CS6140: Mini Project Report

Image Segmentation on Biomedical Images using U-Net and Capsule Networks

Aswin Shriram Thiagarajan

(thiagarajan.as@northeastern.edu)

Khoury College of Computer Sciences

Northeastern University

Boston, MA

Advised by Prof. Dr. Predrag Radivojac (predrag@northeastern.edu)

1. Objectives & Significance

1.1. Goal

The original goal of this project was to implement **Capsule Networks** [1] for a multi class Biomedical image segmentation task (multi-modal Brain Tumor dataset) and compare its performance with that of **HoVer-Net** [2] with modifications to the architecture to only contain the segmentation task.

However, from the feedback received during the project proposal submission it seemed more appropriate to compare the results with a **U-Net** [3] architecture than a **HoVer-Net** [2]. Hence, decided to benchmark it against the **U-Net** Architecture. The author of the **Seg-Caps** [1] paper also used U-net as one of the benchmark model architectures.

1.2. Significance

The State-of-the-art deep learning methods employed in all the image related tasks in Computer Vision use the traditional Convolutional Neural Networks (CNN). Even though they all could achieve significant performance on the tasks like image classification, image segmentation, facial recognition, image generation, etc. they still have shortcomings like

- Inability to understand the pose invariance in features
- Loss of information in the feature representation (pooling layers)
- Requiring more examples/data
- Needs more parameters for a model to perform efficiently

and are susceptible to perturbations such as

- Changing the orientation/pose of the feature set representations (e.g., Placing the eyes and mouth of a face elsewhere in the image containing a face)
- Class imbalance while training
- Noisy pixels (adversarial attacks on CNNs)

In a biomedical image classification or segmentation task the accuracy is of pinnacle importance and we cannot afford to make mistakes. We need a more robust solution that can overcome the above limitations. Hence, this project focuses on methods that can be relied more on when applying them in the biomedical domain.

1.3. Motivation

CNNs use scalar to represent the features, they are just mere probability that says whether a feature is present or not. We need a better representation of the features (like pose/orientation of a feature) and the information should capture the relationship between the features effectively. This main motivation of this project is to develop a more robust solution that can perform well in practical biomedical application like identifying and localizing a tumor given an MRI image. Hence, wanted to implement Capsule Networks for Image Segmentation.

1.4. Novelty

The paper [1] was the first to implement Capsule Networks for an image segmentation task and did it successfully for a binary segmentation task. However, I wanted to be the first to implement Capsule Networks for a Multi-class Semantic Segmentation task. Another additional complexity is the multi-modality (modality = 4) of the dataset. The original paper had used single modal dataset. For the U-Net architecture I used padding instead of cropping (proposed in the original paper) while concatenating the layers so that the output size of the image is retained as the size of the input image. The original paper's output image is smaller in dimension (388 x 388) compared to the input dimension (572 x 572).

2. Background

2.1. Background Concepts

2.1.1. Capsule Networks

Capsules are group of neurons that represents a feature in a vector form as opposed to conventional CNNs that represents a feature in a scalar form. The feature vector in a capsule represents the instantiation parameters (color, hue, angle, etc.) of the feature object. This concept is inspired from

the field of computer graphics. The inception of this concept happened to overcome the challenges faced by a CNN such as invariant property of an object's class. The networks have the potential to handle images effortlessly even when the spatial relationship is perturbed. This cannot happen with the CNNs due to its pooling of features and their relative occurrence is not captured in terms of inclusion of a pose vector.

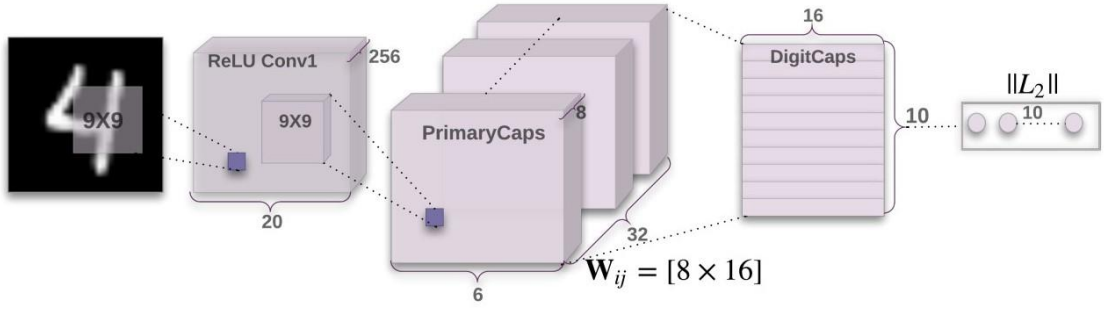


Figure 1: Capsule Networks from the original paper [4]

They have a paradigm called routing by agreement which enables to learn the hierarchical relationship between the parent and children capsules. The children capsules individually predict their parent outputs which is a weighted average that is learned by the dynamic routing algorithm. To tell whether an object is present or not depends on whether the overall agreement between the children capsules that predicts the same parent with higher probability or not. The agreement can be pictorially represented as shown in figure 2.a and 2.b.

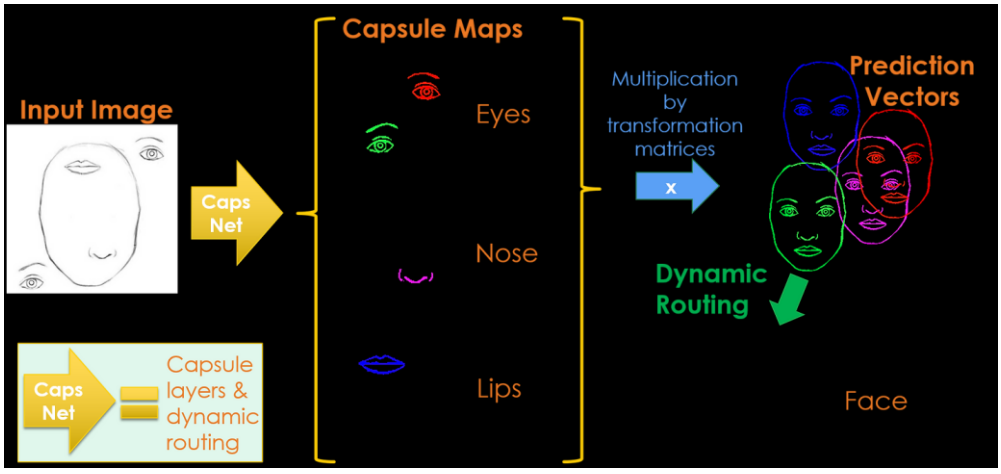


Figure 2.a: Disagreement between the children capsules [5]

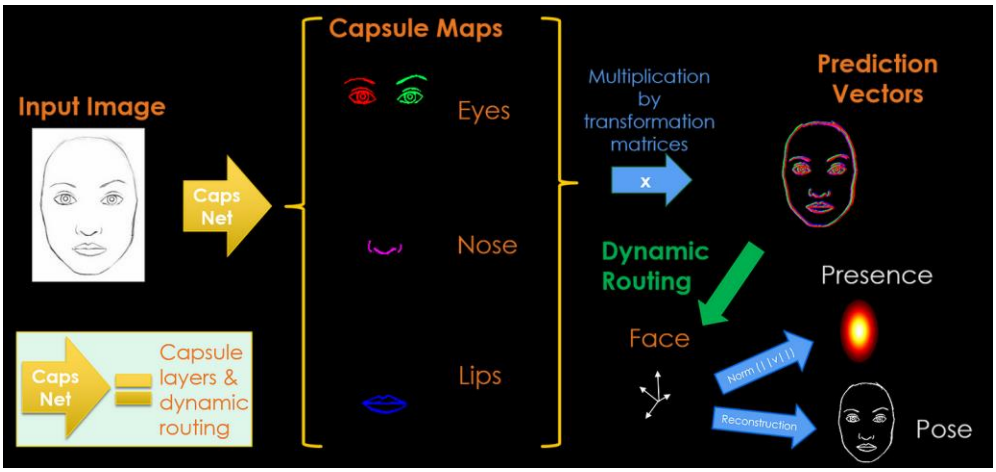


Figure 2.b: Agreement between the children capsules [5]

Procedure 1 Routing algorithm. (by Sabour et al., 2017)

procedure ROUTING($\hat{\boldsymbol{u}}_{j|i}, r, l$)
  for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
  for r iterations **do**
    for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$
    for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
    for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$
    for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
  return $\mathbf{v}_j$

Figure 2c: Routing Algorithm [5]

2.1.2. U-Net

This network architecture is completely made up of only CNN layers for a semantic segmentation task. It was exclusively developed for biomedical image segmentation tasks. The architecture consists of an encoder-decoder like structure to capture the context of the input and precise localization respectively. This architecture is a sample efficient architecture relying on less annotated images and more data augmentation. There is an interesting operation which is concatenation of layers from the contracting path (encoder) to the expanding path (decoder). This enables direct mapping of information from both the actual image and the label localization. This architecture does not have any fully connected layers and entirely works on Convolution operations and feature extraction.

