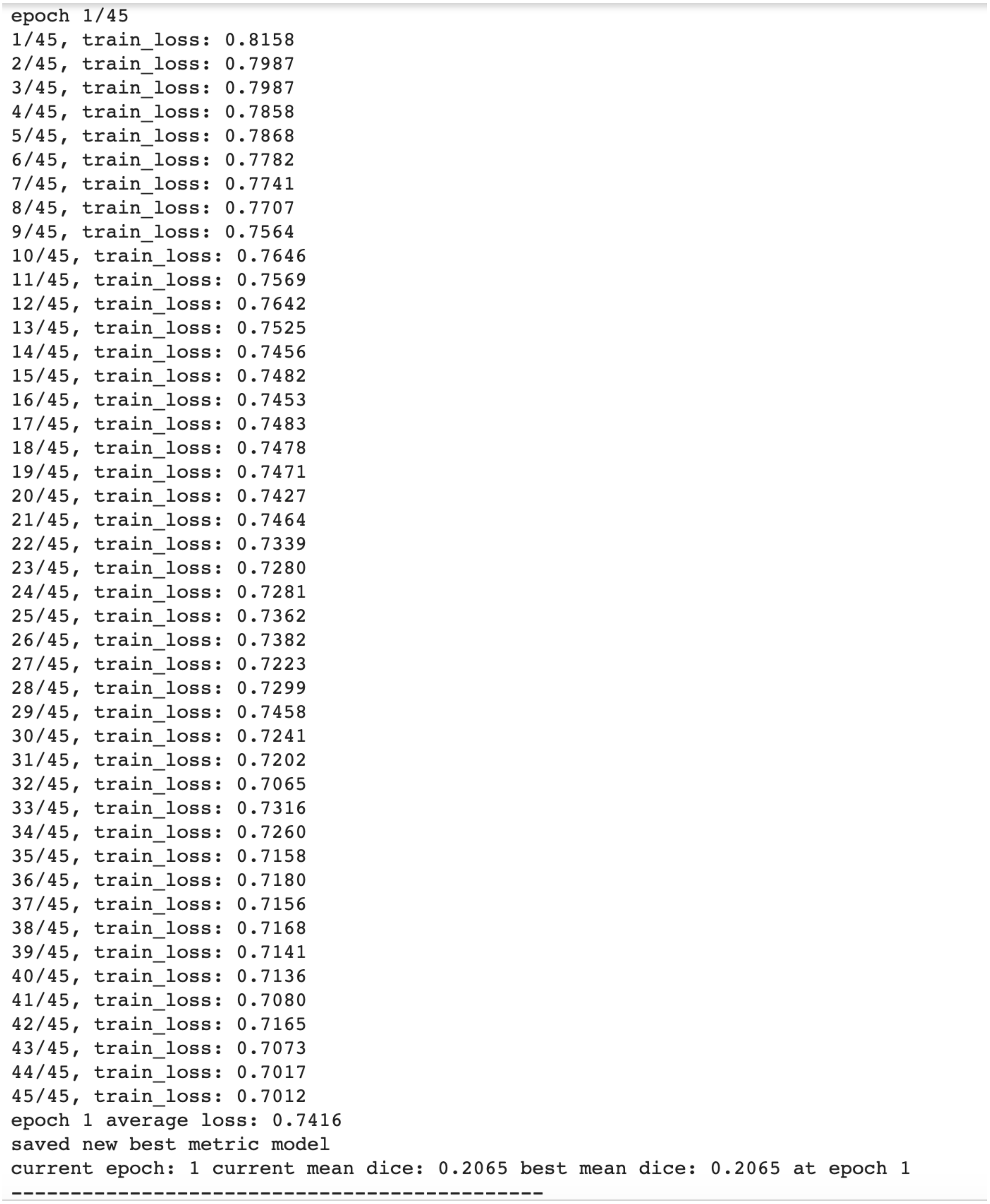
Abstract

This summer (June through August 2021), I worked with Professor Joshua Stough of the Computer Science department at Bucknell University on semantic segmentation of echocardiograms. The goal of this project is to train a neural network model on the CAMUS data set of image echocardiograms and apply the EchoNet video echocardiogram data on the model to see if the original accuracy is retained. The first dataset, CAMUS, was split into training and testing folders, and each further into separate subfolders, one for each patient. Each patient's folder under training contained two chamber and four chamber end diastolic and end systolic image frames along with labels for frames. However the data in the testing folder did not have the associated labels of the frames, and hence was disregarded for this project. The second dataset, EchoNet, consisted of video echocardiograms. The first step of the project was to split this data into test, train, and validation subsets. I implemented a function in Collab utilizing the KFold cross validation function from the *scikit learn* library for five different folds and created test, train, and validation sets for each patient.

