ASSIGNMENT 1

Submittes By : Prateek Kumar Agnihotri (2016BB10033)

Report in a go-

1) Pre-processing : Shuffle the patients directories → Took top 64 2D slices with highest net intensity from each 3D image→ Normalize each slice to zero mean and unit variance → Shuffle again.

2) Model : ResUnet for sementation followed by a threshold on net pixel confidence.

3) Hyper-parameter tuning : Bayesian optimization using Gaussian Process.

4) Reasons of failure on test data: Presence of extra structures like:skull, ears, eyes in test data and difference in contrast mappings of different tissues and tumors in test and train data .

5) <u>Improvized Model Training</u>: Unlearn intensity information.

6) Further Possible Experiments.

(Note: Part V is my favorite.)

I. Pre-processing

To remove any kind of bias present in the selection order of patients as the 1st step, I merged both HGG and LGG and shuffled all the sub-directories present in them.

The second step was to select relevant 2D slices or the slices with full brain only. One possible way to do this is to set a threshold on the net pixel intensity in a slice. But as it can be seen in Fig-1, both (a) & (c) have full brain but the net pixel intensity varies lot, even a partial brain image from (c) can give pixel intensity much higher than (a).

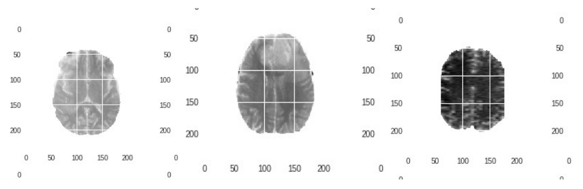


Fig-1 Net pixel intensity across images varies a lot.

So, a natural intuition says, in the full 3D image, the middle most slices will have full brain, and which is the case in Fig-2 (a). But as it can be seen in Fig-2 (b), middle most slices are not the slices with highest net pixel intensities i.e. they might not have full brain.

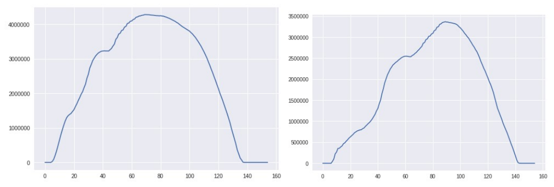


Fig-2 Net pixel intensity of a 2D slice vs its depth in 3D image.

In order to overcome these issues, I manually visit across the depth of some 3D images and found that atleast 70 out of 115 slices have full brain images. So, to be on safer side, I took 64 highest net intensity slices.

Now, the issue is the highly varying net intensity across images. Since, the tumor presence depends on relative contrast of pixels in an image, not on the net contrast of the image. I solve this problem by normalizing each image i.e. zero mean and unit variance (mentioned as InstanceNormailzation in the code).

As final step shuffled the images again.

II. Model (The Function Approximator)

The way to select/build a deep NN model is, clearly formulate the problem and encode your priors into it.

In the training dataset, we have segmented tumor images, so instead of asking at the image level, whether the given image has tumor or not, we can ask at the pixel level, whether the given pixel is tumorous or non-tumorous. Pixel level classification ensures model is trying to learn the correct region in the image, instead of modelling some other non-tumorous variation.

Now, after getting the response of all the pixels, a simple way could be, if the net confidence of all the pixels is above certain threshold, then the image is tumorous.

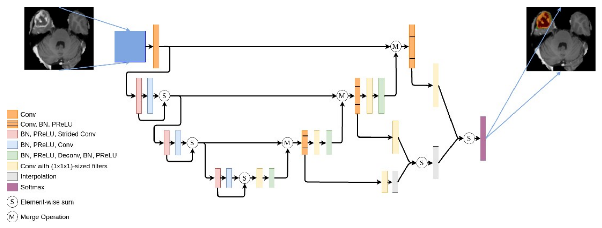


Fig-3 ResUnet

The present SOTA for segmentation on biological data is ResUnet with instance-normalization. Many reasons are given in papers for its performance. So, I implemented that.