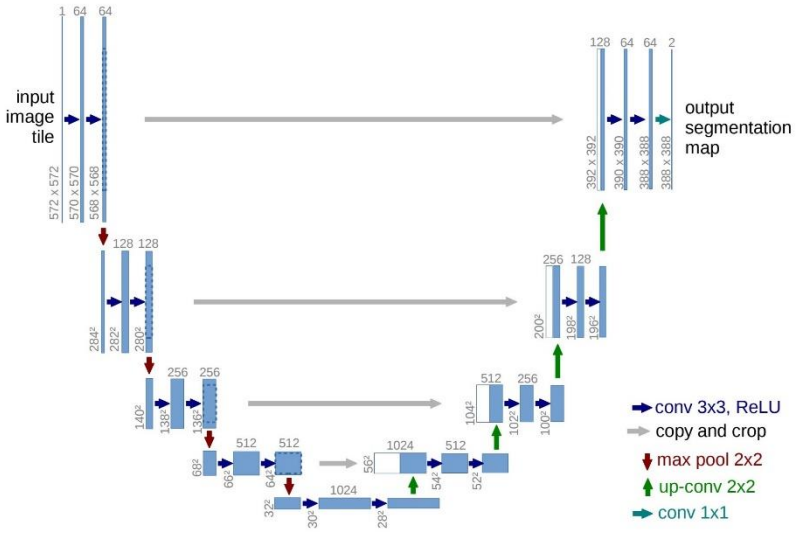


Figure 3: U-Net from the original paper [3]

The network is trained over the weighted cross entropy loss, where the weights are used to balance the unbalanced class size in the datasets. This loss is calculated for each pixel.

$$L_{\{\text{cross-entroy}\}} = -\sum_{i=1}^{n} \frac{\sum_{j=1}^{c} w_j \, y_{ij} \log(p_{ij})}{\sum_{j=1}^{c} w_j}$$

Where, n → number of examples

c → number of classes

$p_{ij} \rightarrow \text{Softmax}(x_i) \rightarrow \frac{e^{x_{ij}}}{\sum_{j=1}^{c} e^{x_{ij}}}$

x_i → c dimensional vector

$y_{ij} \in \{0,1\}$

w_j → weight of each class

2.1.3. Image Segmentation

In Computer Vision, the concept of image segmentation is dividing an image into multiple objects by localizing them. That is identifying an object and highlighting it with bounded shapes or contours like the ones shown below. It can also be simplified as tagging every pixel to its corresponding label.

Figure 4: Example of image segmentation from [6]

Several deep learning techniques have been used in image segmentation applications. Especially with biomedical images that are predominantly CT scans, X-Ray images and MRI images needs to be segmented into objects of specific characteristics (like tumor, vessels, nucleus, etc.) which will be consumed for disease diagnosis.

2.2. Previous Work

The paper [1] that I mainly followed will be discussed here in brief.

Dataset used: LIDC-IDRI Pathological Lung dataset, 3D CT scan image, 512 x 512 image size, 1 channel (1 modality)

Architectures implemented: Capsule Networks for Segmentation (Seg-Caps), U – Net, Tiramisu, P-HNN

The Seg-Caps in the paper had a similar structure to that of U-Net. Except the CNN layers were replaced with Capsule Network Layers. The paper had couple of new novelties like

- Locally Constrained dynamic routing
- Deconvolutional Capsules
- Weight Parameter sharing between the capsules (to reduce the number of parameters)

The comparative results motivated me to pursue the implementation of the Seg-Caps architecture for image segmentation task. The Seg-Caps had a better performance despite having lesser number of parameters than the competing algorithms.

2.3. My Work

Since the U-Net architecture and Seg-Caps architecture are similar, I started off with U-Net implementation from scratch to get a good understanding and made modifications to work on the multi-class multi-modality cases.

Dataset used: Brain Tumor Dataset [7] [8] [9] [10], 4D MRI Image, 240 x 240 x 155 image size, 4 channels (4 modality)

Architectures implemented: U-Net, 3 Layer Seg-Caps (Able to train, but the reconstruction loss is not converging)

My work is particularly interesting because of the following novelties.

- Multi-class segmentation using Capsule Networks
- Multi modal dataset
- Custom Generated dataset (processed the given dataset and curated for my task)
- Seg-Caps was not implemented on this dataset before

There were various combinations of approaches and datasets that people have earlier worked on. My work is a result of various thought process to solve the problem of multi-class biomedical segmentation. Starting from how I can effectively use the dataset, what architecture will solve my problem, how can I adapt and generalize the current binary-class implementation to multi-class, to what loss functions I can utilize for training, the whole project was conceptually intensive and I working on them all for the first time. The challenges faced in the problem solving itself is an interesting aspect of my project.

My Iterative efforts:

- Should I train with Cross-Entropy Loss
- Should I train using Dice Loss
- Should I train using IoU Loss/Jacquard Index
- Should I train using combination of losses
- Should I do post processing to improve my accuracy

3. Methods

3.1. Data

This project will use Brain Tumor dataset from the Medical Segmentation Decathlon competition intended for providing an open-source benchmark for medical image segmentation tasks. This dataset is an aggregation of BraTs 2015, 2016, 2017, 2018 competitions and the image is registered & centered.

3.1.1. Data Notation

4-dimensional dataset with 484 images for training and 266 for testing. It is a multi-modal dataset and a multi-instance semantic segmentation problem with different density of tumors. I chose this for its complexity and its multi-modal nature.

The data is a 4D image with axes – Height, Width, Depth/Num of Slices, Modality.

- Transversal – Slicing across Depth axis
- Coronal – Slicing across Height Axis
- Sagittal – Slicing across Width Axis

From the below images, we can see how the tumor localization is different and can be seen clearly. Since we have 484 4D images, we can generate multiple images for training the model.

Further exploration and data visualization can be referred from the 'data_loader_brain_tumour.ipynb' file submitted along with the report.

