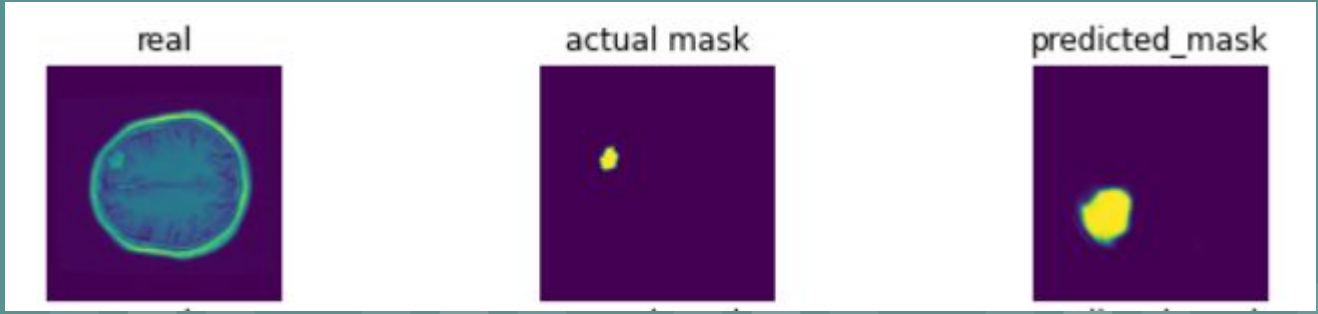
Brain tumor segmentation with RESUNET/UNET

Using UNET to determine brain tumors

By-
Shivansh Singla
UIET CSE

Explaining the problem

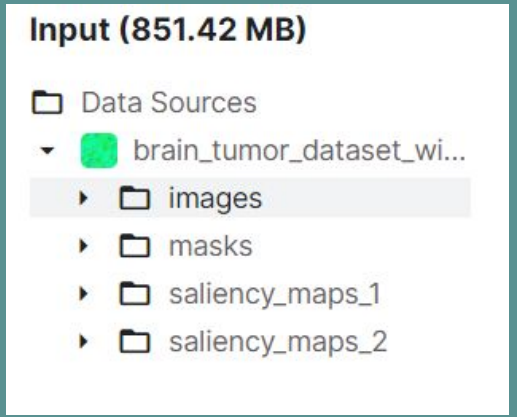
We are given various images of brain scans and brain tumor masks corresponding to it.Our goal is to train our model such that given an image of brain it should be able to detect the tumor .



Dataset description

Our dataset has 4 folders

1. Images contain the images of brain scans.
2. Masks contain the segmented tumors which we need to detect
3. Saliency maps 1
4. Saliency maps 2
5. Approx size of dataset 900MB
6. Thera are 3064 images and masks



What Is a Mask?

The goal of image segmentation is to understand image at the pixel level. It associates each pixel with a certain class. The output produced by image segmentation model is called MASK of the image.

Mask can be represented by associating pixel values with their coordinates for example it 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.

How to download the dataset

We will be using Kaggle API to download the dataset

1. Install kaggle package in python using !pip install kaggle.You can check if its already present using !pip show kaggle.
2. Upload the kaggle.json file which contains the api key to root directory
   `~/.kaggle/kaggle.json`
3. Use !kaggle competitions download -c state-farm-distracted-driver-detection to download it in zip format

Loading/Analysing/Cleaning the data

1. The dataset is in zip folder so first we need to unzip the dataset which can simply be done by
   a. `!unzip /content/brain-tumor-dataset-with-saliency.zip`
2. First we need the natsort library which is used to perform natural sort on strings i.e 1,2,3,4,10 instead of 1,10,2,3,4
3. Since correspondence of images and masks are messed up natsort will bring them in order
4. We need to eliminate folders from images and masks that can be done by running a loop and only including files and skipping folders
   a. `os.path.isfile(img_path + '/' + i)` if this is false we skip because path is a folder
5. Then we declare np array of all zeros which has the shape of (images,image_dimensions)
6. Then we run a loop loading and processing each image before storing it in X,Y which contain the brain images and masks respectively

