Brain Tumor Classification and Segmentation

Soham Sheth[#1], Vrushali Shinde[*2]

[#]*College of Computing, Michigan Technological University*
Houghton, Michigan, United States-49931

[1]`ssheth1@mtu.edu`

[2]`vshinde2@mtu.edu`

Abstract— **This brief project report is based on classification and segmentation of brain tumors. The project experiments on a public dataset. The dataset is used for classifying the dataset and then further segmenting the same. The method used for classification is value calculation. If the diagnosis value is greater than zero, then it is tumor brain otherwise no and for segmentation, Unet model is used. The main reason behind selecting the Unet model is its flexibility, and encoder-decoder architecture where encoder part obtains contextual information from the input images and decoder decodes regenerates the segmentation map. The Unet model is also known for capturing fine details.**
The accuracy in classification is determined by performance metrics "Accuracy" and for Unet is "iou" and "dice coefficient".
Note: The project was initially proposed for doing just brain tumor classification but however due to scope and the outer scope that segmentation is needed after classification for making it more relatable in the world, the project proceeded with later parts as well.

Keywords*— *Unet, dice-coefficient, iou, loss, encoder, decoder, convolutional block, attention block.

I. Introduction

Brain tumor is one of the most significant and damage causing diseases in the world. There are various types of it but most importantly if a person has a brain tumor, then they go through stages of severity. If the condition worsens then no operation or chemo can make the health good. Perhaps identifying the brain tumor at an early stage really plays an important role in the medical field. Another hiccup that is very important to address when detecting the brain tumor is that the size and shape of the tumor is not fixed. Hence, it is crucial to identify whether it is a tumor or just a mere abnormality in the brain. However, if due to segmentation with the help of medical scan imaging, if we correctly and timely detect the tumor, it can be cured early. The scope of the project includes classifying the MRI images of the brain and then mapping a contextual information from the input image and then detecting the shape of the tumor and exactly where it is present in the brain. The process used in this project is timely run in GPU mode of the computational device and availability of data in the form of downloading the existing dataset from kaggle which includes multiple MRI images of brains. The project includes preprocessing of the input images (resizing, normalizing) mainly and then applying a Unet model on the same with and without attention. Further, the report will contain the methodology, dataset information, considered performance metrics information and then detailed Unet explanation and the results and discussion on the same. The motivation for initiating this project is pre-knowledge of Unet in machine learning and then the concerned real-world challenge called brain tumor. The project model aims to achieve segmentation with utmost accuracy without overfitting or underfitting and then putting forth the thought of using the same in some way in the real world.

II. Related Work

Below are the few resources which were referred for project reference. These are the publications which have related work for the project which was made.

[1] This is the original paper that introduced the UNet architecture for biomedical image segmentation, including brain tumor segmentation. It describes the architecture in detail and provides insights into its design choices and performance on different tasks, including brain tumor segmentation. ("U-Net: Convolutional Networks for Biomedical Image Segmentation")

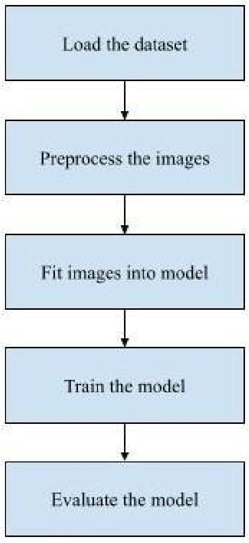
[2] This paper proposes a modified UNet architecture for brain tumor segmentation, incorporating additional skip connections and dense connections to improve segmentation accuracy. The authors evaluated their approach on the BraTS (Brain Tumor Segmentation) challenge dataset and achieved competitive results. (S. Pereira, A. Pinto, V. Alves and C. A. Silva, "Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images," in IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1240-1251, May 2016, doi: 10.1109/TMI.2016.2538465.)

[3]This paper proposes an attention mechanism for the UNet architecture to selectively attend to informative regions in medical images, specifically for pancreas segmentation. Although not directly focused on brain tumor segmentation,

the attention mechanism may be relevant for brain tumor segmentation tasks as well. ("Attention U-Net: Learning Where to Look for the Pancreas")
The above works are original and cited in this paper to let know that the idea behind these contributed to making this project and all the original idea credits goes to their respective authors.