Transversal Coronal Sagittal

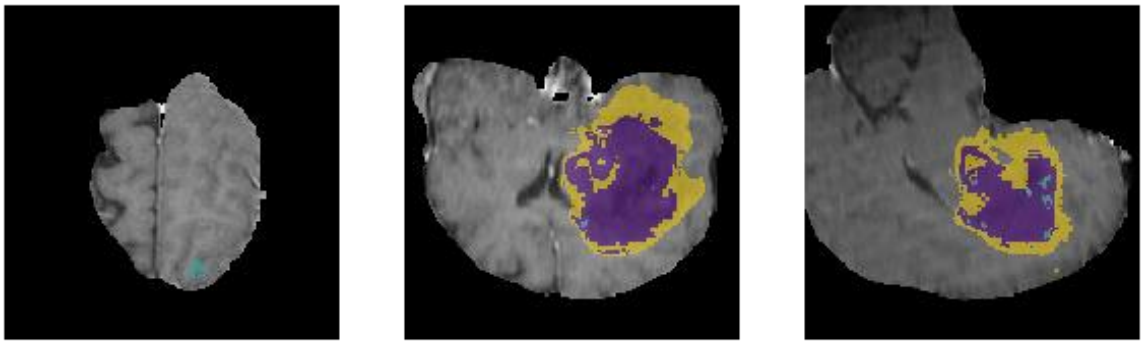


Figure 5: Axis wise Images of one modality

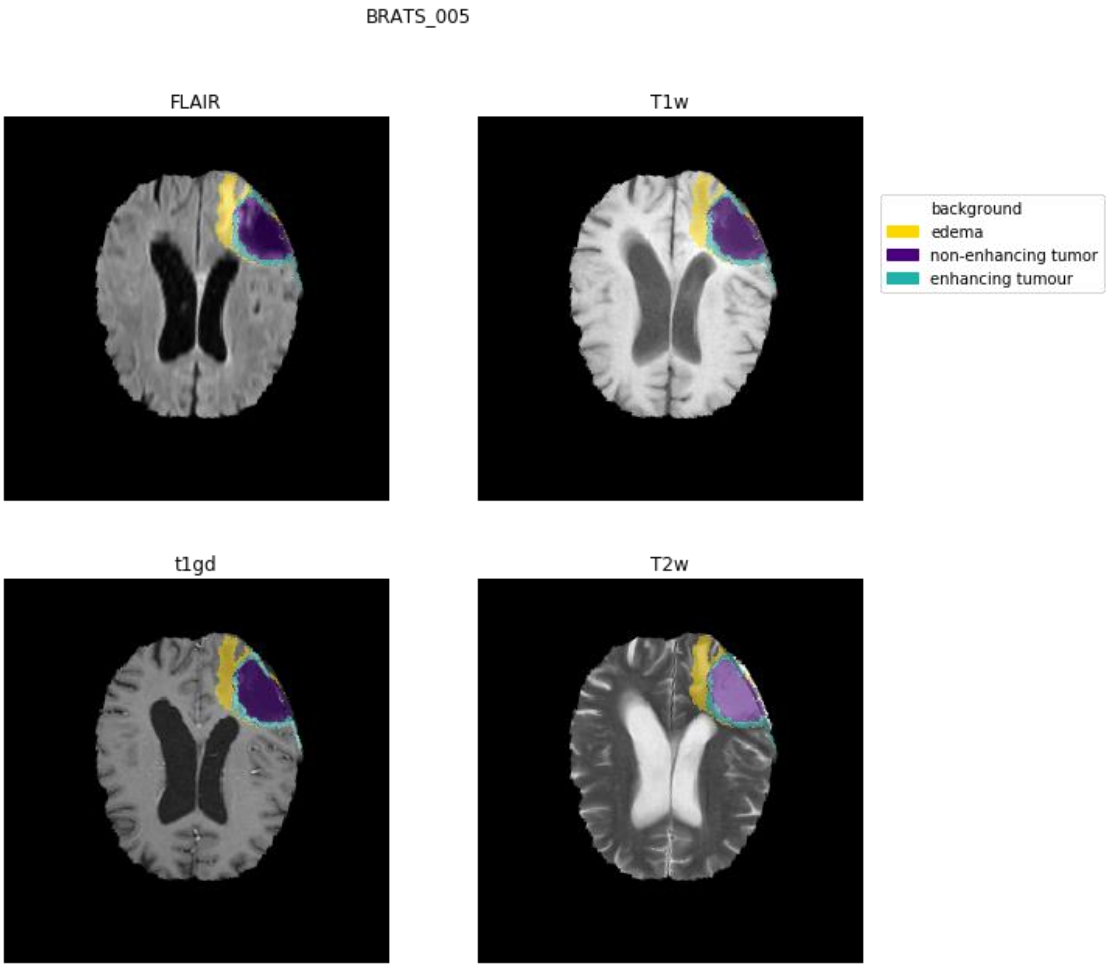


Figure 6: Modality wise images of the same transversal axis

The four modality (FLAIR, T1w, T1gd, T2w) corresponds to different image processing techniques used to process the images from the MRI scanners. Images below show the difference in contrast

of each modality of the same layer image. The labels of tumor (edema, non-enhancing tumor, enhancing tumor) are super imposed on the actual image.

It can clearly be seen from the histogram images, that the intensity of the pixels are in the range of 1000s unlike the traditional computer vision problems with RGB channels and intensity maxed out at 255. The distribution of pixel intensities vary differently across different modalities. This becomes a challenging task without the domain knowledge to use the dataset for training.

Challenges:

- Should I train each model for every channel/modality separately
- Should I train a 3D U-Net using 3D voxel of images
- Should I obtain images of size 240 x 240 from the transversal plane for training (slicing across the 155 slices/dimensions)
- Should I obtain images of size 154 x 154 slicing from all the 3 planes (coronal, transversal, sagittal)
- Should I combine all the four modal images as images with 4 channels

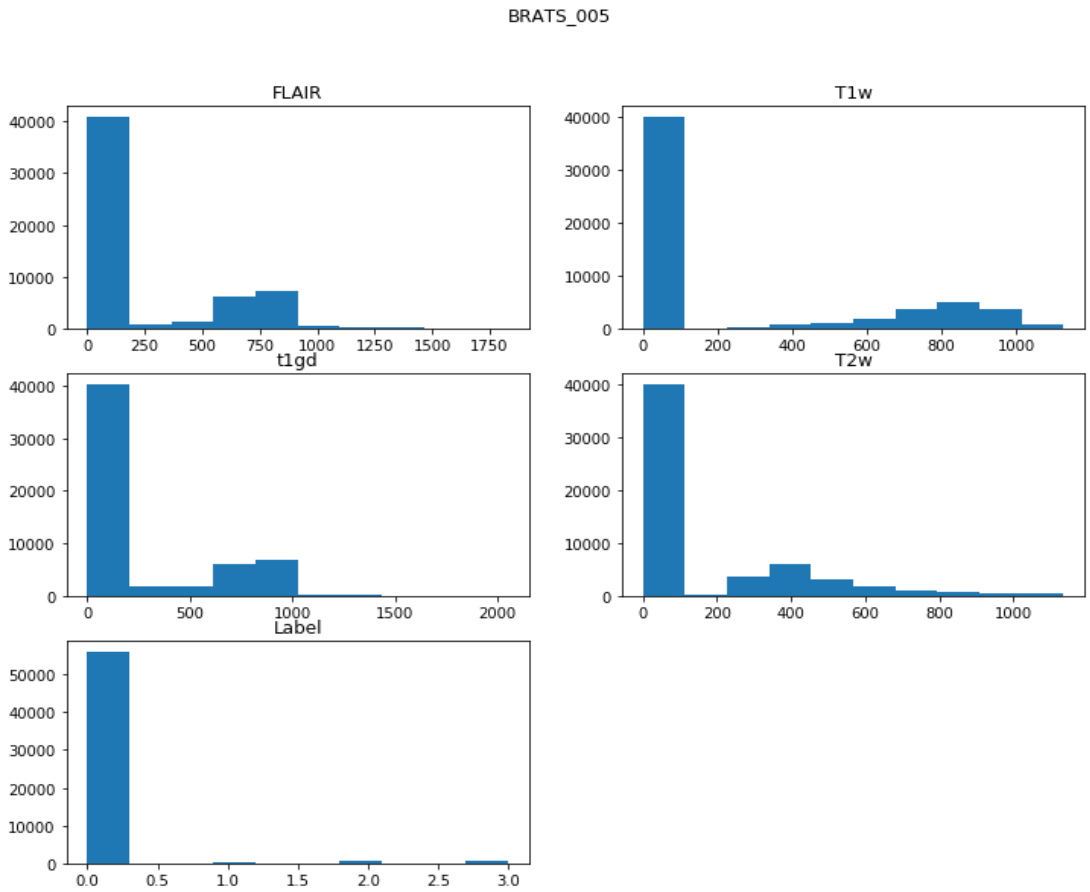


Figure 7: Histogram of Modality wise images of the same transversal axis

The label has 4 values.

0 – background

1 – Edema

2- Non-Enhancing Tumor

3 – Enhancing Tumor

3.1.2. Training Data Generation

For training, I generated 2D slices of images from all the 3-axis mentioned. The logic I used for generating slices shown below. The image size I finalized was 128 x 128 x 4. This is done to suit the U-Net architecture, because the network required a image dimensions divisible by 2. I cropped the image 154 x 154 (the maximum size uniformly across all axes) to 128 x 128.

- Used OpenCV package to estimate the center of the object in the image. This is done because in some axis the image slices will have brain object on one side away from the center. Since we need all the images centered for training for better performance, I cropped them based on this logic.
- Discarded the image slices which is empty or has no tumor labels.
- Had a threshold for the percentage (at least 15%) of tumor (labels 1, 2, 3) present in the image so that the class is not too skewed. (even though it already is)
- The images were saved to Mongo DB for easy, organized and compressed storage.
- The images were saved as 128 x 128 x 4 dimensions to the DB, where the last dimension is the number of channel equivalent here representing the modality. So, I finalized with stacking up of the modalities as channels.
- The image labels were changed from 4 dimensions to 5 dimensions. This was done to differentiate between the brain background and empty pixels in the image. The number of channels is five because I added 1 to all the labels that are not empty.

New label has 5 values.

- 0 – Empty Background (Black space in images)
- 1 – Brain Background
- 2 – Edema
- 3- Non-Enhancing Tumor

- 4 – Enhancing Tumor

Roughly 5839 images were generated in the process that was used for training, validation, and testing (80-10-10).