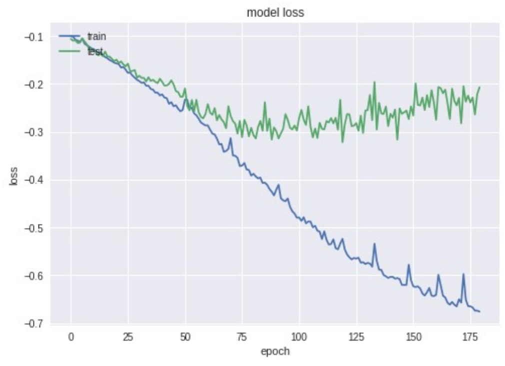


Fig- 4 Green: Validation Loss & Blue: Training loss.

Since, it requires a large time to complete 80 epochs. So, to get fig-4, I used a portion of the complete data. The fig-4 depicts that the model capacity of the model (obtained in sec-

IV) is sufficient enough to approximate the required function. Optimization with suitable regularizations will make it generalize well.

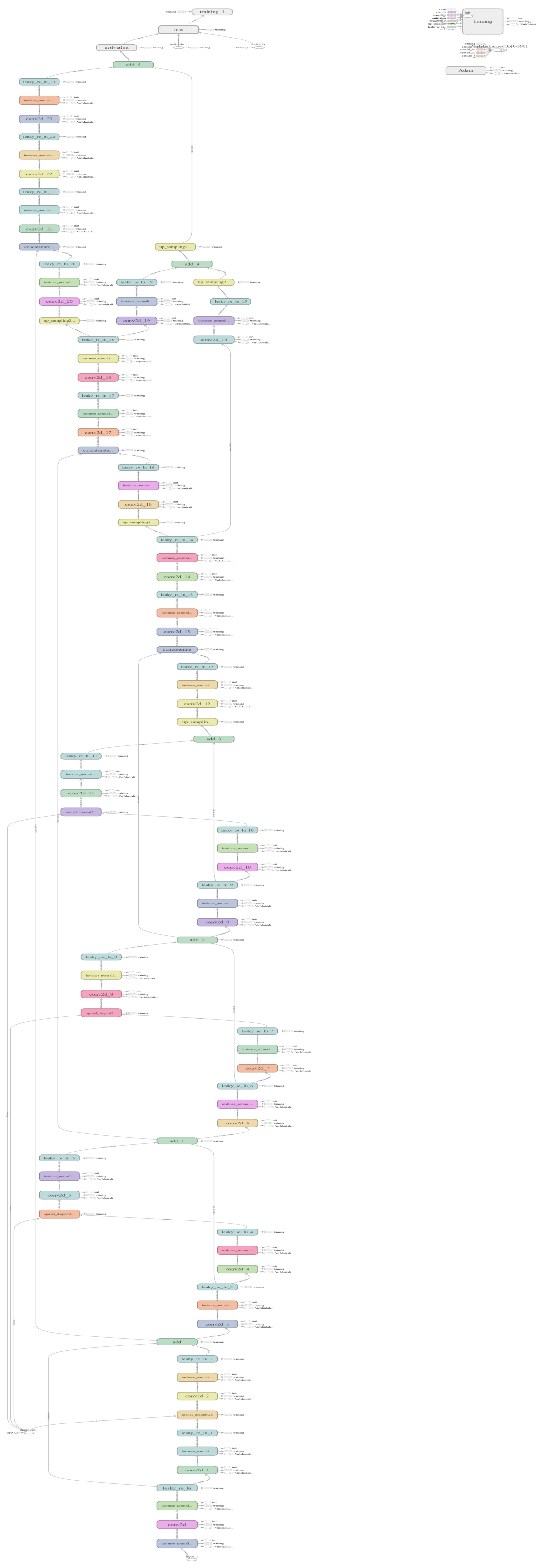
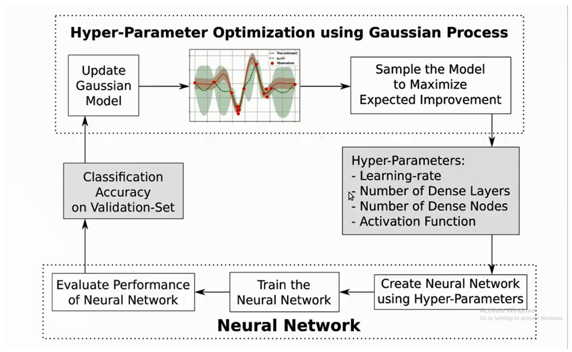


Fig-5 Detailed model structure.