III. Methodology

Workflow of proposed model



The u-net technique with and without attention is same expect addition of attention block and using adam optimizer

II. A. *Data Acquisition*
The dataset used in the project consists of brain MRI's along with segmentation masks. This dataset is publicly available in kaggle. The data is of 110 patients having lower glioma. The extension of this image file is ".tif" which is lossless compressed image with highest detailed quality and stands for Tagged Image File Format.

II. B. *Source of the Dataset*
The dataset used in this project is obtained from ("Brain MRI segmentation")

II. C. *Data Preprocessing*
The data preprocessing involves substeps like data augmentation, resizing, normalization. The data resizing includes resizing the image into dimension 256 X 256. Data normalization includes representation of the data into 0's and 1's. This data augmentation is used in terms of creating a batch of normalized images and masks so that we can provide these batches as input to the Unet model. The image generator is used for the same. The scale is grayscale for masked images and rgb for normal ones and the batch size is fixed to 32. The seed value is kept fixed so to make sure that when data is augmented, it should be deterministic and not change for any number of runs of code.

II. D. *Data Splitting*
The data splitting is carried out with the help of the train_test_split function in machine learning python and for this data is splitted in 80-20% i.e. 80% training and 20% testing. However, to note that 10% of the training data is considered for validation. Hence, we can say distribution as 70-10-20% in terms of training, validation and testing respectively. The random state is kept fixed to 42 to ensure that every time the same split happens when we run the code. This helps to minimize the inconsistency. The data is only concerned with the glioma type of tumor.

II. E. *Data Usage*
The dataset in this experiment is used to train a convolutional neural network (CNN) for brain tumor segmentation using a U-Net technique. The trained model is then evaluated on the testing part to observe the performance and analyze whether the model is a good machine learning model or not.

IV. Performance Metrics Used In The Project

The two main performance metrics used in this project are Dice coefficient and Intersection Over Union (IoU). By definition, dice coefficient is used for similarity purposes between predicted and ground tumor region. The dice coefficient being 1 indicates that there is a perfect match between the predicted and ground truth masks and zero being the bad extreme. The IoU which is commonly called as Jaccard Index is also used for similarity. The higher the value of IoU indicates the more similarity. Naturally its value lies between 0 to 1. The losses of both these metrics are calculated by subtracting these respective from one.
The performance metrics are in general used for evaluating the performance of the trained model. In the case of the project, higher the value of these, indicating that much more efficient is the trained model and that the predicted tumor region image is closely matching the ground truth image.

$$Iou = \frac{Intersection(TP)}{Union(TP+FP+FN)}$$

$$DiceCoefficient = \frac{2*Intersection(TP)}{Intersection + Union(T\,P + F\,P + F\,N)}$$

$$Losses\ are\ simply\ calculated\ as\ 1 - \ respective\ metrics$$

V. U-Net : The Segmentation Architecture

In U-Net, the encoder part is the downsampling part and the decoder part is the upsampling part.

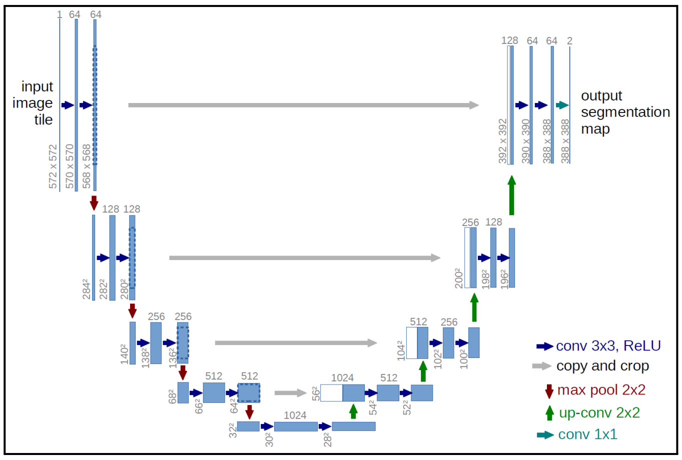


Fig. 1 U-Net Architecture

Feature is mapped from the encoder and concatenated at the decoder through skip connection. At every step, the architecture uses the output of the convolutional layer and then uses it for the further part.