The data generation and saving script can be referred from the ‘training_data_generator.ipynb’ file submitted along with the report

3.2. Methodology

U-Net:

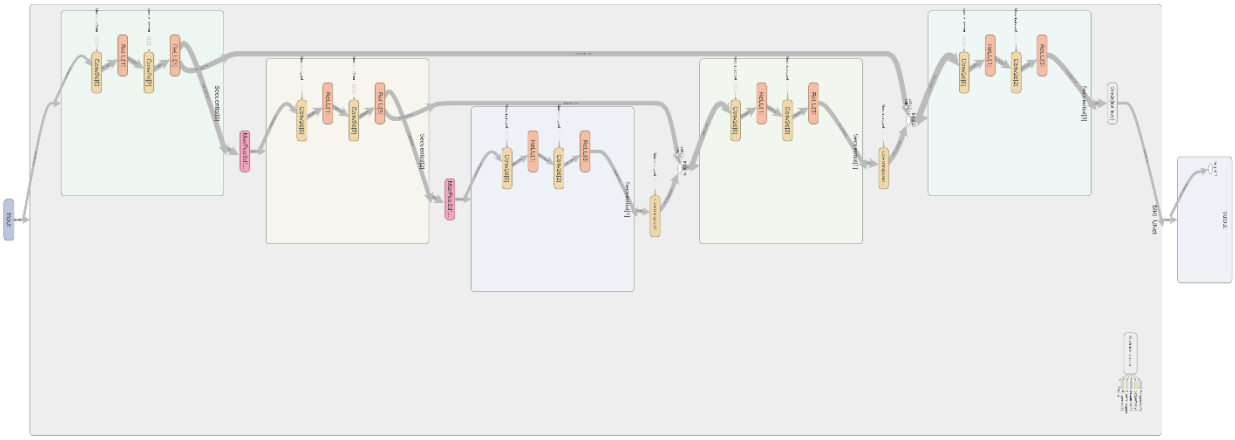


Figure 8: U-Net Architecture I implemented

(The image can be found along this submission { figures/U-net-my-architecture-tensorboard.jpg} for better clarity)

The 2 big square blocks on the left is a sequence of Convolution (3x3 kernel, stride = 1, padding = ‘same as input’) operation followed by ReLU activation. From one block to another there is a max-pool operation to reduce the height and width to half the initial. From the 3rd to 4th and 4th to 5th big block there is a deconvolution operation (3x3 kernel, stride = 1). This operation doubles the dimensions of the previous layer output. The thick grey arrow shows the concatenation of the layers. Finally, there is a 1x1 convolution operation to get the output of shape (128 x 128 x 5).

The difference in my implementation is the depth of the network. The original paper [3] dealt with a depth of 5 and worked with a 572 x 572 x 1 image and output as 388 x 388 x 2. Since my image size is small, I stopped with a depth of 3 and my input image size is 128 x 128 x 4 and output is 128 x 128 x 5. I retained the image size of the output label by padding the image from the contracting path for every

convolutional operation. This restored the image dimensions and did not require the cropping technique proposed in the paper to match the shapes of the concatenating layers.

The script for this training and validation can be found at 'segmentation_u_net_pytorch.ipynb' file along with this submission.

Since, I was working alone, new to various technical stack I used for this project and lot of iterative adjustments to my training paradigm of U-Net, I reprioritized to concentrate more on U-net and as a result I had just 3 days to work on the Seg-Caps Network. Another major challenge is the shapes of vectors in the Seg-Caps paper which took a lot of time to understand and implement. The author had implemented the model using Keras. I used PyTorch to create the code and I did not find any reference code (clean and legible to understand code) on GitHub to refer and quickly implement this architecture for segmentation. The one I implemented from scratch is in the file 'segmentation_capsule_networks_pytorch.ipynb' but the training did not yield any results because the loss was not decreasing and fluctuating. This could be due to some bugs in my code which I am determined to debug and rectify after the submission/end of semester. The one I implemented is explained below.

Seg-Caps-Net:

I implemented the following architecture from the Seg-Caps paper.

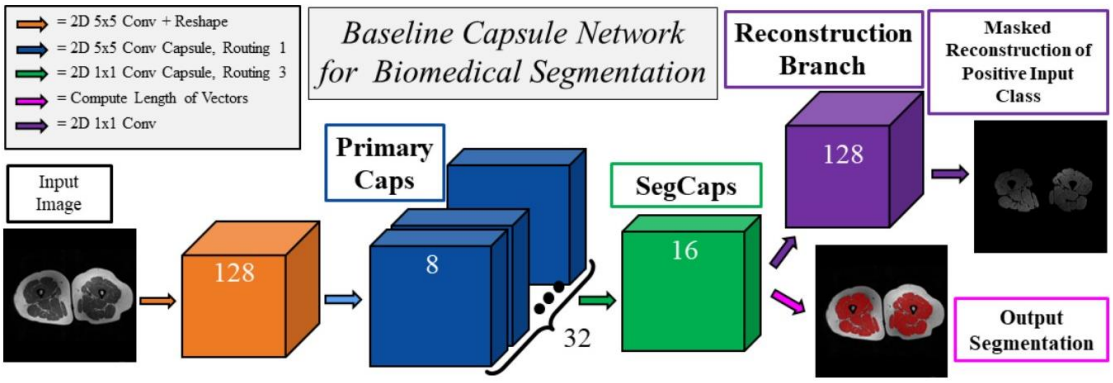


Figure 9: Basic Architecture from the Seg-Caps paper [1]

The architecture takes image input (128 x 128 x 4) and a convolution operation (3x3 kernel, stride = 1, padding = 'same as input') converts it to 128 x 128 x 16 channel features. Now I reshape this into a capsule with shape 128 x 128 x 1 x 16 denoting 1 capsule type with 128 x 128 capsules each of which

has a depth of 16. The Convolution Capsule operation is like a normal convolution operation. But here the convolution happens between the child capsule and each of the parent capsule.

Formulas involved in Capsule Networks:

- input to the parent capsule is the output from the child capsule → u_matrix
- prediction of the parent vector → u_hat_matrix = W x u_matrix; This weight W is learnt in the convolution operation from the backpropagation. This is like a shared kernel across different features in a traditional CNN.
- actual parent capsule output → v_matrix
- To calculate v_matrix we define a learnable parameter b_vector called as the routing co-efficient through the dynamic routing
- c_vector is obtaining by SoftMax(b). This c_vector says how much information this child capsule should route to a particular parent.
- There is a p_vector which is the predicted parent output from each of the child capsule → weight sum of the u_hat_matrix → $\sum_{i=1}^{n} c\_vector[i] * u\_hat\_matrix$
  where n is the number of parent capsules
- Then the predicted parent output will be normalized such the length of vectors (norm of vectors) is between 0 and 1. This can be thought of as the sigmoid like activation function in a traditional neural network.
- This normalization operation is known as squashing function → $\frac{\|p\_vector^2\|}{1+\|p\_vector^2\|} \frac{p\_vector}{\|p\_vector\|}$
- There is a concept called agreement (a_vector) between capsules, when the product of predicted output and the actual output is more it is said that there is a strong agreement between child capsule and parent capsule. Based on the agreement the b_vector is updated for the number of routing iterations.
- Mathematically, a_vector → v_matrix * u_hat_matrix
- b_vector → b_vector + a_vector

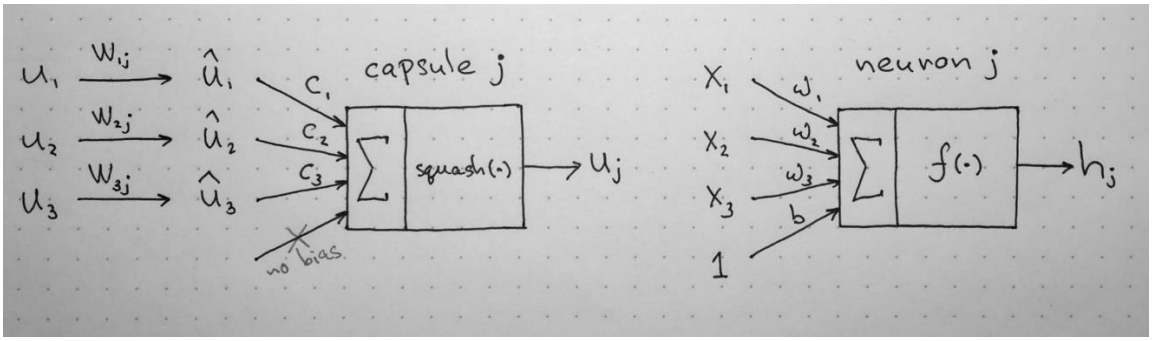


Figure 10: Analogy between Capsule Networks and Neural Networks [11]

3.3. Evaluation Strategy

U-Net:

The training of the model was performed using cross validation of 5-fold. The training data was augmented using random rotate and flipping of the images. The dataset had approximately 5800 images. PyTorch data loader module will apply augmentation as and when the image batch is generated for the model to consume.