The next step was to retrieve the frames and labels from each patient folder and populate them into the respective lists. I only considered the four chamber end systolic and end diastolic views. Once these frames and their labels were retrieved from each patient folder, I populated them into a list of dictionaries. Each dictionary contains 'image' and 'label' values. The next steps were populating the lists into datasets, one for train set and validation set), and applying transforms on the data. Throughout the project, I utilized Project Monai, an open source framework for medical image AI related tasks. Using Monai's transforms framework, I loaded the frames, resized them, and converted the NumPy image array to Torch tensors. After populating the dataset with the transformed data, I loaded the datasets into data loaders sourced from PyTorch's data utilities. I utilized Monai's UNet architecture for the neural network which has approximately 6.5 million trainable parameters, DiceLoss for loss computation, and Adam optimizer which was sourced from torch. The model trained over 45 epochs. The loss was calculated in the training phase and the model accuracy was calculated in the validation phase, using Monai's function to compute Dice score as well as to save the new model when a new highest Dice score is reached. Given that the KFoldFunction returns five different folds, I trained and saved five separate neural networks, each trained on a different fold. The next step was to test each model on EchoNet. As the EchoNet data is in video form, the end systolic and end diastolic frames needed to be extracted. I used EchoNet's utilities to instantiate a dataset that contains NumPy array data for the end diastolic/end systolic frames and their corresponding labels, and extract these frames and labels. The next was applying transformations on the EchoNet images, resizing the frames and labels to match the CAMUS image size, and modifying the image tensors to match the statistical moments of the CAMUS images ([0,255] from [-1,1]). Again, I populated a list of dictionaries using the frames and labels with the 'image' and 'label' keys and iterated over the data loader for model testing. Monai's *meandice* compute function was used to calculate the accuracy. EchoNet labels vary from CAMUS labels as EchoNet labels have 2 classes (background, LV) with CAMUS having 4 classes (background,

LV, etc.). Thus, the Dice score was computed on the LV class. The process for testing on EchoNet data was done on each of the five models.

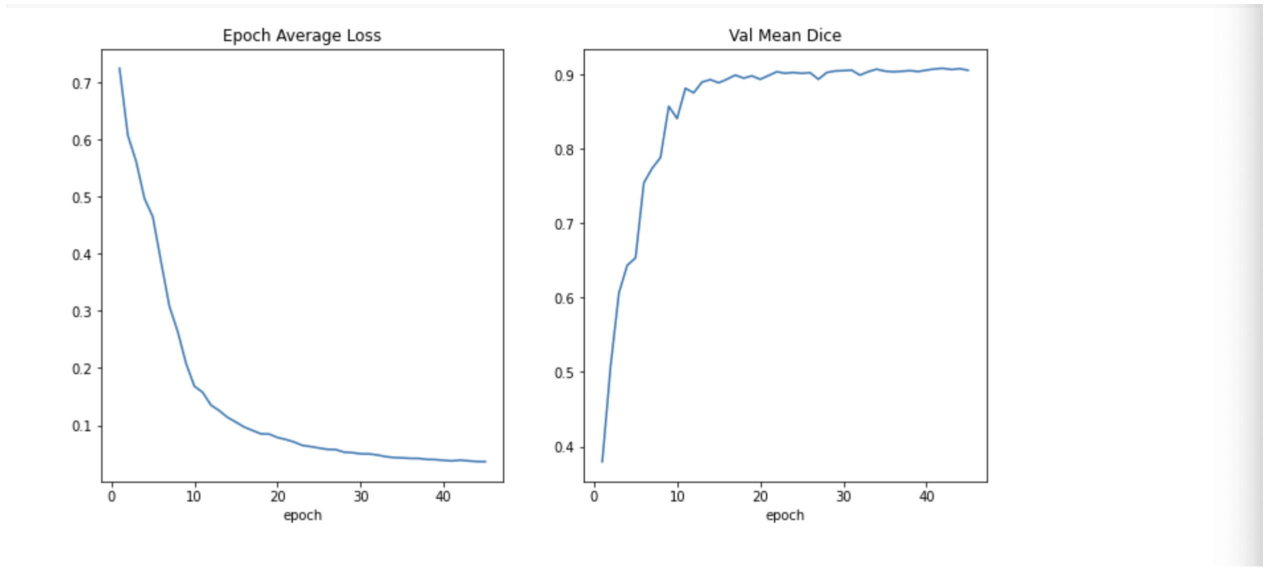
Below are images/graphs of the model training/testing process on the first fold of data.