The steps involved in the u-net implementation are as follows:

1. First operation is 2 convolutional layers. The number of channels changes from 1 to 64 and therefore the convolutional process will increase the depth of image and red arrow pointing down max pooling which will half the size of image. Also, changes of dimensions of image are due to padding issues. So in u-net the padding is set to "same".
2. Same operation is repeated three more times to downsample and then the decoding part starts.
3. In this, the image is resized to its original size.
4. And these are also convolutional layers, so we use transpose convolutional layers.

For Implementation:

The first block is the basic convolutional block which uses the Relu activation block followed by batch normalization. It takes input tensor along with filters and then returns the output tensor.

The second block defined is the encoder block. This is the initial block which is used for downsampling the images and with each layer, the architecture digs deep into the image.

It uses max pooling to reduce the size of the image and focus only on the necessary part.

The number of filters passed to this encoder block are 64, then 126, then 256 and lastly 512.

The third block is a decoder block which consists of a transposed convolutional layer. It takes input tensor, filters and then skip connection tensor as argument and returns the output tensor.

Again the number of times this block will be called depends on the number of filters passed in the encoder.

The fourth block is the attention block which uses activation and padding and then this block is mainly used so that the network dynamically gathers the spatial information and vital information in the feature maps ignoring the other one.

The fifth block is decoder block with attention is used as a way of combining the task of decoder block and attention together. The decoder block is responsible for upsampling the image to its original resolution to display results along with an attention mechanism surpassing the necessary and emphasizing on the necessary features only.

The project tries to run the u-net technique into one with attention and one without and then use optimizer on the same.

For this, it has used adam optimizer (adaptive moment estimation) which is a proper optimization algorithm which uses the given learning rate to learn from past information of convolutional layers and then tries to optimize the convolutional layers. This technique too contains the same blocks with optimization parameters extra. The optimizer updates the model's parameters during training during each epoch and then tries to minimize the loss value and improve the model performance.

VI. Experimental Results

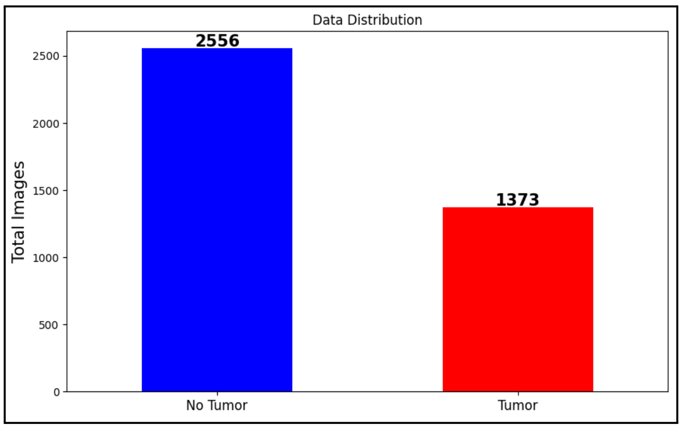


Fig. 2 Classification into yes and no tumor

The above figure illustrates the output generated by code when we downloaded the dataset and classified the images as the ones having tumor and ones with no tumor based on the diagnosis column value which was calculated based on mask values

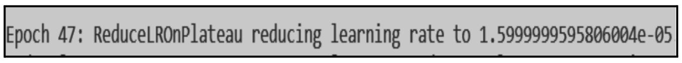
If the diagnosis column has value 0 then the image is the one without tumor.
If the diagnosis column has value 1 then the image is the one with tumor.

The project code implements the u-net model with and without attention parameters. Further the model architectures are stored in png format which will be attached in the deliverables and then the model is stored in .h5 file which ideally contains the architecture, weights, concentration and other needed configuration settings which can be used to recreate the model later. Then these models are trained and evaluated to analyze the results.
The callback function is used which has parameters like ModelCheckpoint which makes sure that only best trained weights are stored in the h5 file, ReduceLROnPlateau which ensure that learning rate is reduced at the point where we see no decrease in loss value and EarlyStopping is used which makes sure that when all the hyperparameters that were provided to improve the performance of model are no longer effective, the best weights of the model is restored.