III. Hyper-parameter Tuning

(Bayesian Optimization using Gaussian Process)



Hyper-Parameter Optimization using Gaussian Process
Update Gaussian Model
Sample the Model to Maximize Expected Improvement
Hyper-Parameters:
- Learning-rate
- Number of Dense Layers
- Number of Dense Nodes
- Activation Function
Classification Accuracy on Validation-Set
Evaluate Performance of Neural Network
Train the Neural Network
Create Neural Network using Hyper-Parameters
Neural Network

In previous course, we have used Gaussian Models like: GMM to approximate functions. The similar approach can also be used to approximate validation loss vs hyperparamter surface.

To implement Bayesian Optimization using Gaussian Process, I have used meta optimizer skopt with expected improvement (EI) as the criteria of optimization-

$$EI(x)=E_{max}\,(f(x)-f(x+),0)$$

Where f(x) is the fitness function, returning optimized weighted dice score.

Due to computational limits, I took following as hyperpameters and performed 11 iterations only-

- Initial learning rate – Range = [1e-6,1e-2], prior = log-uniform.
- Depth - Range = [2,5]
- Initial number of filters – Range = [2,32]
- Dropout rate - Range = [0,1]
- Optimizer – [Adam, SGD]
- Loss function – [weighted dice coefficient loss, ‘binary cross entropy’, ‘mean squared error']

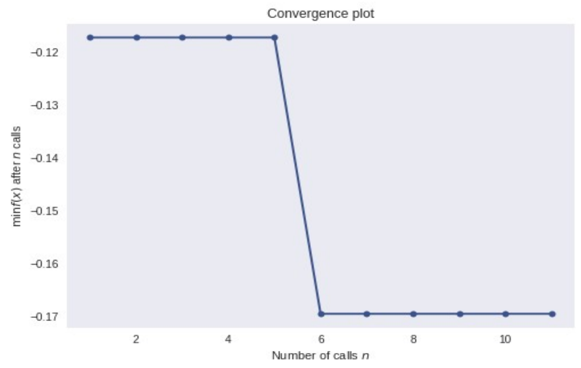
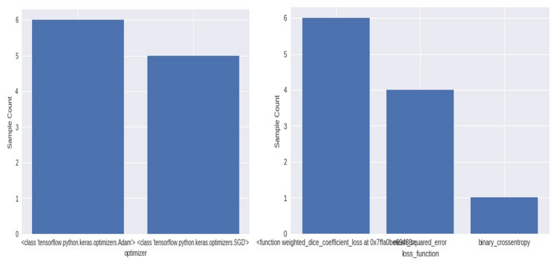


Fig-6 minimum value the model can achieve vs iteration number.

On 6th iteration it explored a value which reduced the minimum value that model can achieve as compared to my initial guess.



(a) (b)

Fig- 7 (a) frequency count vs optimizer used (Adam vs SGD). (b) frequency count vs loss function used.

Even in such low number of iterations, where most of them are used for exploration instead of exploitation, most of the time it used Adam optimizer and weighted dice loss (Fig-7 (a)&(b)). Results as per our prior knowledge.

