Generating Fake Ultrasounds to Improve Deeplab V3 Performance

Amir Zaman

**Introduction:**

In this project we aim to improve the performance of image segmentation of ultrasound tumors on the deeplab v3 model by using fake ultrasound images. These fake images will be generated so there are seemingly infinite unlabeled images to help training. We rely on two data sets for labeled training. The BUSI breast tumor ultrasound set includes 437 benign and 210 malignant images with corresponding masks. The USG breast tumor ultrasound dataset contains a total of 256 images with corresponding masks. The sets are not combined and are trained separately to compare metrics. All code is done using python and various machine learning libraries including pytorch and sci-kit-learn. The initial model training was performed on a GE76 Raider laptop, but was soon migrated to the Kean GPU cluster to run reliable operations all day.

**Methods:**

We start by performing a supervised learning task by training deeplab v3 on image segmentation using the two datasets separately. Then, to improve performance, we will attempt to use the mean teacher method which combines the use of labeled and unlabeled images to train the segmentation model. The source of these unlabeled images will be from a separate diffusion model trained to create ultrasound tumor image look-alikes. This gives us an infinite reservoir of unlabeled images. For the labeled images the training and test sets are defined in corresponding excel files for reproducibility.

The code used to make the diffusion model heavily references this github project:

<https://github.com/mueller-franzes/medfusion/tree/main>

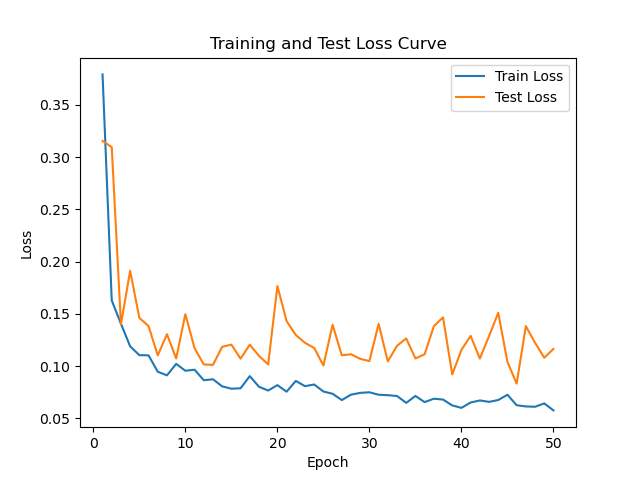
**Fully Supervised Training:**

First we perform fully supervised learning using the labeled images from the USG set and BUSI set separately. To optimize we try artificially inflating the datasets by adding randomly flipped and rotated versions of each of the images essentially doubling the training data. Additionally we train the model several times using different hyperparameters, the most optimal found was learning rate for the optimizer fo 1\*10-4 with batch size of 8 over 50 epochs. Since the training tapers off quickly and just bounces without improvement of the loss we train the BUSI for 30 epochs.

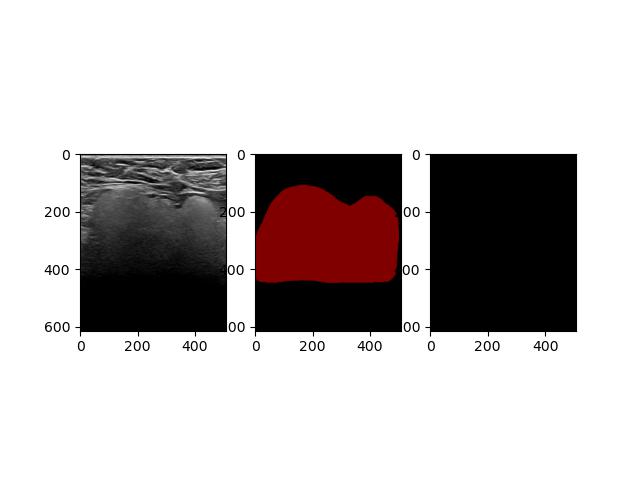
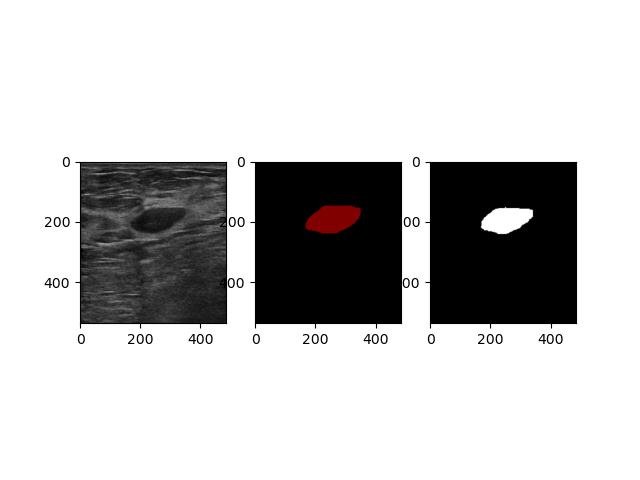
The initial metrics can be found in Table 1 and visual comparisons in figures 1-4. Performance is quite poor across the board however training on the BUSI set is slightly better likely due to being a somewhat larger dataset.