Constant values are:

```
IMAGE_SIZE = (256, 256)
EPOCHS = 60
BATCH_SIZE = 32
learning_rate = 1e-4
```

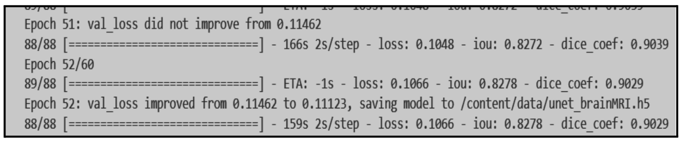
U-Net without Attention:



```
Epoch 47: ReduceLROnPlateau reducing learning rate to 1.5999999595806004e-05
```

Fig. 3 ReduceLROnPlateau working

As we can see in the fig. 3, when val-loss is not improving itself, the ReduceLROnPlateau adjusts the learning rate so that model performance can be improved.



```
Epoch 51: val_loss did not improve from 0.11462
88/88 [==============================] - 166s 2s/step - loss: 0.1048 - iou: 0.8272 - dice_coef: 0.9039
Epoch 52/60
89/88 [==============================] - ETA: -1s - loss: 0.1066 - iou: 0.8278 - dice_coef: 0.9029
Epoch 52: val_loss improved from 0.11462 to 0.11123, saving model to /content/data/unet_brainMRI.h5
88/88 [==============================] - 159s 2s/step - loss: 0.1066 - iou: 0.8278 - dice_coef: 0.9029
```

Fig. 4 Epoch running sample

The above figure 4. is the epoch run in the project and as we can observe when the value of val_loss is not improved, accordingly the statement is printed and when it is improved, the model saves the latest into the h5 file.

Although the total number of epochs run for the project are 60, this was the last and latest observed val_loss improved. Hence, the h5 file will contain this model weight saved.

Further model fit and history run, the trend plots are plotted.