Image preprocessing

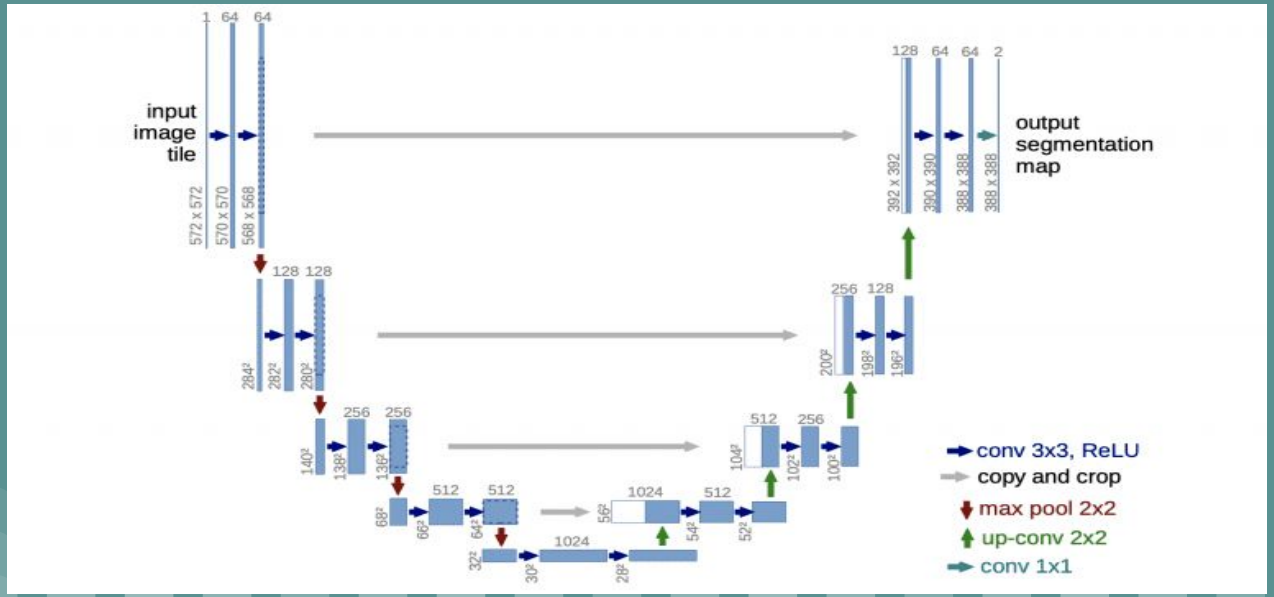
1. Initially sizes of images is 512*512*3 which we change to 256*256*1 grayscale
2. We also normalize the pixel intensities by bringing them b/w 0 and 1 by dividing them by 255.0
3. 255.0 because they need to be in float

What are Saliency Maps?

1. Saliency Map is an important concept of deep learning and Computer vision. While training images of birds how does CNN knows to focus on bird-related pixels and ignore the leaves and the other background things in the image? By using the concept of Saliency Map.
2. Saliency Map is an image in which the brightness of a pixel represents how salient the pixel is i.e brightness of a pixel is directly proportional to its saliency. It is generally a grayscale image.
3. Saliency maps are also called as a heat map where hotness refers to those regions of the image which have a big impact on predicting the class which the object belongs to.
4. Here is an example, the picture shown in the right is the saliency map of the left one which shows the regions which are more attentive part to CNN.

