Brain MRI Tumor Detection Using UNet Model

1st Parshwa Shah
Student of Computer Science
Ahmedabad University
Ahmedabad, India
parshwa.s@ahduni.edu.in

2nd Yash Doshi
Student of Computer Science
Ahmedabad University
Ahmedabad, India
yash.d1@ahduni.edu.in

3rd Jay Mehta
Student of Computer Science
Ahmedabad University
Ahmedabad, India
jay.m@ahduni.edu.in

***Abstract*—The goal of brain tumour segmentation is to distinguish normal tissue from tumorous areas. To increase the chances of a successful treatment, this is an important stage in the diagnosis and treatment planning process. Magnetic resonance imaging (MRI) gives extensive information on brain tumour anatomy, making it a vital tool for accurate diagnosis. It is needed to replace the current manual detection technique, which relies on the skills and knowledge of a human. Using a U-Net-based deep learning model, a fully automatic system for segmenting malignant tumors in pre-operative MRI scans has been established. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The results show that our approach successfully segments every contrast in the data, performing slightly better than classical Bayesian segmentation, and three orders of magnitude faster. Moreover, even within the same type of MRI contrast, our strategy generalizes significantly better across datasets, compared to training using real images.**

***Index Terms*—Computer Vision, UNet Architecture, Brain MRI, Sementic Segmentation**

I. Introduction

A brain tumour is described as a malignant or noncancerous mass in the brain that develops aberrant cells.This area includes techniques such as tumour identification and classification, malignant region detection and classification, testing, and inspection of critical sections of the human brain. Magnetic Resonance Imaging (MRI) is used to diagnose brain tumours (MRI). Magnetic resonance segmentation is used in a difficult clinical imaging process to isolate the questionable regions. Although brain tumour diagnosis is still a manual process that is inspected and certified by specialists, MRI is the most often used process that highlights the tumorous region in the brain. As a result, we need a compelling procedure to detect brain cancers early on and save patients with high accuracy. For each required contrast, these methods frequently necessitate a significant number of manually segmented preprocessed photos.

II. Literature Survey

The primary research paper of our Brain MRI Segmentation was Classical generative model for Bayesian segmentation of brain MRI. In this method Kernels of size 3*3 are used in convolutions, and the Exponential Linear Unit is used as the activation function. Several hyperparameters regulate the priors of model parameters in the proposed generative model. We sample the hyperparameters of the GMM from vast ranges in an independent manner to achieve invariance to input contrast, which generally results in unrealistic images. This proposed research paper could be able to achieve upto 0.6 f-1 score. Another research paper Brain Tumor Segmentation of MRI Images Using Processed Image Driven U-Net Architecture which was better than previous one as it uses concepts of U-Net deep learning models for brain tumour image segmentation for detection were investigated in depth, and the conjectured performance evidence of U-Net deep learning models for brain tumour image segmentation for detection was proven. Experimented with standard U-Net segmentation, then looked into and incorporated subset division, category brain slicing, feature scaling, and narrow object region filtering before feeding to the U-Net model, and looked at how these pre-learning processes improved the performance of the U-Net architecture. It even examined some difficult failed tumour segmentation test instances in detail and discovered a significant amount of parameter adjustment.

III. Code Implementation

Firstly we imported all the necessary libraries for the project like NumPy, pandas, matplotlib, and TensorFlow. Then we differentiated all the images from the 'kaggle_3m' folder into the mri images with and without tumor and mask images with and without tumor. Later we shuffled all four with seed 1337 so that we maintain the order of all the four arrays relative to each other. Next, we plotted a pie chart to visualise the total number of MRI images, the total number of mask images, the total number of images with tumors, and a total number of images without tumors. Later we converted all this arrays to np arrays to perform several complex operations on them.

Now we defined a Class named BrainMRIs. In the constructor of this Class, we pass information such as batch size, image size, input, and target image path. Then we define a function named 'adjust_data' which is used for normalizing the data. If its normalized value is greater than 0.5 we replace it with 1 else 0. Then we define another function named 'dice_coef' which calculates the dice coefficient from the predicted value and true value by the below formula.

$$\mathbf{L}_{dice} = \frac{2 * \sum p_{true} * p_{pred}}{\sum p_{true}^2 + \sum p_{pred}^2 + \epsilon}$$

Then we build over the model with UNET architecture using tensor flow keras. We pass the image with dimension 256 x 256 x 3, where three are the RGB channels for the image. In this model, the first convolution 2D layer uses 32 filters, 3 x 3 as the kernel size and activation function 'relu'. Then we dropout 20% of the data from the output of the above convolution. Then we again perform convolution on the new data with the same convolution 2D layer. Further, we perform the BatchNoralizaiton on the third axis of the output data obtained from the above layer. Then we perform the activation with 'relu' on the above output. So, if in case any of the image pixels are negative we convert it to zero. Further, we do the MaxPooling of the above image with 2 x 2 kernel size, as a result of which our image dimensions change to 128 x 128 pixels.