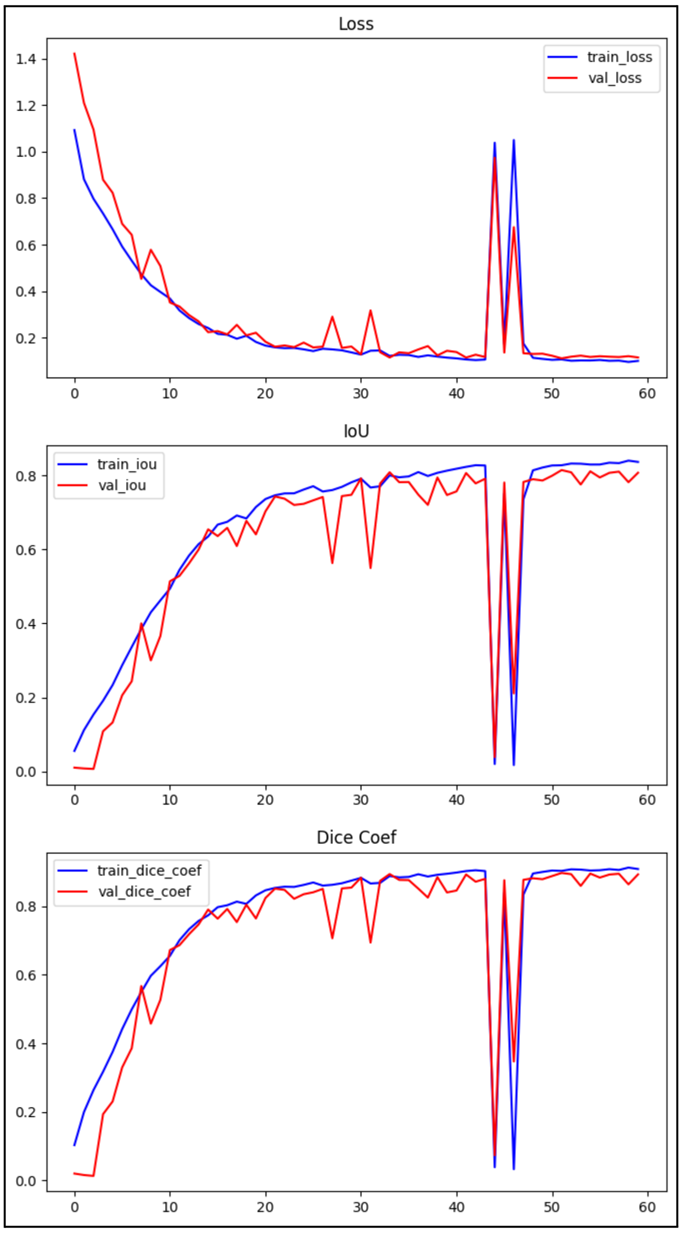


Fig. 5 Trend plots

The first subplot (top) shows the training and validation loss across epochs. Blue represents training loss and red represents validation loss. The legend is placed in the best place and the subplot title is set to "Loss".

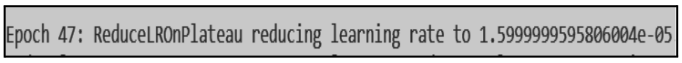
The second subplot (middle) shows the Intersection over Union (IoU) metric for training and validation across epochs.

Blue represents training IoU and red represents validation IoU. The legend is placed in the best place and the subplot title is set to "IoU".

The third subplot (below) shows the Dice coefficient (Dice coefficient) for training and validation across epochs. Blue represents Dice coefficient for training and red represents Dice coefficient for validation. The legend is placed in the best place and the subplot title is set to "Dice Coef".

As we can see from the figure 5. The loss generally decreases over increase in number of epochs and Intersection over Union (IoU) and dice coefficient increase over increase in number of epochs. However, there are certain spikes observed in all three plots. The reason is that when val_loss was not improving based on the optimizer, the ReduceLROnPlateau adjusted the learning rate and hence the trend continued as it was supposed to be. These spikes also indicate certain outliers which differ from the usual trend of the expected three(that would be the best reason according to authors)

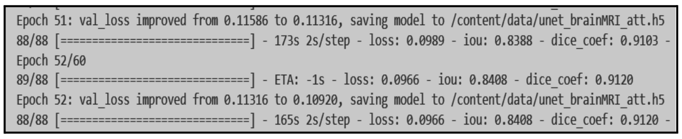
U-Net with Attention:



```
Epoch 47: ReduceLROnPlateau reducing learning rate to 1.5999999595806004e-05
```

Fig. 6 ReduceLROnPlateau working

As we can see in the fig. 6, when val-loss is not improving itself, the ReduceLROnPlateau adjusts the learning rate so that model performance can be improved.



```
Epoch 51: val_loss improved from 0.11586 to 0.11316, saving model to /content/data/unet_brainMRI_att.h5
88/88 [==============================] - 173s 2s/step - loss: 0.0989 - iou: 0.8388 - dice_coef: 0.9103 -
Epoch 52/60
89/88 [==============================] - ETA: -1s - loss: 0.0966 - iou: 0.8408 - dice_coef: 0.9120
Epoch 52: val_loss improved from 0.11316 to 0.10920, saving model to /content/data/unet_brainMRI_att.h5
88/88 [==============================] - 165s 2s/step - loss: 0.0966 - iou: 0.8408 - dice_coef: 0.9120 -
```

Fig. 7 Epoch running sample

The above figure 3. is the epoch run in the project and as we can observe when the value of val_loss is not improved, accordingly the statement is printed and when it is improved, the model saves the latest into the h5 file.
Although the total number of epochs run for the project are 60, this was the last and latest observed val_loss improved. Hence, the h5 file will contain this model weight saved.

Further model fit and history run, the trend plots are plotted.