Total image dataset size: ~5800

Training : Validation : Test → 80% : 10 % : 10%

Since we used CV, 90% of the dataset (~5200 images) were using in train test split according to the fold.

Every Fold: ~4200 images training : ~1000 images validation

Final model Each trained for 25 epochs and the best model was chosen. The validation dataset was used for early stopping.

The performance of the model was tested on the ~600 test images and the Dice similarity and IoU of each class were determined to show and compare the performance of the models.

Seg-Caps-Net:

The data split was same as that of the U-Net architecture, but the training was stopped because there were no noticeable decrease in the loss, and it was fluctuating. Also, the capsule networks were very heavy in computation hence the training took a long time. The number of routing iterations = 3.

The evaluation is performed based on two metrics namely the 3D Dice similarity co-efficient or the Sørensen–Dice coefficient and the Intersection over Union (IoU).

3.3.1. Sørensen–Dice coefficient (Dice)

Given 2 sets X and Y,

$$\text{Dice} = \frac{2\,|X \cap Y|}{|X| + |Y|}$$

3.3.2. Intersection over Union (IoU)

Given 2 sets X and Y,

$$\text{IoU} = \frac{|X| * |Y|}{|X| + |Y|}$$

Losses that were monitored and optimized:

- Cross Entropy Loss
- Dice Loss = 1 – Dice Co-efficient
- IoU Loss = 1 – IoU

4. Results

4.1. Configuration

4.1.1. Model Hyper Parameters experimented with:

- Architecture Name: U-Net
- L1 regularization co-ef: [1, 0.1, 0.01, 0.0001, 0.00001, 0]
- L2 regularization co-ef: [1, 0.1, 0.01, 0.0001, 0.00001, 0]
- K-Fold: 5-fold CV (N splits = 5)
- Batch size: [2, 4, 8, 16, 32]
- Epochs: [10, 25, 50]
- Class weights: { [0.1, 0.1, 0.5, 0.5, 0.5] , [0.1, 0.1, 0.25, 0.5, 0.5], [1, 1, 1, 1, 1] }
- train data size: 80% of the data (~4200 images)
- validation data size: 10% of the data (~ 800 images)
- test data size: 10% of the data (~ 800 images)
- initial weight matrix: Xavier Initialization, Normal Initialization
- activation function: ReLU (as described in the U-Net paper)
- loss function: Cross Entropy (Multi-Class), Dice Loss, IoU Loss

- Optimizer: [Adam, RMSProp, SGD]
- optimizer learning rate: [0.1, 0.01, 0.001, 0.0001, 0.00001]
- DL Framework: PyTorch

4.1.2. Model Hyper Parameters experimented with:

- Architecture Name: Seg-Caps
- L1 regularization co-ef: [0, 0.1]
- L2 regularization co-ef: [0, 0.1]
- K-Fold: 5-fold CV (N splits = 5)
- Batch size: [8, 16, 32]
- Epochs: [2, 5, 10]
- Class weights: { [0.1, 0.1, 0.5, 0.5, 0.5], [1, 1, 1, 1, 1] }
- train data size: 80% of the data (~4200 images)
- validation data size: 10% of the data (~ 800 images)
- test data size: 10% of the data (~ 800 images)
- initial weight matrix: Xavier Initialization, Normal Initialization
- b_vector initialization: Normal Initialization
- Number of Capsule Layers: [2, 3]
- Number of Dynamic Routing Iterations: [1, 2, 3]
- activation function: ReLU
- loss function: Cross Entropy (Multi-Class)
- Optimizer: [Adam, RMSProp, SGD]
- optimizer learning rate: [0.1, 0.01, 0.001, 0.0001, 0.00001]
- DL Framework: PyTorch

Best Model Config (Architecture 1):

- Architecture Name: U-Net
- L1 regularization co-ef: 0
- L2 regularization co-ef: 0
- K-Fold: 5-fold CV (N splits = 5)
- Batch size: 4
- Epochs: 25
- Class weights: [0.1, 0.1, 0.5, 0.5, 0.5]
- train data size: 80% of the data (~4200 images)
- validation data size: 10% of the data (~ 800 images)

- test data size: 10% of the data (~ 800 images)
- initial weight matrix: Xavier Initialization
- activation function: ReLU (as described in the U-Net paper)
- loss function: Cross Entropy (Multi-Class)
- Optimizer: Adam
- optimizer learning rate: 0.001
- DL Framework: PyTorch

Second Best Model Config (Architecture 2):

- Architecture Name: U-Net
- L1 regularization co-ef: 0
- L2 regularization co-ef: 0
- K-Fold: 5-fold CV (N splits = 5)
- Batch size: 8
- Epochs: 25
- Class weights: [1, 1, 1, 1, 1]
- train data size: 80% of the data (~4200 images)
- validation data size: 10% of the data (~ 800 images)
- test data size: 10% of the data (~ 800 images)
- initial weight matrix: Xavier Initialization, Normal Initialization
- activation function: ReLU (as described in the U-Net paper)
- loss function: Cross Entropy (Multi-Class) + Dice Loss + IoU Loss
- Optimizer: Adam
- optimizer learning rate: 0.001
- DL Framework: PyTorch