We perform a similar task again with the convolution layer having 64 filters and after the MaxPooling of the image with 2 x 2 kernel size the image dimension gets changed to 64 x 64. We perform the same operation again with the convolution layer having 128 filters. And the image dimension changes to 32 x 32 after MaxPooling with a 2 x 2 kernel. Further, again we do a similar task with the convolution layer having 256 filters. And the image dimensions get changed to 16 x 16 pixels. After that, we perform convolution on the output data with the convolution 2D layer having 256 filters and 3 x 3 kernel size.

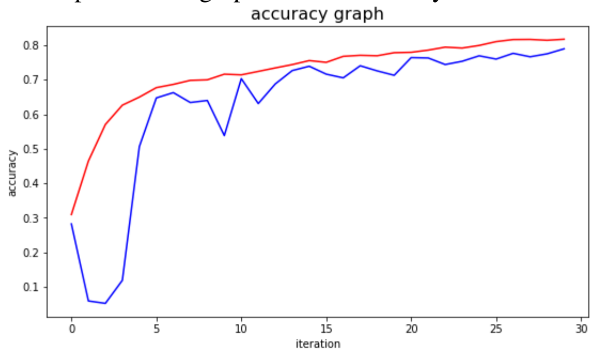
Then we perform the Conv2DTranspose with 256 filters and 3 x 3 kernel size and 'relu' activation on the output data. Then we concatenate the output of this layer with the output obtained in the fourth layer. Now again we perform the convolution on this layer with the convolution 2D layer having 256 filters and 3 x 3 kernel size. Next, we drop out 20% of the data from the above output as we did earlier. Then we again perform the convolution on this layer with the same convolution 2D layer. Further, we perform the BatchNoralizaiton on the third axis of the output data obtained from the above layer. Then we perform the activation with 'relu' on the above output. So, if in case any of the image pixels are negative we convert it to zero.

Next, we perform a similar operation with a convolution layer having 128 filters. In this, we concatenate the output of the above data with the output obtained in the third layer. Next, we do the same process with the convolution layer having 64 filters, and concatenate the output with output obtained in the second layer. Similarly, we perform the Conv2DTranspose on the above data with the convolution layer having 32 filters, and concatenate it with the output obtained in the first layer. Then we do the convolution on the final output with 1 filter, 1 x 1 kernel size, and activation function as 'sigmoid'.

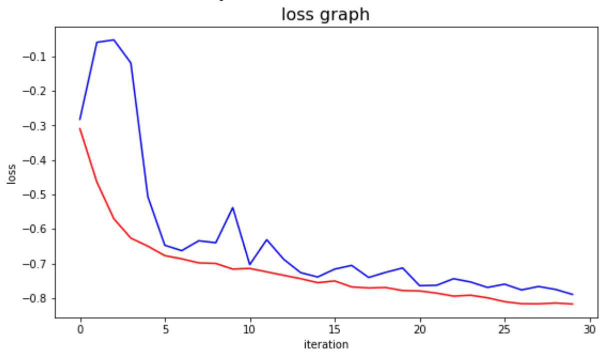
Then we compile our model with our optimizer as 'Adam' and loss as 'dice_coef_loss'. We will train the model with 30 epochs, learning rate as 10$\hat{}$(-2) and decay rate as 10$\hat{}$(-3).

IV. Results

By fitting our model with the training and validation data after 30 epochs our model is trained with 81.72% accuracy. We have plotted the graph for the accuracy and loss as well.



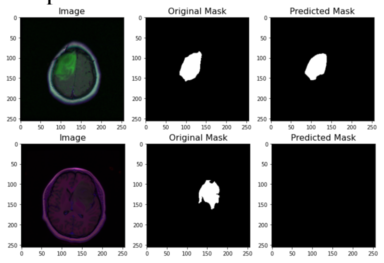
As we can see in the above graph the accuracy for the first 3 epochs decreases with each interaction. But after that the accuracy increases drastically with every iteration. At the last epoch we had accuracy of 0.81 i.e 81.72



As we can see in the above loss graph the loss increases for the first three iterations, but after the three epochs the loss decreases. So our loss will be -0.81.

V. Conclusion

As we know our model gives 81.72% accuracy. And the model works perfectly after abstracting it any testing it on any random image. Loss was not improving after the 2nd epoch(it was remaining the same). However that issue was resolved after adding batch normalisation layers. Masks predicted were not accurate. It was covering the majority area of image. This issue was resolved after tuning no. of filters and learning rate and prediction input normalisation.



Here as we can see two images, the first image gives the perfect prediction of the tumor as the tumour is highlighted in the original image. But on the other hand the model failed to predict the tumor, because it becomes hard for the model to distinguish the tumour in that picture. So it seems like there is need for the preprocessing of the data before feeding to the model.

References

[1] Papers with code - U-net explained. Explained — Papers With Code. (n.d.). Retrieved March 20, 2022, from https://paperswithcode.com/method/u-net

[2] Arora, A., Jayal, A., &amp; Gupta, M. (2021). Brain Tumor Segmentation of MRI Images Using Processed Image Driven U-Net Architecture.