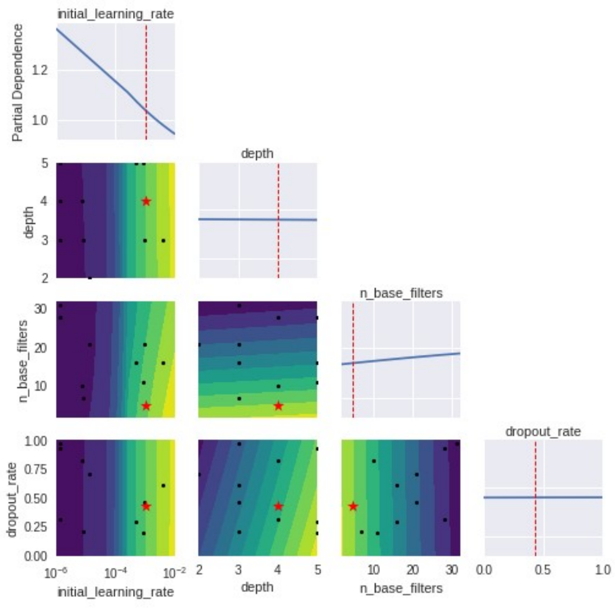


Fig-8 Plots on main diagonal- effect of hyperparameter on negative of fitness score. Others- Partial dependency plots (fitness score surfaces on taking 2 hyperparameters at a time). Red star is the optimal point. Blue → Yellow, low score → higher score.

Because of less no. of points, these partial dependency plots are not that reliable, but for the sake of interpretation, we can say:

- As the learning rate increase from 10^{-6} to 10^{-2} model performs better.
- Depth has almost no effect on fitness score. So, given any depth we can always find same optima by tuning other hyperparameters like: no. of filters.
- As the no. of filters increases learning reduces.
- Similar to depth, dropout also has no effect on fitness score.

Next fig.6 shows, after exploring a better fitness score, metaoptimizer tried to exploit it by searching more hyperparameters near to it like in case of learning rate, while in case of dropout, where it has not observed any significant change, it has sampled evenly.

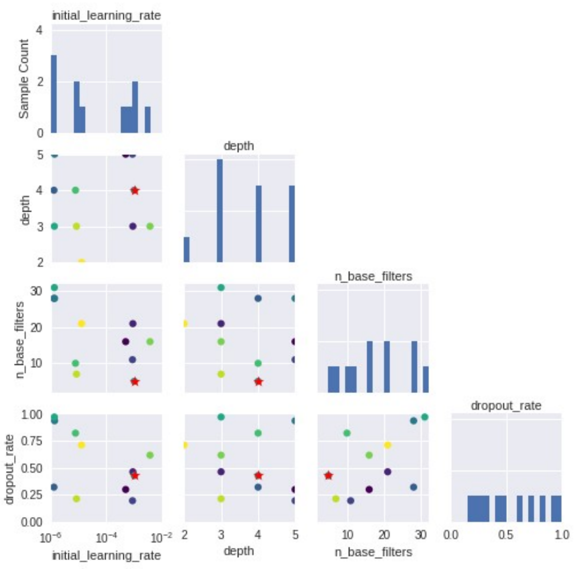


Fig-9 Plots on the main diagonal are histograms, depicting how many samples are taken from where in this search space dimension and other plots depict what different hyperparameters metaoptimizer has tried.

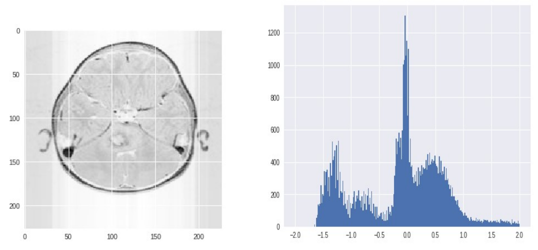
Result of experiment: optimal point is-

- Initial learning rate - `0.001`
- Depth – `4`
- Initial number of filters – `5`
- Dropout rate – `0.4285`
- Optimizer – `Adam`
- Loss function - `weighted_dice_loss`

IV. Reasons Of Failure On Test Data

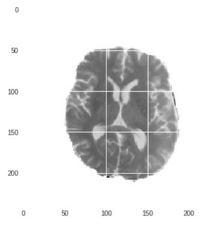
The presence of extra structures like: eyes, ears… makes it difficult to segment.

But there is one more and more serious problem.

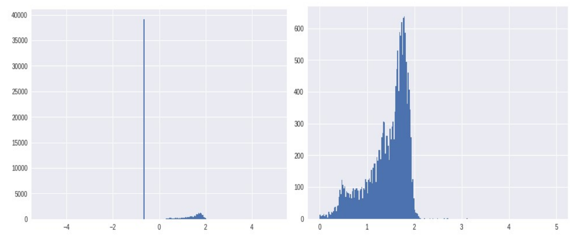


(a) (b)

Fig-10 A normalized Image from test data and its corresponding intensity histogram.