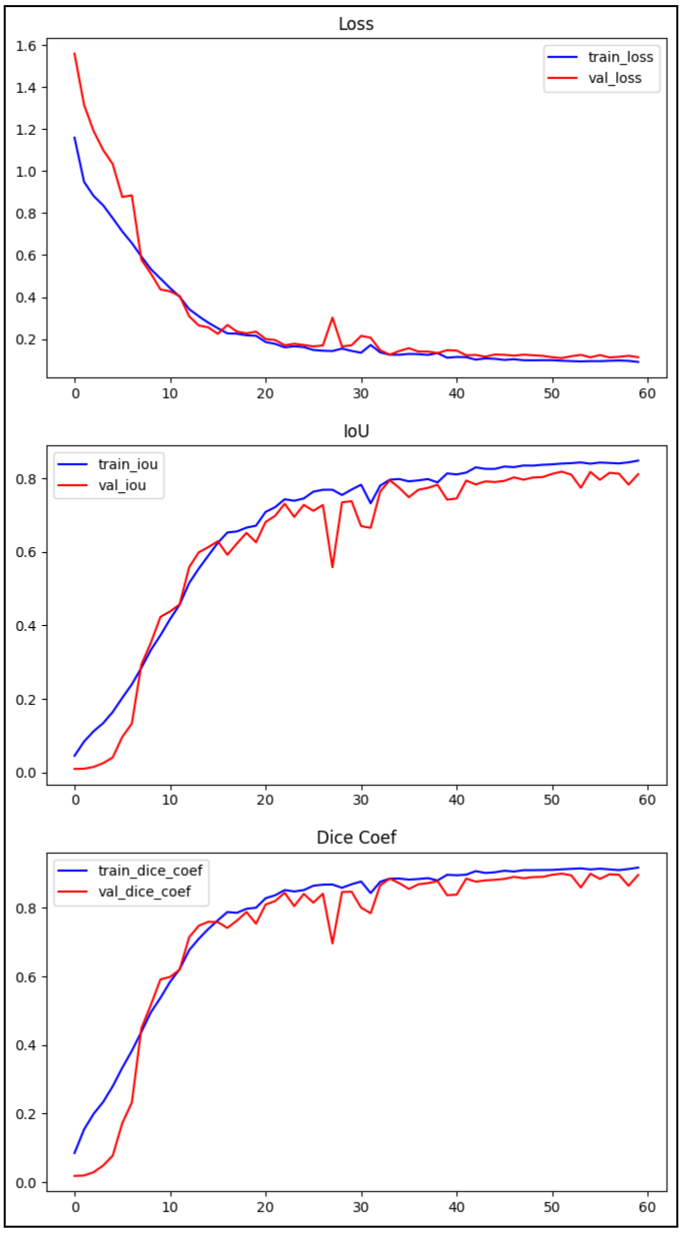


Fig. 8 Trend plots

The first subplot (top) shows the training and validation loss across epochs. Blue represents training loss and red represents validation loss. The legend is placed in the best place and the subplot title is set to "Loss".

The second subplot (middle) shows the Intersection over Union (IoU) metric for training and validation across epochs. Blue represents training IoU and red represents validation IoU. The legend is placed in the best place and the subplot title is set to "IoU".

The third subplot (below) shows the Dice coefficient (Dice coefficient) for training and validation across epochs. Blue represents Dice coefficient for training and red represents Dice coefficient for validation. The legend is placed in the best place and the subplot title is set to "Dice Coef".

As we can see from the figure 8. The loss generally decreases over increase in number of epochs and Intersection over Union (IoU) and dice coefficient increase over increase in number of epochs. Although there are certain minute spikes observed and the reason for that will be ReduceLROnPlateau adjusting the learning rate since the val_loss is not improving by itself on that same learning rate. Clearly, the trend of u-net with attention is better than the one without attention.

We can see from the earlier without attention and this trend plot that the one with attention behaves more theoretically well and the reason for that is the attention block addition. The attention mechanism in u-net helps the model to capture more relevant information and make the performance of the model more robust and more accurate. The below table has practically shown the one with attention has more accuracy than the other.

Comparison between the performance metrics of U-net model with Attention and Without Attention for 60 epochs:

Table 1: Performance Metrics Evaluation

Metric Name	**U-Net with attention**	**U-Net without attention**
IoU	0.8063067197799683	0.7959467172622681
Dice Coefficient	0.8919795155525208	0.8854154348373413
Val Loss	0.10920	0.11123

Although in figure 4 and figure 7 we can see that the individual epochs have different values than the one mentioned in the above table 1, this table represents the final IoU and dice coefficient after evaluating the model whose weights are loaded from the h5 file.

Explanation for performance metrics values:

The study and references show ideally the value of both iou and dice coefficient for any machine learning techniques or specifically for this brain tumor segmentation should be around ~88-89% or maybe even ~90%

However, the number of epochs run for such accuracy is around 150. Our results validate pretty much the dice coefficient since that is also considered as the accuracy measure.