Best Model Training Loss Curves

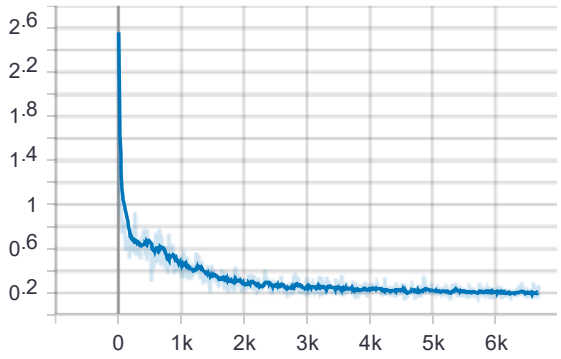


Figure 11: Cross Entropy Loss

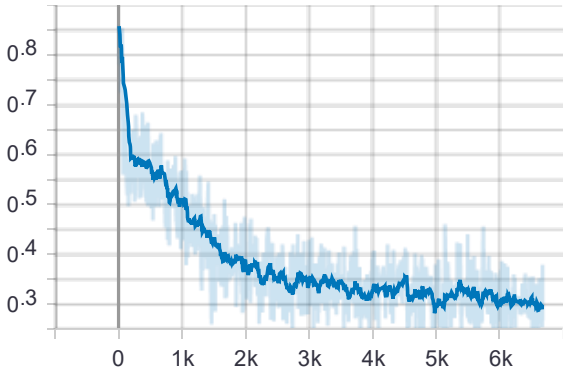


Figure 12: Dice Loss

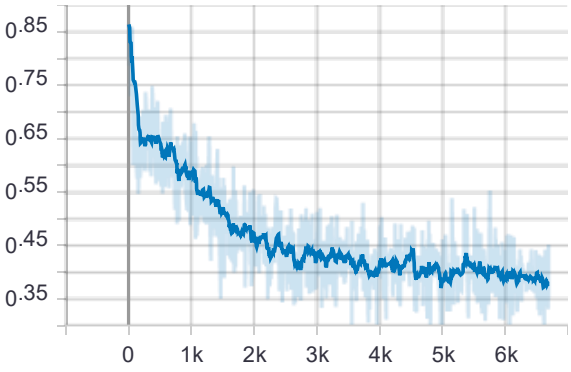


Figure 13: IoU Loss

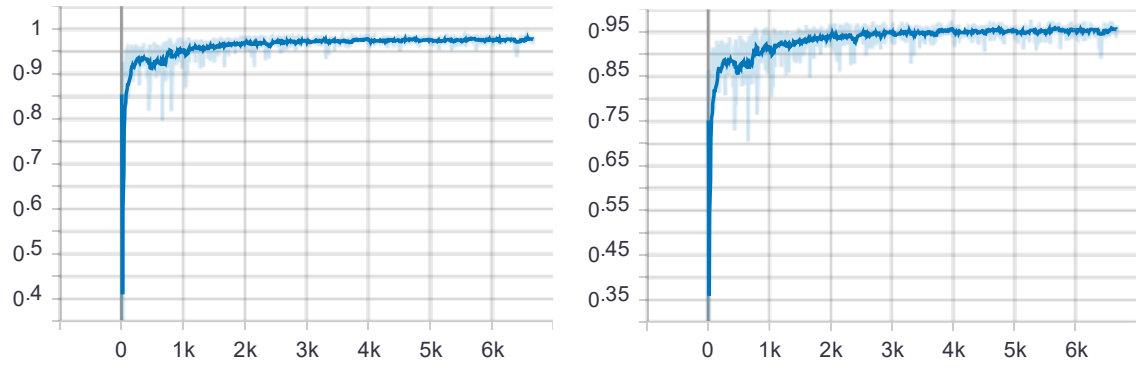


Figure 14: Training - Brain Background Dice Co-efficient and IoU

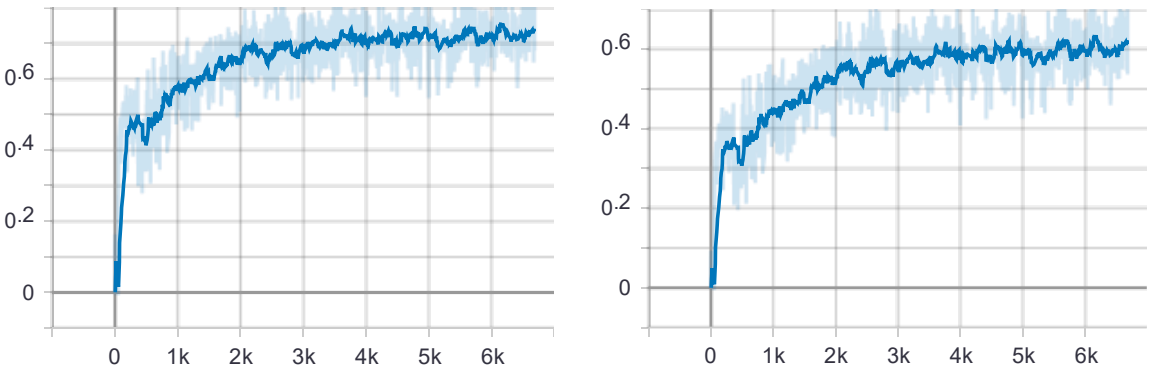


Figure 15: Training - Edema Dice Co-efficient and IoU

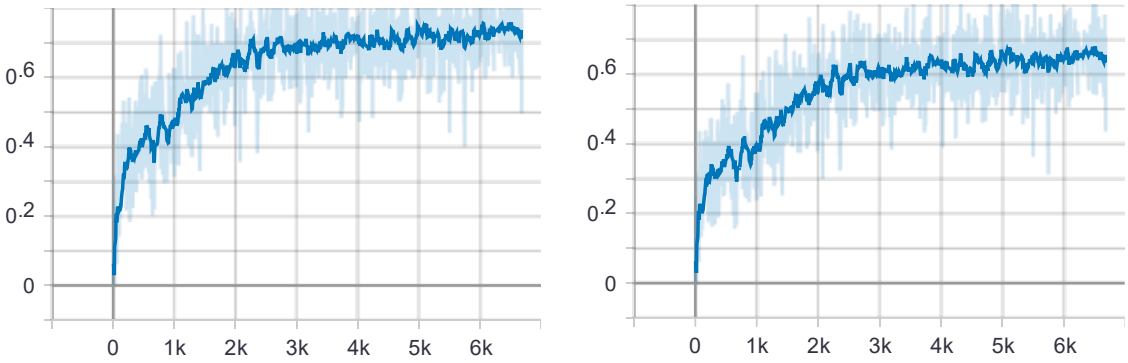


Figure 16: Training - ET Dice Co-efficient and IoU

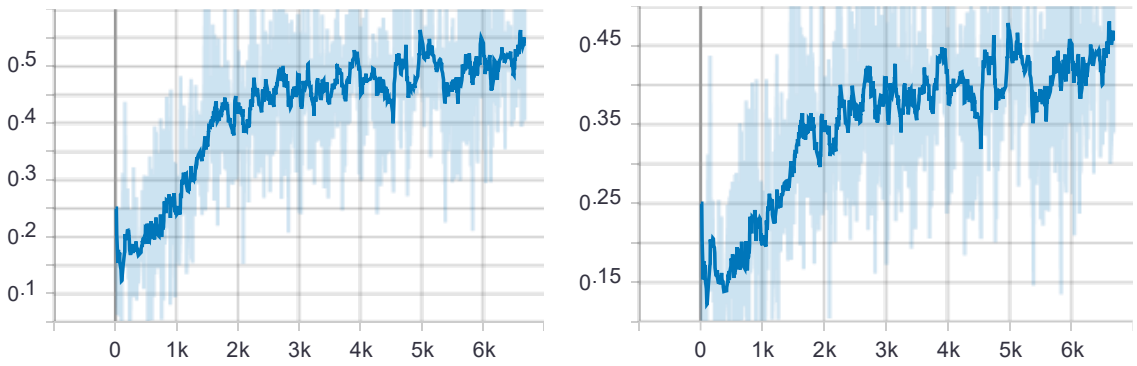


Figure 17: Training - NET Dice Co-efficient and IoU

Box Plot comparison between test results of 2 best U-Net models

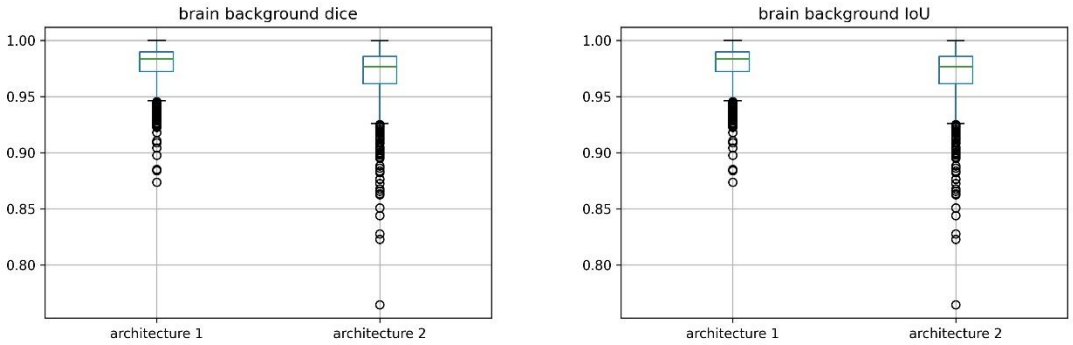


Figure 18: Brain Background

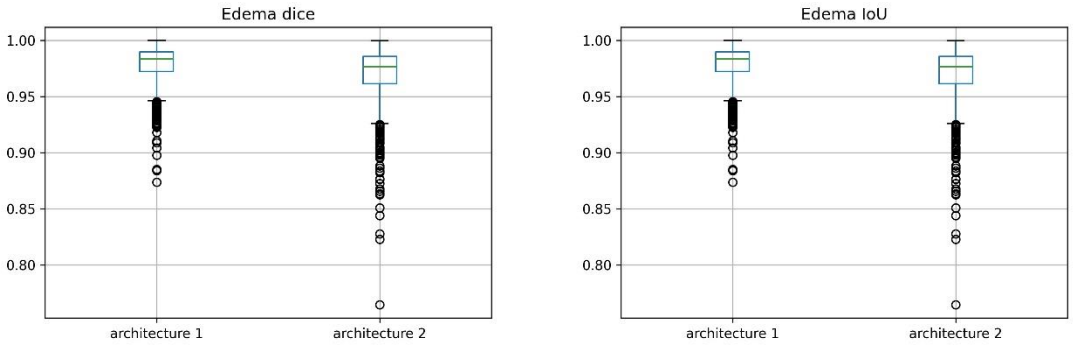


Figure 19: Edema

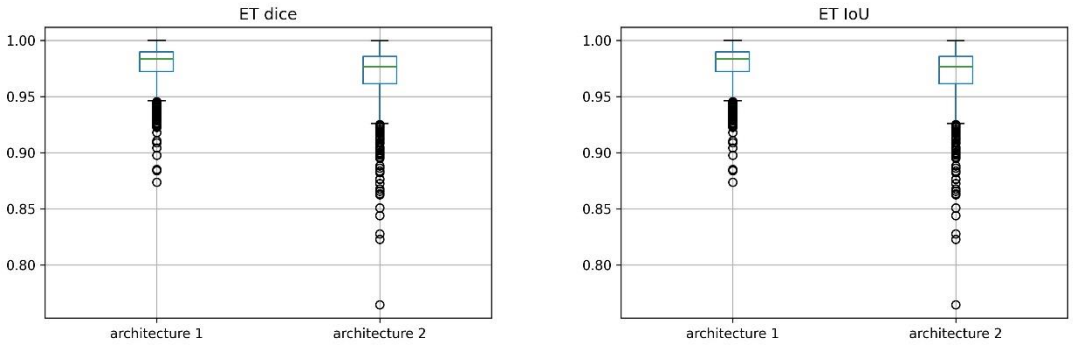


Figure 20: Enhancing Tumor

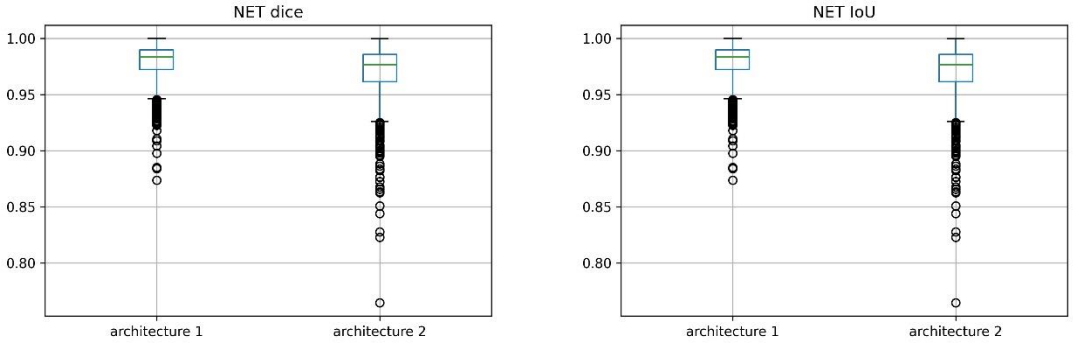


Figure 21: Non-Enhancing Tumor

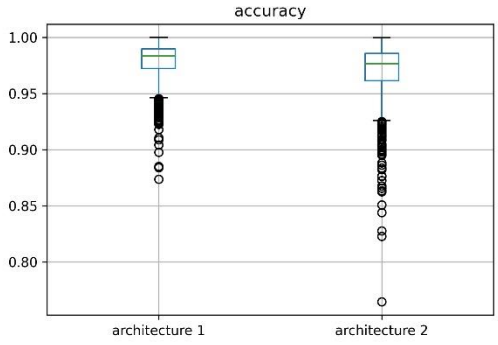