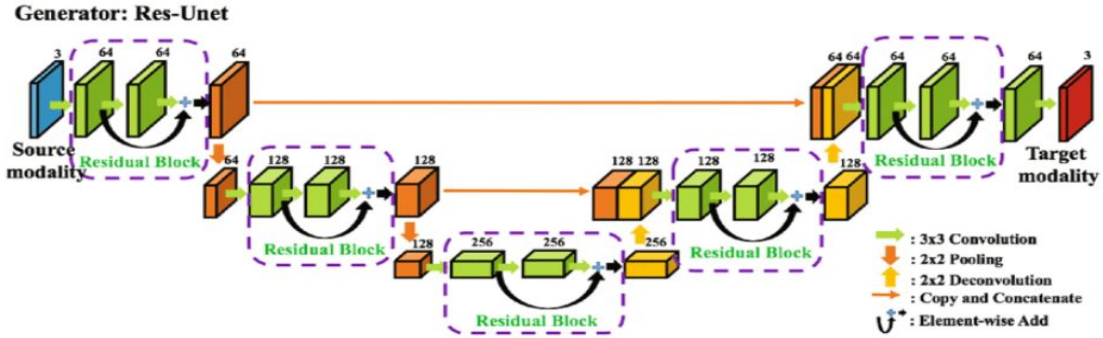
UNET ARCHITECTURE

1. The model architecture is fairly simple:
   a. an encoder (for downsampling)
   b. a decoder (for upsampling) with skip connection
2. it shapes like the letter U hence the name U-Net.



resUNET ARCHITECTURE

ResUNet



Source and Explanation

ResUNet architecture combines UNET backbone architecture with residual blocks to overcome vanishing gradient problem present in deep architecture. ResUNet consists of **three** parts:

- Encoder : The contraction path consist 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.
- Decoder : In decoder each block takes in the up-sampled input from prevoius layer and concatenates with the corresponding output features from the res-block in the contraction path. this is then passed through the res-block followed by 2x2 upsampling convolution layers this helps to ensure that features learned while contracting are used while reconstructing the image.
- 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 2 x 2 up-sampling convolution layers.

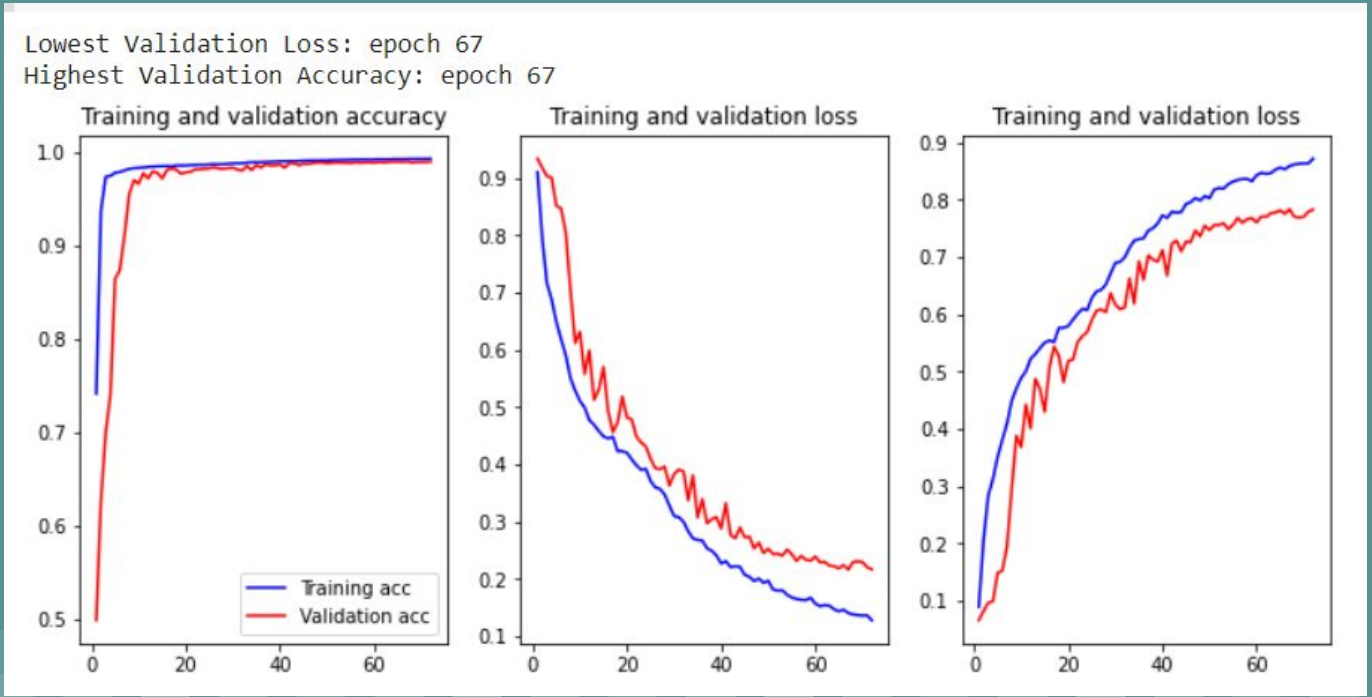
Why are we using Dice score instead of binary cross entropy