(a)



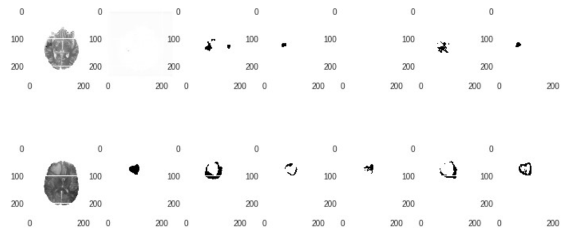
(b) (c)

Fig-11 (a)A normalized image from training data (b) Corresponding intensity histogram (c) Zoomed region between 0 and 5.

The **<u>tissue to pixel intensity mappings are totally different</u>** in test and train data. Like in training data tumor is lighter than other tissues, while in case of test data tumor is darker than other tissues.

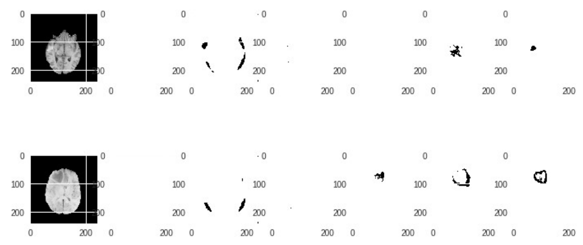
V. Improvized Model Training

As we see in part IV tissue to intensity mappings are totally different in both the cases. So, if the model's segmentation criteria is based on intensity then it is going to fail. So, I performed an experiment, just multiplied the inputs by -1 and observed the output.



(a) (b) (c) (d) (e) (f) (g)

Fig-12 (a) Brain image from training data (b) Predicted segment type I tumor (c) Predicted segment type II tumor (d) Predicted segment type III tumor (e) True segment type I tumor (f) True segment type II tumor (g) True segment type III tumor



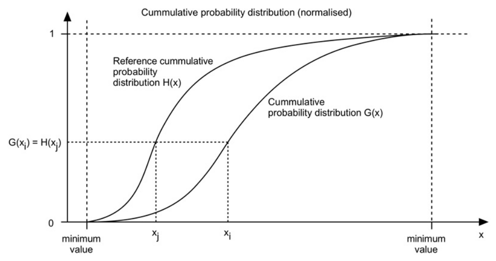
(a) (b) (c) (d) (e) (f) (g)

Fig-13 (a) Brain image from training data (b) Predicted segment type I tumor (c) Predicted segment type II tumor (d) Predicted segment type III tumor (e) True segment type I tumor (f) True segment type II tumor (g) True segment type III tumor.

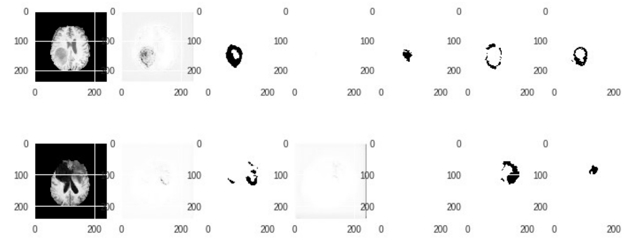
As we can see, it performed well in case 1 but failed terribly in case 2.

Now, the solution is either the model has to unlearn these intensity mapping based segmentation dependency or make the tissue to intensity mapping same.

I performed both. Trained it by multiplying input images by -1 (even I wanted to do different-different intensity mappings but due to time constrain performed this only) and also transform the histogram of test images to get it as near as possible to that of average histogram of training images.