Also, to note that with each epoch run, we can see certain numbers found to be validated image file names which means that these files meet the generator criteria and the more the number that being a good sign of performance of trained model.

After that the model.predict() function is used to predict the test dataset against the trained model.

10 sample results of these predictions are saved in a file which is shared with deliverables, however the small snippet is given below.

Prediction against model without attention:

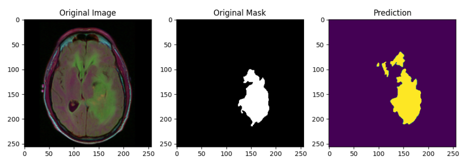


Fig. 9 Sample Prediction without attention block model

From the above figure, we can see that the brain having a tumor, the shape and best approximate position is visible in the prediction.

The purpose is hence fulfilled that the code has successfully segmented the tumor. On the other hand, the ones having no tumor appear to be black. Similarly,

Prediction against model with attention:

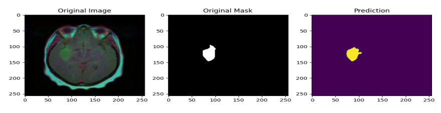


Fig. 10 Sample Prediction with attention block model

VII. Challenges

The primary challenge team members faced in the entire project was the time taken for the model to evaluate was huge and if the number of epochs were increased even on GPU for the given pc configuration it was difficult to compute. Also, the generated results seem fair (by clearly looking at the prediction images and dice_coefficient percentage)for the given epochs. This project is for academic purposes and team members thought of it being more of a learning process throughout.

VIII.Conclusion

The disease brain tumor is dangerous and can even cause death if not treated on time. The victims of the same desire to get cured on time and for that it is important to have the operation successful and for that segmentation model needs to give as much accurate data as possible. This project provides the comparison between u-net model with and without attention parameter and u-net is selected as the references did show u-net works better than ordinary cnn model. Further study of the same would have a huge impact in the medical field and improvements and enhancements should continue to yield better results. We also aim to explore more different kinds of 3D and 2D U-Net models.

Clearly, u-net with attention was more accurate than the one without attention by looking at Table 1.

IX. Achievements

This project was created for academic purposes and from the point of view of achieving certain goals.

1. Both team members learnt in depth about image processing.
2. Learned about deep learning algorithm concepts related to image processing and image segmentation model U-Net.
3. The idea behind implementing this project was to be aware of how important it is to clearly classify and locate where the tumor is situated in the brain in realtime to practically treat early and save life. That is achieved.

These primary goals were achieved.

X. Contribution

Both team members (Soham Sheth and Vrushali Shinde) actively participated in pair programming sessions throughout the project. reviewed and validated each other's work to ensure their correctness. The report writing process was a collaborative effort, with both members contributing equally to the final document.

XI. Acknowledgement

We would like to express our sincere gratitude to our Professor. Xiaoyong (Brian) Yuan for his mentorship and constant guidance throughout this project. His expertise and enthusiasm have been invaluable in helping us navigate the complexities of machine learning and the analysis of these algorithms. Additionally, we would like to acknowledge our fellow students in the class and both the TA's for their insightful suggestions and constructive feedback, which contributed to the quality and depth of this entire course.

XII. References

1. "U-Net: Convolutional Networks for Biomedical Image Segmentation." arXiv, 18 May 2015, https://arxiv.org/abs/1505.04597 Accessed 24 April 2023.

2. "U-Net Architecture Diagram." *YouTube*, https://tuatini.me/content/images/2017/09/u-net-architecture.png Accessed 23 April 2023.

3. "Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images." S. Pereira, A. Pinto, V. Alves and C. A. Silva, 29 July 2022, https://ieeexplore.ieee.org/document/7426413 . Accessed 24 April 2023

4. "Brain MRI segmentation." *Kaggle*, https://www.kaggle.com/datasets/mateuszbuda/lgg-mri-segmentation . Accessed 22 April 2023.

5. "Attention U-Net: Learning Where to Look for the Pancreas." *arXiv*, 11 April 2018, https://arxiv.org/abs/1804.03999 . Accessed 24 April 2023.