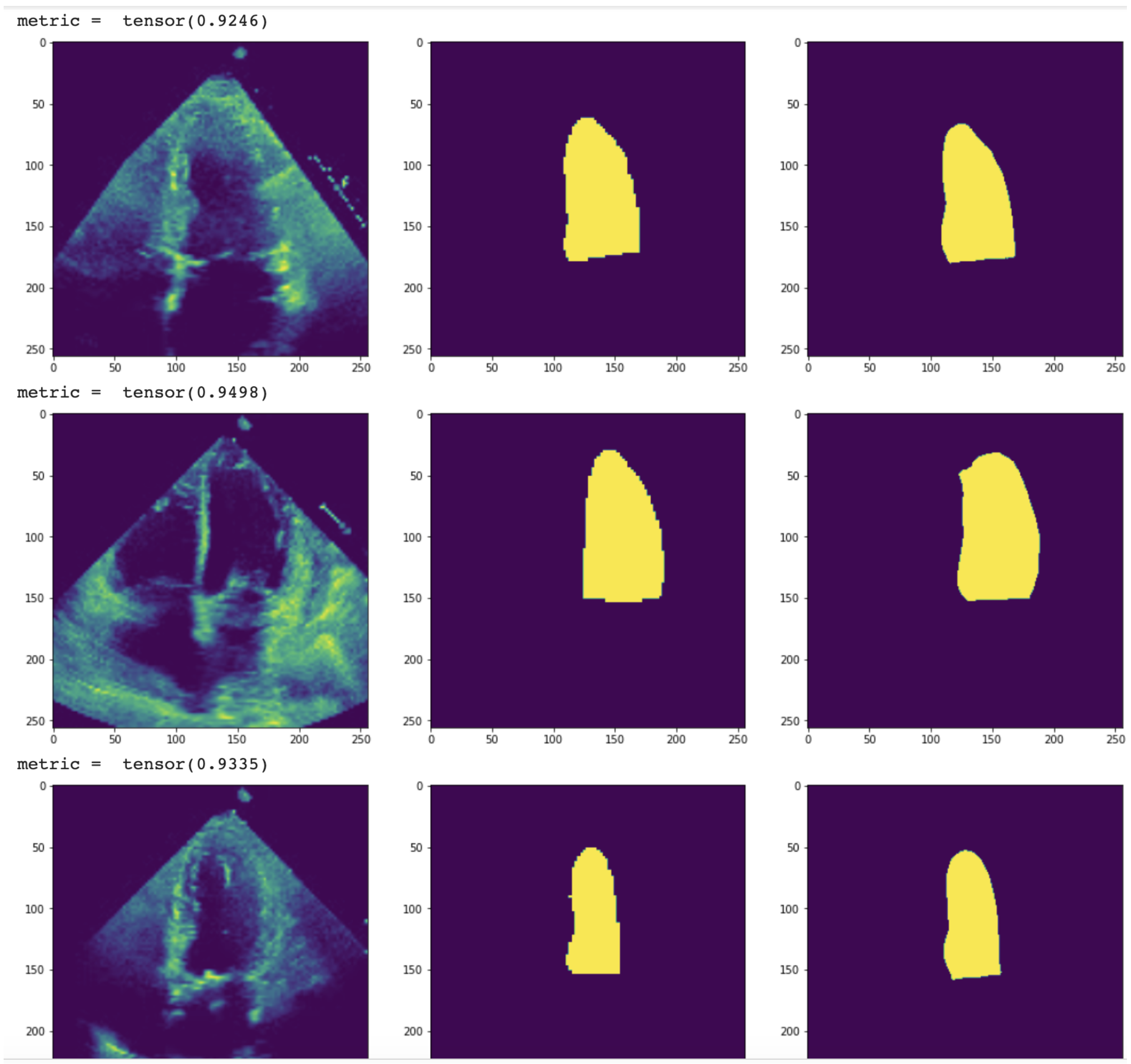


```
epoch 1/45
1/45, train_loss: 0.8158
2/45, train_loss: 0.7987
3/45, train_loss: 0.7987
4/45, train_loss: 0.7858
5/45, train_loss: 0.7868
6/45, train_loss: 0.7782
7/45, train_loss: 0.7741
8/45, train_loss: 0.7707
9/45, train_loss: 0.7564
10/45, train_loss: 0.7646
11/45, train_loss: 0.7569
12/45, train_loss: 0.7642
13/45, train_loss: 0.7525
14/45, train_loss: 0.7456
15/45, train_loss: 0.7482
16/45, train_loss: 0.7453
17/45, train_loss: 0.7483
18/45, train_loss: 0.7478
19/45, train_loss: 0.7471
20/45, train_loss: 0.7427
21/45, train_loss: 0.7464
22/45, train_loss: 0.7339
23/45, train_loss: 0.7280
24/45, train_loss: 0.7281
25/45, train_loss: 0.7362
26/45, train_loss: 0.7382
27/45, train_loss: 0.7223
28/45, train_loss: 0.7299
29/45, train_loss: 0.7458
30/45, train_loss: 0.7241
31/45, train_loss: 0.7202
32/45, train_loss: 0.7065
33/45, train_loss: 0.7316
34/45, train_loss: 0.7260
35/45, train_loss: 0.7158
36/45, train_loss: 0.7180
37/45, train_loss: 0.7156
38/45, train_loss: 0.7168
39/45, train_loss: 0.7141
40/45, train_loss: 0.7136
41/45, train_loss: 0.7080
42/45, train_loss: 0.7165
43/45, train_loss: 0.7073
44/45, train_loss: 0.7017
45/45, train_loss: 0.7012
epoch 1 average loss: 0.7416
saved new best metric model
current epoch: 1 current mean dice: 0.2065 best mean dice: 0.2065 at epoch 1
---------------------------------------------
```