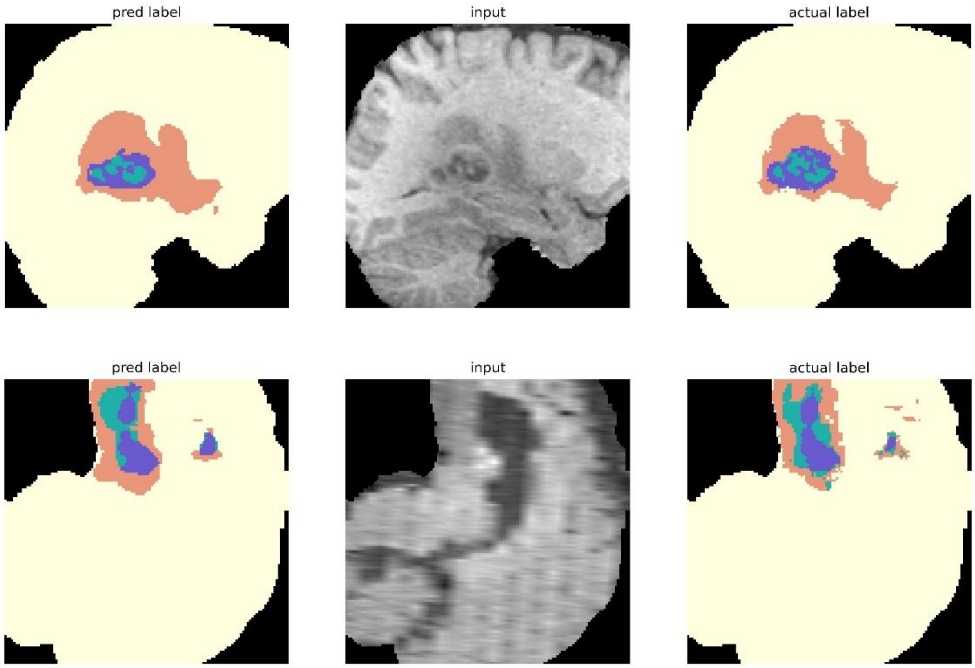


Figure 22: Test – Accuracy

Architecture\ Metric	Edema Dice	Edema IoU	ET Dice	ET IoU	NET Dice	NET IoU	Brain BG Dice	Brain BG IoU	Accuracy
Architecture 1	**74.8 ± 18.4**	**62.6 ± 19.6**	**72.5 ± 32.1**	**64.8± 32.2**	**51.8 ± 36.9**	**43.8 ± 35.9**	**97.8 ± 1.64**	**95.8 ± 3.06**	**95.8 ± 2.64**
Architecture 2	71.2 ± 19.1	58.3 ± 20.0	68.0 ± 34	60.0 ± 33.3	48.2 ± 37.5	40.5 ± 36.0	96.9 ± 2.02	94.2 ± 4.3	94.6 ± 3.3

Table 1: Test Metric Comparison



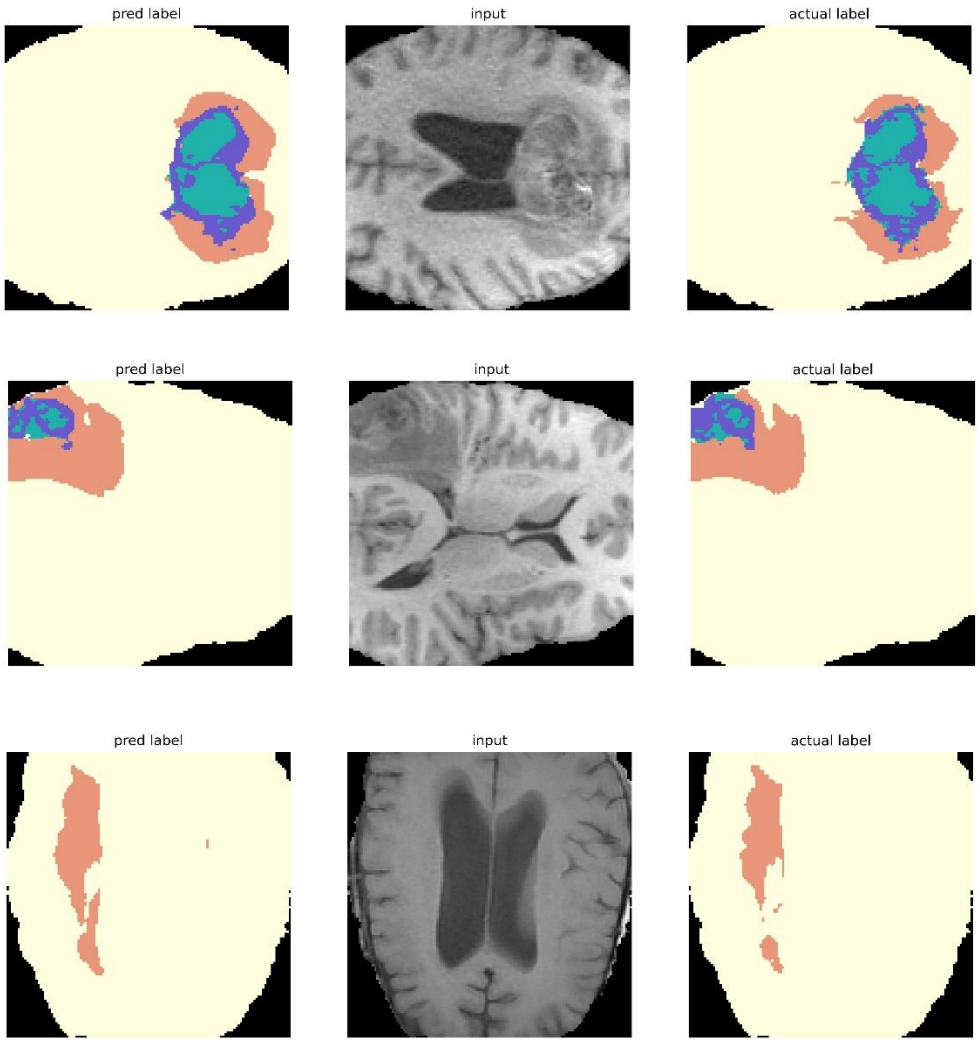


Figure 23: Model Outputs on Test data (Predicted Labels, Input, Actual Labels)

5. Conclusions

5.1. Discussion of the Results Section

The Fig. 10 to 17 are the training loss curves for the best model architecture and configuration for the U-Net. This training was done on full training data (training + validation). The configuration was arrived after I experimented with various hyperparameter settings, cross validation and suitable loss functions.

