**Project Report 2**

**Robust Split Federated Learning for U-shaped Medical Image Networks**

**Group Member and Contribution:**

**Group No: 8**

|  |  |  |  |
| --- | --- | --- | --- |
| **Team member Name** | **Team member Email IDs** | **R# IDs** | **Contribution** |
| Venkata Ramana Sathivada | venkata.sathivada@ttu.edu | R11850665 | * Separating the model into 3 different notebook. FCT Division into Contraction, Expansion and Final sigmoid layer |
| Hemanth Reddy Nagireddy | henagire@ttu.edu | R11804804 |
| Madhu Kiran Jalla | mjalla@ttu.edu | R11848145 | * Set up of forward and backword propagation from the head->body->tail and reverse |
| Sri Ram Koppaku | ksriram@ttu.edu | R11842335 |
| Charishma Chokkakula | cchokkak@ttu.edu | R11869144 | * TCP socket communication between the model to exchange information during forward and back propagation |
| Manisha Ratna | mratna@ttu.edu | R11847768 |
| Manoj Metuku | mmetuku@ttu.edu | R11847308 | * Setup the default environment, including resources and devices simulate notebook run independent of each other |
| Mani Madhuri Gokaraju | mgokaraj@ttu.edu | R11848127 |

**Problem Statement:**

Creating a U-Shaped Medical Image segmentation that uses both Federated and Split learning to increase the model privacy with decentralized machine learning and reducing the computation cost on the agents. The problem also is defined to tackle the drift between local and global model and their parameters that is caused due to data Homogeneity that is mostly encountered in Medical data ML cases.

**Problem Solution Status:**

The Status of the given problem is as:

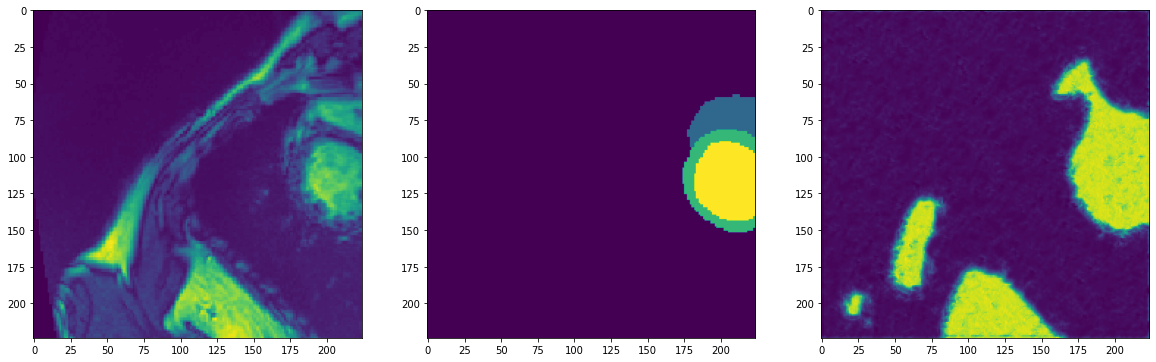
1. After dividing the FCT main model in to 3 parts which is `head`, `body` and `tail` which basically the Head consisting of the Contraction layer of the U-Type FCT network, the Body being the Expansion layer of the U-Type FCT network and tail being the final, sigmoid layer of the above segmentation Model

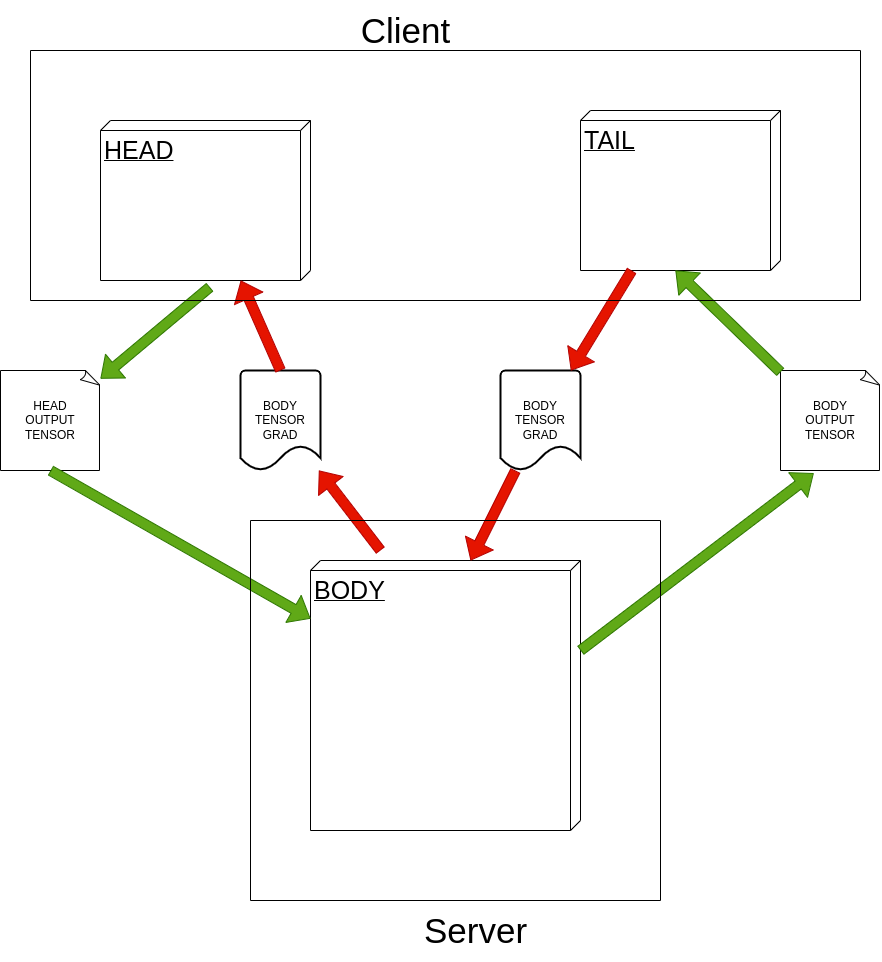
2. Saving in HDF5 file format of the forward propagation from head in “head\_forward\_pass” and same is used as input for the Body server which generates the “body\_forward\_pass” to be used by the tail model as input. The train label is also sent to tail for loss computation as “train\_value”, after the optimizer is step is computed the generated grad of the input is saved as “tail\_back\_prop” and the same would be used in the server containing Body model, to calculate its gradient. Further once the gradient is calculate the input tensor gradient is saved as “body\_back\_prop” that would be again used in the “Head”model client and used to calculates its gradient.

