_model()
earlystopper = EarlyStopping(patience=5, verbose=1)
checkpointer = ModelCheckpoint('IOUModel.h5', verbose=1, save_best_only=True)
```

b. Graphical Results:



IOU Loss Original vs Predicted

8. Conclusion

Based on the dataset I used which contains 3929 brain MR images with manual FLAIR abnormality segmentation mass, I enhanced the efficiency and accuracy to the segmentation. Not only did I improve the accuracy but also used the data augmentation technique to increase the robustness of the model. I performed three experiments and each experiments have its own advantages and disadvantages. Among all the experiments I have done I observe that IOU Loss is better suited compared to Dice and BCE loss. I hope this experimentation gives a verification on tumor contouring and also, I hope this can be used for other medical analysis.

9. References:

1. Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation" , Computer Science Department and BIOSS Centre for Biological Signaling Studies.
2. Luis Perez, Jason Wang, “The Effectiveness of Data Augmentation in Image Classification using Deep Learning”.
3. Adam Abdulhamid, ”Semantic Segmentation with Histological Image Data: Cancer Cell vs. Stroma”
4. D. Godoy, understanding binary cross-entropy / log loss: a visual explanation, 2019 (accessed May 10, 2020).

Code: Google Colab Link –

https://colab.research.google.com/drive/1bjRQs889GJlV_tplkKYiPLona9EflxBF?usp=sharing