(Above) Training Losses and Mean Dice Score of One Epoch During Model Training

Average Loss and Mean Dice Trends Over All 45 Epochs

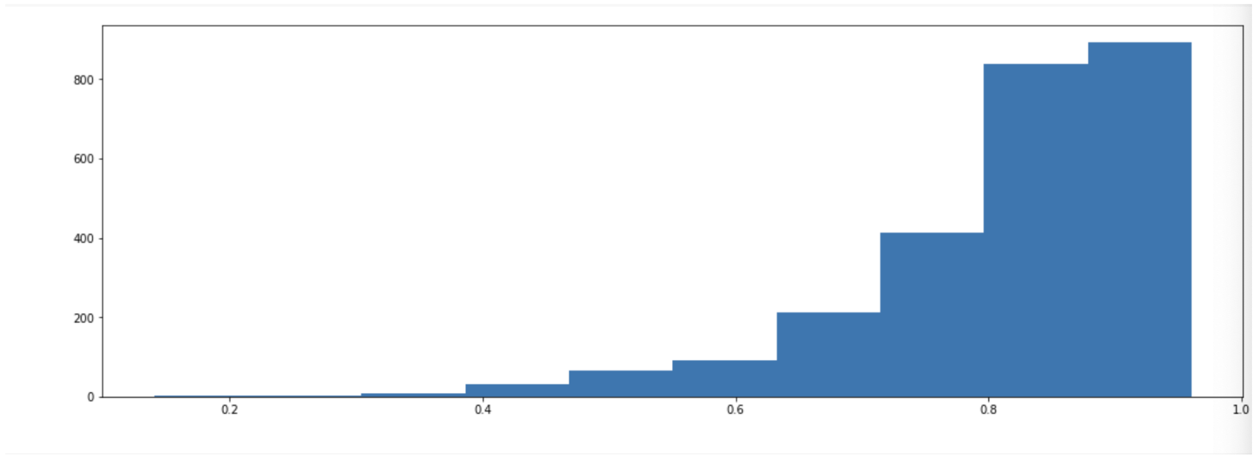




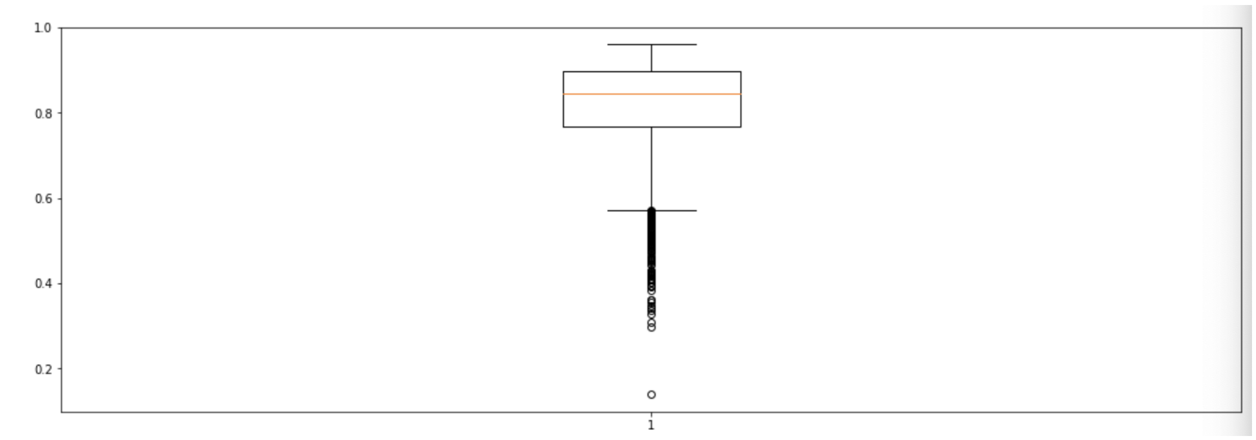
```
metric =  tensor(0.9246)
metric =  tensor(0.9498)
metric =  tensor(0.9335)
```

(Above) Leftmost is EchoNet frame passed through model, middle is EchoNet label, rightmost is Model's label outputs (along with the dice score on the top)

Histogram of Mean Dice Scores from Model Testing



Box plot of Mean Dice Scores from Model Testing



Result

There was a wide range of mean dice scores when the model was tested on EchoNet data, as can be seen in the above graphs. While some model outputs were highly inaccurate (around the 0.2-0.4 mark), there was a far larger concentration of considerably more accurate outputs (hovering around the 0.8 mark).

Link to Final Product

[Project Folder]

Reflection

Highlights

Given that this was a research project, it required me to work independently, but I felt the balance between that and the guidance from my mentor was perfect. The project first started with my mentor explaining the objective and then with me understanding the concepts of semantic segmentation or domain adaptation through reading the research papers that my mentor shared and Medium's Towards Data Science articles. Once I familiarized myself with the subject, I explored the CAMUS and Echonet datasest. This meant visualizing the frames and labels using Pyplot as well as testing various NumPy methods on the data which was in NumPy array format. I then implemented a simple segmentation workflow. Using the Project Monai framework, I reviewed their tutorials on both abdominal CT and brain tumor segmentation. Their code provided me with a foundation to build upon. Implementing the neural network took the longest time out of all the tasks because there were a lot of moving parts as well as several errors that took time to understand and fix. Each task built on what I learned previously and so it was important to take small steps and completely understand what I was building.

Collaboration