The box plots from Fig. 18 to 22 all denote the comparison in performance between the architecture 1 and 2. The table below summarizes the performance in numbers (mean ± std). It can clearly be seen that Architecture 1 performs better than Architecture 2.

Also, we can note that the accuracy in both the cases is good. But since it is a segmentation task with unbalanced dataset the accuracy metric does not say more about the performance. The individual Dice and IoU metric of each of the tumor classes are a better measure for this task.

5.2. Things that worked

- Class wise tumor segmentation performance are on a average 70s, 65s, 60s for Edema, ET and NET respectively. This is clearly an indication of the percentage of their existence in the training dataset. The Edema was found more than Enhancing Tumor which in turn was present more than Non-Enhancing Tumor. This is clearly a problem of class imbalance in the training dataset.
- This could be improved by giving more weights to the classes that are less in number. That is, a class weight of [0.01, 0.01, 0.5, 0.75, 1] would have had a better performance than the current best which is [0.1, 0.1, 0.5, 0.5, 0.5].
- Another possibility of improved performance could be to have a stricter criterion while creating the dataset and then compensating with having better data augmentation techniques like oversampling of under presented classes.
- The loss curves decreased as expected especially the cross-entropy loss.
- The padding and concatenating the layers in the U-Net architecture worked well as it retained the shape of the output label and was easy to train. This ensured feature propagation between the layers (contracting path and expansive path).
- The idea of combining images from different layers/axis worked because I got more real data with a different configuration and from the same distribution.
- Bringing the image pixel intensity to zero and one was a better choice as it had a stabilized training and agnostic across different modality. Plus, the activation functions are ReLU and it is better to have positive inputs as negative values will simply be discarded.

5.3. Things that did not work

- The cross-entropy loss + dice loss + IoU loss combination did not reduce as expected when compared to just using CE loss. I expected the combination would yield a better performance. This could be because the dice loss and IoU loss are not a smooth minimization curve for optimization which could have problems while performing backpropagation. Sometimes they

behave like non-differentiable functions. This could be why the combined loss backprop was rougher than using just the CE loss.

- The Seg-Caps-Net did not work as expected. I faced the following challenges when implementing Seg-Caps.
    - Shapes of the tensors were highly confusing from the papers published
    - The capsule network was very heavy on computation and even a forward pass was slower than a backprop in a U-Net.
    - Lack of clarity in computing the length of the vectors to determine the class output.
    - The reconstruction loss that I implemented in PyTorch did not work effectively. Had some issues like the node detaching from the graph resulting in back-prop not working as intended.

6. Individual Tasks

Since I am working on this project alone all tasks were done by me. The following are the tasks:

- Researching to find a suitable dataset
- Researching for semantic segmentation algorithms and models to work with
- Studying journals, watching videos of author's presentations and model building, technical fundamentals for the State-of-the-art models used
- Worked on the proposal and submitted the same
- Set up the complete ML pipeline
- Analysis and visualization of the BraTs dataset
- Pre-processing the dataset
- Storing it in a database to be streamed for training and testing
- Discussion with Professor Predrag for suitable normalization and standardization techniques, image input to the architecture (initially mentioned above as 'My Iterative Efforts')
- Model building in PyTorch from scratch by referring various existing implementations, blog posts and repositories
- Storing the models, results, and plots during training
- Setting up of TensorBoard logger for online training tracking and visualization
- Post training on different hyperparameter combinations, training the model with the best configuration on complete training data (training + validation)
- Analysis of the results and writing of this report

I took up this project to stay on the research track and whatever the work I did towards this project was with the same mindset. Currently, I was not able to complete the Seg-Caps-Net to get legible results but have

learnt a lot on the process. I wish to continue the journey of what I started in this class and will continue to work after the project submission to improvise the novelties I proposed. If it works with Prof. Predrag, I would like to extend this project as my Master's Thesis. I would be happy to be advised by him for the same.

7. Future Work

- Writing a clean and intuitive code to explain the Seg-Caps clearly and aim for reproducibility (possibly with a blog post to explain them clearly)
- Write the Deconvolution Capsule block in PyTorch
- Extend the Seg-Caps for Multi class semantic segmentation task
- I read another paper that proposed the use of EM algorithm for Dynamic Routing. I will try to incorporate the same if the previous step can be achieved.
- Research more on Loss functions as I really feel it is the key for solving any Deep Learning problem

8. References

[1] Rodney LaLonde, Ziyue Xu, Ismail Irmakci, Sanjay Jain, Ulas Bagci, "Capsules for biomedical image segmentation", Medical Image Analysis, Volume 68, 2021, 101889, ISSN 1361-8415, https://doi.org/10.1016/j.media.2020.101889.

[2] Simon Graham, Quoc Dang Vu, Shan E Ahmed Raza, Ayesha Azam, Yee Wah Tsang, Jin Tae Kwak, Nasir Rajpoot, "Hover-Net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images", Medical Image Analysis, Volume 58, 2019, 101563, ISSN 1361-8415, https://doi.org/10.1016/j.media.2019.101563.

[3] Ronneberger, O., P. Fischer and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." ArXiv abs/1505.04597 (2015)

[4] S. Sabour, N. Frosst, and G. Hinton. Dynamic Routing Between Capsules, arXiv preprint arXiv:1710.09829.

[5] https://rodneylalonde.wixsite.com/personal/post/algorithms-and-applications-of-novel-capsule-networks

[6] https://en.wikipedia.org/wiki/Image_segmentation

[7] Biomedical Image Segmentation Dataset Repository: https://arxiv.org/abs/1902.09063, Medical Segmentation Decathlon (medicaldecathlon.com)

[8] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, et al. "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)", IEEE Transactions on Medical Imaging 34(10), 1993-2024 (2015) DOI: 10.1109/TMI.2014.2377694

[9] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J.S. Kirby, et al., "Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features", Nature Scientific Data, 4:170117 (2017) DOI: 10.1038/sdata.2017.117

[10] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, et al., "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge", arXiv preprint arXiv:1811.02629 (2018)

[11] https://pechyonkin.me/capsules-2/

[12] LaLonde, Rodney. (2018). “Capsules for Object Segmentation”. https://arxiv.org/pdf/1804.04241.pdf

[13] Project Proposal: “Recognition and Representation of Facial Features using a Multi-Layer Capsule Network”. Authors: Vikram Shenoy, Alexander Chowdhury, Harry Hartenstine

[14] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel,

1989. Backpropagation applied to handwritten zip code recognition, Neural Comput. 1, 4 (December 1989), 541–551. DOI:https://doi.org/10.1162/neco.1989.1.4.541

[15] https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient

[16] https://en.wikipedia.org/wiki/Hausdorff_distance

[17] https://autonomous-driving.org/2018/07/15/semantic-segmentation-datasets-for-urban-driving-scenes/

[18] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth: “The Cityscapes Dataset for Semantic Urban Scene Understanding”, 2016; [http://arxiv.org/abs/1604.01685 arXiv:1604.01685]

[19] https://en.wikipedia.org/wiki/Convolutional_neural_network

[20] https://en.wikipedia.org/wiki/Capsule_neural_network

[21] Xiaopeng Yang, Jae Do Yang, Hong Pil Hwang, Hee Chul Yu, Sungwoo Ahn, Bong-Wan Kim, Heecheon You, “Segmentation of liver and vessels from CT images and classification of liver segments for preoperative liver surgical planning in living donor liver transplantation”, Computer Methods and

Programs in Biomedicine, Volume 158, 2018, Pages 41-52, ISSN 0169-2607, https://doi.org/10.1016/j.cmpb.2017.12.008.

[22] Qing Huang, Jinfeng Sun, Hui Ding, Xiaodong Wang, Guangzhi Wang, “Robust liver vessel extraction using 3D U-Net with variant dice loss function”, Computers in Biology and Medicine, Volume 101, 2018, Pages 153-162, ISSN 0010-4825, https://doi.org/10.1016/j.compbiomed.2018.08.018.

[23] https://github.com/wiqaaas/youtube/blob/master/Deep_Learning_Using_Tensorflow/Image_Segmentation_using_U-Net/Image%20Segmentation%20using%20U-Net%20for%20MRI%20(3D%20Images).ipynb

[24] https://github.com/cezannec/capsule_net_pytorch/blob/master/Capsule_Network.ipynb

[25] https://github.com/lalonderodney/SegCaps

[26] https://www.kaggle.com/bigironsphere/loss-function-library-keras-pytorch#Focal-Loss