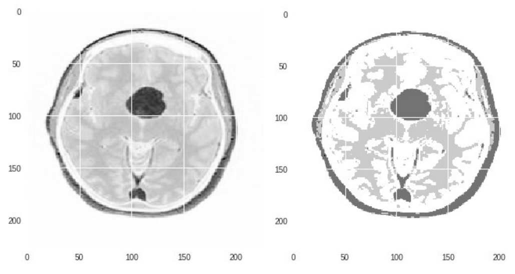

Cummulative probability distribution (normalised)
1
Reference cummulative probability distribution H(x)
Cummulative probability distribution G(x)
G(x_i) = H(x_j)
0
minimum value
x_j
x_i
minimum value
x

The algorithm used for histogram matching is as follows. The cumulative histogram is computed for the image and average cumulative histogram is computed for training data. For any particular intensity value (x_i) in the image to be adjusted has a cumulative histogram value given by $G(x_i)$. This in turn is the cumulative distribution value in the average cumulative histogram for training data, namely $H(x_j)$. The input data value xi is replaced by x_j.(Procedure mentioned pictorially in the above diagram).

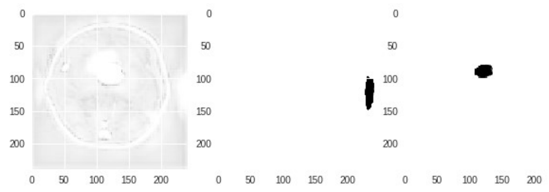


(a) (b) (c) (d) (e) (f) (g)

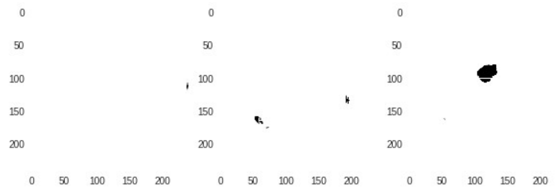
Fig-14 (a) Brain image from training data (b) Predicted segment type I tumor (c) Predicted segment type II tumor (d) Predicted segment type III tumor (e) True segment type I tumor (f) True segment type II tumor (g) True segment type III tumor.



(a) (b)



(c) without histogram matching



(d) with histogram matching

Fig-15 (a) An abnormal brain from test data. (b) Image after histogram matching (c) Result of the model without histogram matching. (d) Results after histogram matching.

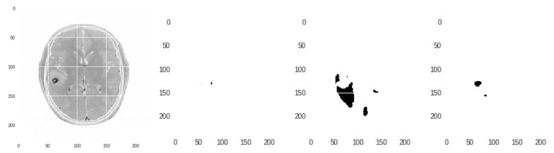


Fig- 16 Abnormal brain and Corresponding tumor segmentation results on histogram matched image.

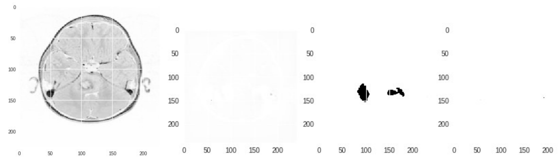


Fig- 17 Abnormal brain and Corresponding tumor segmentation results on histogram matched image.

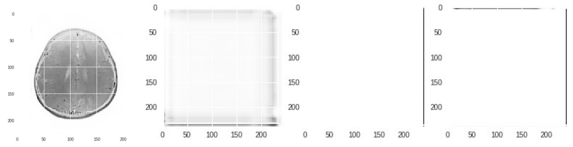


Fig-18 Normal brain and Corresponding tumor Segmentation results on histogram matched image.

Results:

- As it can be seen in fig-14 (c) &(d) that histogram matched image performed slightly better than non-histogram matched image, but I am sure that after a little more training there will be **no requirement of histogram matching.**
- In fig-15 tumor is darker than brain, while in fig-16 and fig-17 it has lighter intensities but still model is able to identify. Implying model **has unlearned intensity based mappings**.
- Model is quiet **robust**. Even it has not seen during training but still, **extra structures** like ears and skull in fig-17, have almost **no effect on model prediction**.
- Model's accuracy is quiet high on training data (almost always it predicts correctly, whether there is tumor or not) and fairly good on test data.

VI. FURTHER EXPERIMENTS.

- Making it more robust by augmenting data further by different - different histogram mappings.
- Even structures (like: ears and skull) are not affecting its performance significantly, but still to be more robust, it needs to get trained on segmented images containing structures.
- Needs to hyper-parameter tuned again.
- A classifier on the top of this model can improve the accuracy significantly.