I met with my mentor weekly for an hour on Google Meet. During these meetings, I would go over the tasks I completed the past week, work through any errors I was facing, learn about programming “theory” in the context of the project, and discuss the next steps. Collaboration also happened asynchronously over Discord, where I shared screenshots of my code and code outputs to provide additional updates, seek help to fix bugs and clarify any doubts I had regarding my tasks and next steps.

Skills

This research project furthered my passion and interest in the field of machine learning and medical image analysis. I learned about cutting edge research topics such as semantic segmentation and domain adaptation. This project was also incredibly effective in improving my technical skills. Compared from when I started the task to now, my understanding of medical image processing has increased greatly, along with using machine learning libraries, specifically, NumPy and PyTorch. In the area of data, I learned to visualize and manipulate image data in the form of NumPy arrays and Torch tensors. I also gained an understanding of the different data transformations and learned which ones to apply through the Monai, NumPy, and PyTorch documentation and related online forums. Regarding the model, I learned about the different network architectures I could apply to a semantic segmentation task (DenseNet, ResNet, UNet) through the Monai documentation for Network Architectures. After choosing the UNet architecture, I understood how to modify the network architecture, for example, increase the number of parameters as a way to improve model accuracy. I also learned about metrics that determine the quality of a model such as loss function and dice scores, and the output processing needed for the dice function. For testing the model on the EchoNet data, I learned how to extract a specific frame from the echocardiogram video given the frame numbers, which I retrieved from a CSV file, using the OpenCV library. I also improved my Google Collab and Python skills.

I also significantly improved my debugging skills under the guidance of my mentor. For example, when my dice score computation was returning incorrect values, my mentor and I went back through the documentation to see what format the input the dice score function expected. We worked our way backwards to make sure the output was in that format and realized we had to use one_hot, argmax, etc.