1. The main reason that people try to use dice coefficient or IoU directly is that the actual goal is maximization of those metrics, and cross-entropy is just a proxy which is easier to maximize using backpropagation.
2. In addition, Dice coefficient performs better at class imbalanced problems by design:
3. More advanced reasons can be found here - https://stats.stackexchange.com/questions/321460/dice-coefficient-loss-function-vs-cross-entropy
4. Even on checking the training stats you will notice that accuracy has be pretty much constant since the start only reason but dice score has been constantly improving

Measures to prevent overfitting and improve score

1. I am using Early stopping on val_dice_loss to make sure that score doesn't get bad
2. Score can be improved to become way better by using more epochs I just used 75 due to time constraints 300 is probably a good number

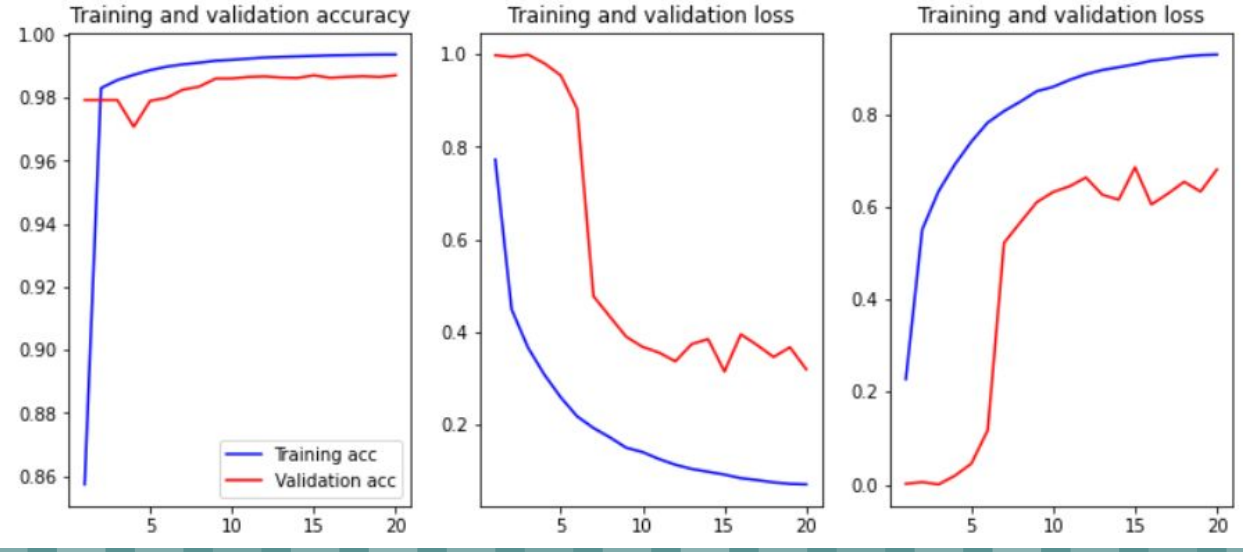
Training statistics visualized of UNET



Training statistics visualized of resUNET

```
1  plot_history(results)
```

```
Lowest Validation Loss: epoch 15
Highest Validation Accuracy: epoch 20
```



COMPARING results

UNET MODEL

```
1  unet_model.evaluate(X_test,Y_test,verbose = 1)
```

```
24/24 [==============================] - 5s 213ms/step - loss: 0.2163 - dice_score: 0.7838 - accuracy: 0.9892
[0.21633897721767426, 0.7837794423103333, 0.989155650138855]
```

RESUNET MODEL

```
1  res_unet_model.evaluate(X_test,Y_test,verbose = 1)
```

```
24/24 [==============================] - 6s 266ms/step - loss: 0.3152 - dice_score: 0.6850 - accuracy: 0.9871
[0.31516724824905396, 0.6850284934043884, 0.9871124029159546]
```

Visualizing the results