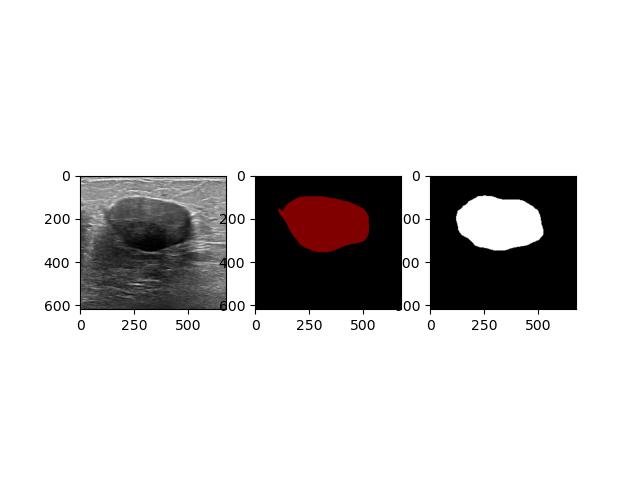
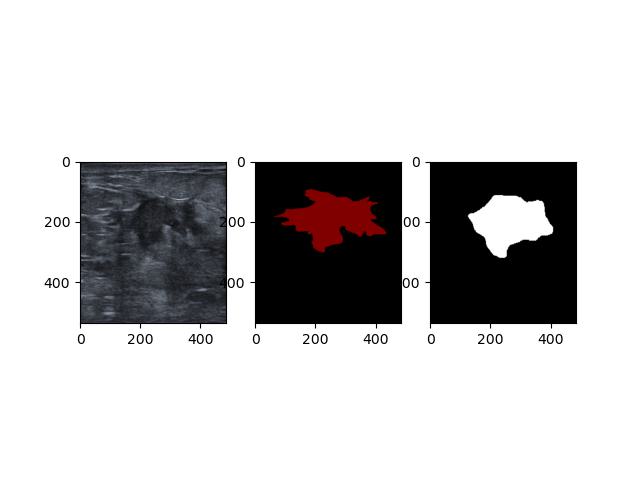
| Dataset | USG | BUSI |
| --- | --- | --- |
| Iou | 0.6125 | 0.6974 |
| F1-Score | 0.753 | 0.7806 |

*Table 1: Comparison of the fully supervised image segmentation results of training on the two sets separately. Iou sees how exactly the mixels of the input mask match the pixels of the output masks closer to 1 is better. F1-Score combines precision and recall to estimate the efficiency of th prediction, closer to 1 is better*