Challenges/Strengths/Weakness/Likes and Dislikes

The greatest strengths I had going into the project was my passion for this field which helped me stay persistent and motivated throughout the project, even during challenging times. Although I had some experience in Convolutional Neural Networks, the concepts of semantic segmentation and domain adaptation were completely new to me. After getting over the hump of understanding these concepts, I constantly ran into numerous errors such as incorrect output or code not executing at all due to environment and set up issues. This meant that I had to spend a lot of time debugging, re-reading documentation, searching online for additional learning sources, and several times completely deleting my code and starting over from scratch. Having to constantly overcome roadblocks rather than being able to steady progress to the next steps frustrated me and made me feel that I was behind schedule. However, I made it a point that I would not give up and forced myself to remain persistent even when things were discouraging at that moment. But that moment where an error is fixed and the code worked as expected was extremely rewarding and the sense of accomplishment is definitely an aspect of this project that I liked a lot.

One major issue I had from the beginning of the project was with the environment setup. I had used Google Cloud Platform for my past research projects, and so I thought I would set up another VM instance on GCP to use Google's GPUs and have a place to store the ML model files. Setting up this instance was a complicated, multi step process.First, I had to first instantiate a new VM on GCP (selecting the zone, GPU, Boot Disk framework, and other specifications). Next, I had to install Google Cloud SDK, which itself was another complicated process. Once I set up the GCloud SDK, I had to run the instance on a localhost port and after doing that, I could run a jupyter notebook on this instance. However, because I was using Google Colab, there was an extra step of connecting this instance to my Colab notebook. When I was first using this method, I didn't run into any issues and I was able to use the GCP GPU without error. However, something happened while I ran the instance (this must have been when I was updating some packages) where suddenly, the notebook started crashing everytime I tried to connect it to the port. This ended up taking up a considerable amount of time to fix and I was stuck. However, I realized that there was an option in Colab to simply choose a GPU for the notebook and that all along, I just had to go to the Notebook settings and select this option. Thus, I could have saved a lot more time trying to look for alternatives rather than try to fix the errors non stop.

A second similar issue I had was with using EchoNet's utilities to work with the EchoNet data. EchoNet has a github repo where it contains utilities to use their datasets, methods, and so forth which has code behind the scenes to work with the EchoNet data, meaning that I didn't have to do much in this case (i.e. reading the CSV files, gathering and storing all the video data into datasets, etc.). However, when I pulled the code from Stanford's EchoNet GitHub, I was running into issues with the configuration file, which wasn't able to locate the folder where all the EchoNet videos were. I tried everything I could (changing the file path, moving different files into different locations, etc.) to fix it, but it didn't work. Then, I found out that I could pull the EchoNet from my mentor's GitHub, which was modified and changed a bit from the original EchoNet Github. This did the trick and again, rather than spending so much time trying to fix my problem, I could have saved a lot more time by looking for an alternative solution.

One minor challenge was communication given that this was a fully remote project. Additionally my mentor and I were in different time zones which limited the time overlap. But for the most part my mentor and I were flexible with our schedules, meeting outside of our scheduled weekly meetings at times and having our weekly meetings go well beyond the planned 1 hour.

Where this research is applicable

Semantic segmentation is applicable in a variety of computer vision tasks - medical image analysis, autonomous vehicles, agriculture, drones, and facial recognition.

What I would do similarly/different

What I would do similarly is continue to heavily use the documentation and tutorials provided with Monai and other online resources such as programming forums and YouTube tutorials. Firstly, they gave me a thorough understanding of where to start a new challenge. For example, the Monai tutorials helped me with understanding how to start coding my neural network. Secondly, during the implementation phase, the online resources were instrumental in helping me solve some tough errors I faced and unblock myself.

What I would do differently is to not be very hesitant to ask questions. There were times where I was insistent on solving an issue by myself, when in hindsight I could have saved time if I had just asked for help from my mentor after a few attempts at solving it myself. I also would be more organized with my tasks. There were a few moments where I was overwhelmed with what I should work next due to lack of visibility. Using project management tools such as Trello to create a list of tasks, identify dependencies and priorities would have helped me stay more focused and be more productive.

Advice/What I wish I know at the outset of the summer

My advice to any person interested in such a research project is to plan well, stay organized, be resourceful, self-reliant, but not hesitate to seek help from your mentor when you are blocked, and most importantly, have fun with the process. If you don’t enjoy what you are doing, it becomes very hard to stay motivated and bring the project to successful completion.