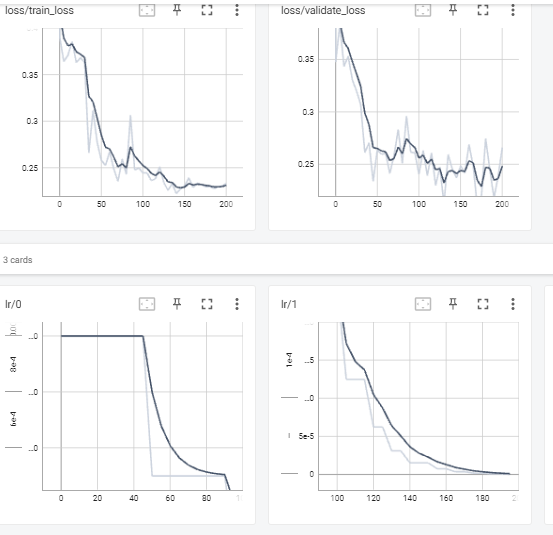
3. Implementation of TCP Client-Server communication module to send the signal to read the respective layer value or back prop gradient value.

**Experiment Results:**

1. After decentralizing the code the FCT head and tail was able to use CPU memory and the Body was assigned GPU memory so because of which for the 250 epoch we could use 4 batch size instead of just 2. The time taken was a bit longer due to the file I/O operation but overall operation was improved.

  
  
Worflow of the overall model break up is mentioned shown below





The steps followed are as follows

1. The overall data for training is split in to 2 sets consisting of overall 80% of data selected for Client 1 and Client 2
2. For the Client 1 we created a Head Model which consists on the FCT encoder layers where input image data will be passed on to each layer in pyramid format
3. For the Overall Computational body the Body Model is created that consist of a single encoder without the image data input and rest of the similar decoder layer of FCT in same number as the Head Model
4. For the Client 1 the Tail model would be a sequential convolution, activation and sigmoid layer that would just give the final out to be compared to the original mask data vs predicted.
5. The input data goes into the Head layer and after forward pass within the model saves the output to a “head\_forward\_pass.hdf5” file and train data in “train\_values.hdf5” file
6. The body would read the “head\_forward\_pass.hdf5” file for input and after running the forward propagation would output the tensors which would be saved to “body\_forward\_pass.hdf5” file
7. The tail would read the “body\_forward\_pass.hdf5” file and calculate the prediction of segmentation which would be then compared to “train\_values.hdf5”
8. The loss calculated by tail above can be used to compute gradient and the tail grad can be saved in the “tail\_back\_prop.hdf5” file
9. The gradient of the tail model can be read by the Body Model for the file “tail\_back\_prop.hdf5” and same can be used to calculate its own gradient using back propagation method and its body gradient can be saved into “body\_back\_prop.hdf5” file
10. The gradient of body in “body\_back\_prop.hdf5” file can be read by Head Model and it gradient can be calculated by using back propagation method.
11. This signifies one epoch is completed and a complete forward pass and a complete back propagation has been carried out, and all the model parameters(weights and biases) would have been adjusted.
12. We can carry the same steps (from a.) all over again for a next set of iteration/epoch and we would be iterating it over around 200-300+ epochs.

**References:**

[1] We are taking the code from the GitHub Link:

<https://github.com/thanos-db/fullyconvolutionaltransformer>.

This link is the code implemented on the paper – “The Fully Convolutional Transformer for Medical Image Segmentation”. This code is being used, implemented and improved further to include federated and split learning. The approach to use this is taken from this paper i.e. – “**Robust Split Federated Learning for U-shaped Medical Image Networks**”.

[2] Paper: Robust Split Federated Learning for U-shaped Medical Image Networks.

The code reference is being used from this paper: The Fully Convolutional Transformer for Medical Image Segmentation.