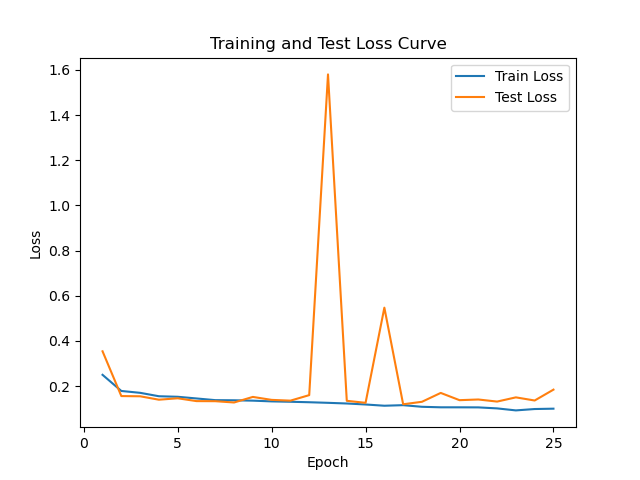


*Fig 1: Test and Training Loss curve of training deeplab v3 on the USG set*

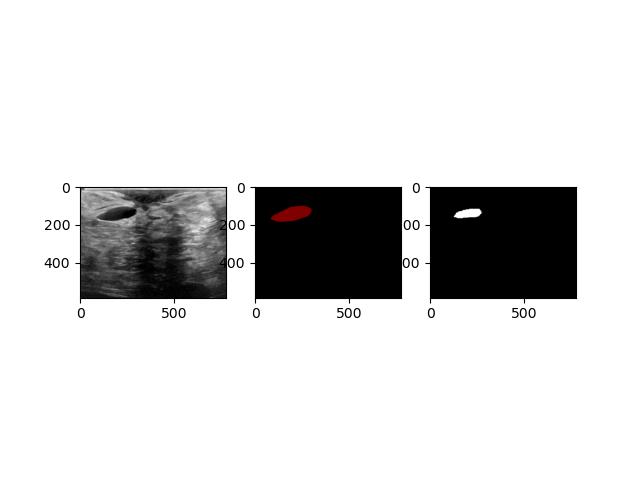


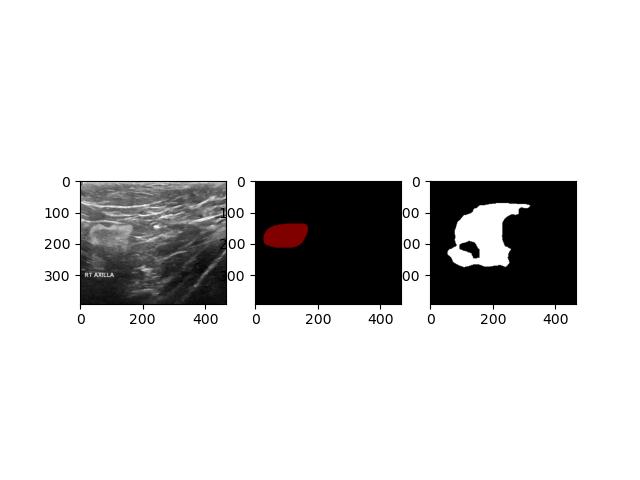
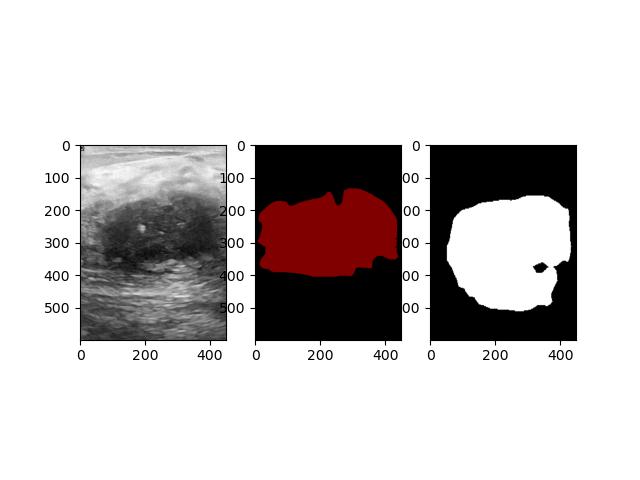


*Fig 2: Four samples of actual (center) vs predicted (right) masks from the USG set.*



*Fig 3: Test and Training Loss curve of training deeplab v3 on the BUSI set*





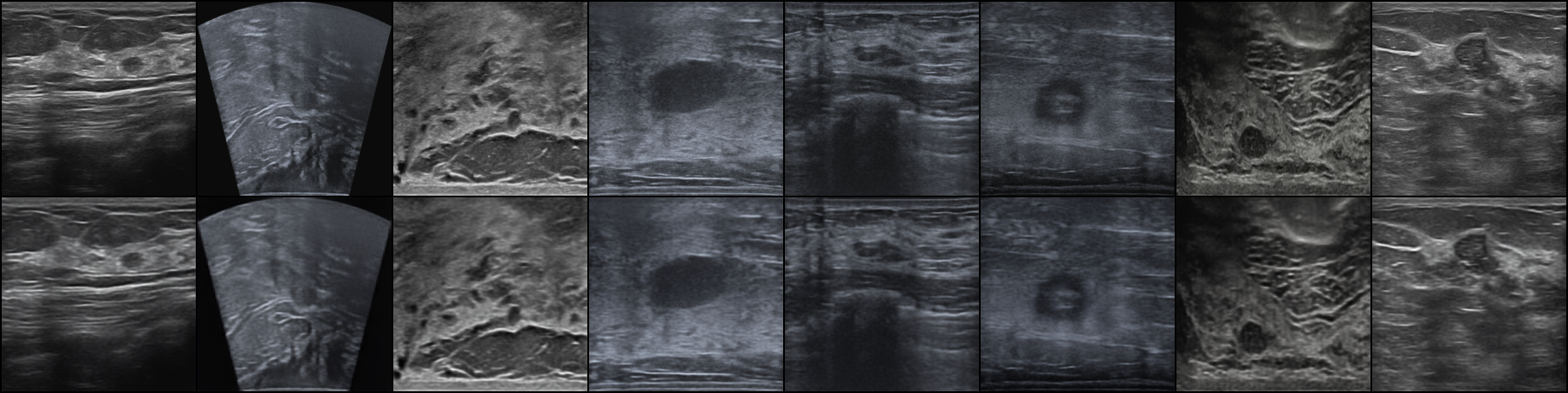
*Fig 4: Four samples of actual (center) vs predicted (right) masks from the BUSI set.*

**Embedder and Diffusion:**

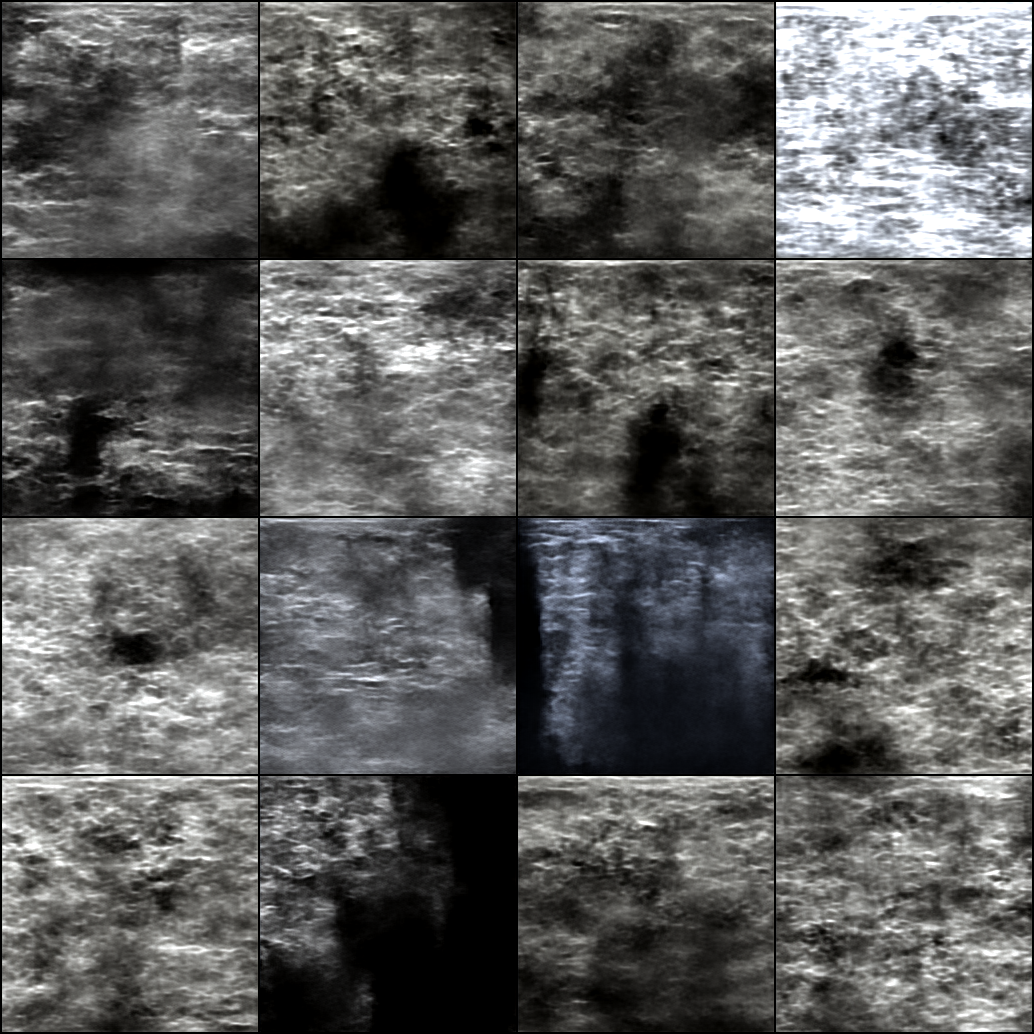
Next we use the “medfusion” project on github to train a latent embedder and then use the resulting embedder to train a diffusion model. That diffusion model then is used to generate 1000 synthetic ultrasound images to be used in semi-supervised training. We train the embedder separately on the USG and BUSI set. The BUSI results in better looking images and better metrics so we use the BUSI diffusion model to generate our 1000 synthetic images. However, the metrics are not that good for both the embedder and diffusion model as seen in Table 2. Most of the time was spent making the model and pipeline work. There is a lot of room for hyperparameter tuning the embedders and diffusion models. This will take some time though as training these are time consuming.

| Dataset | USG | BUSI |
| --- | --- | --- |
| LPIPS | 0.8364 | 0.8617 |
| MS-SSIM | 0.7643 | 0.8771 |
| MSE | 0.0034 | 0.0026 |

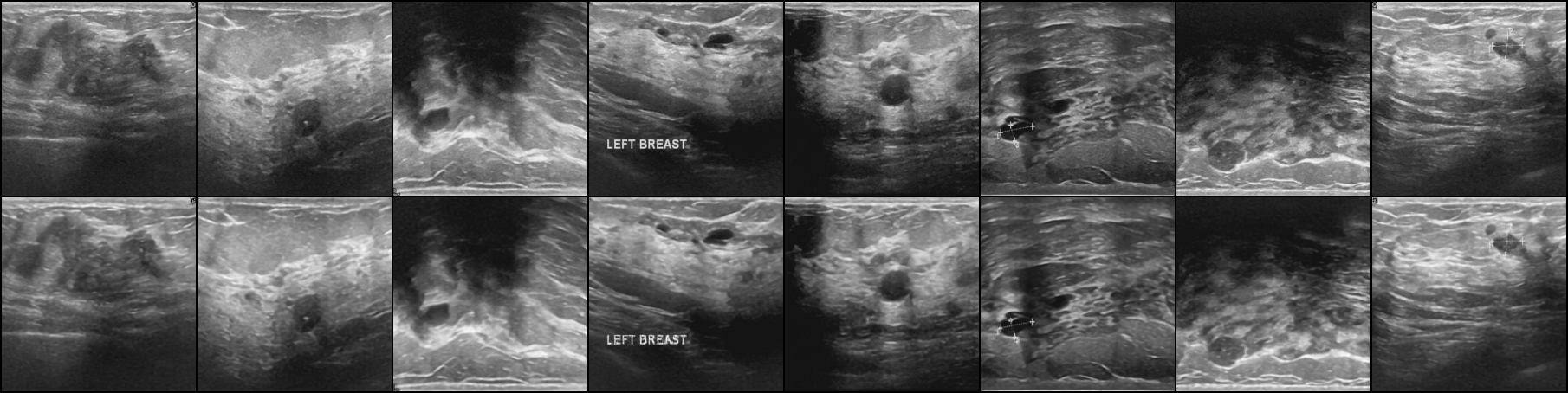
*Table 2: Embedder metrics for the two datasets. LPIPS and MS-SSIM closer to 1 means output is visually similar to input, MSE is the exact numerical difference between the input and outputs pixels. The point of an embedder is to be able to recreate the input images exactly while reducing the input images’ information or “space”.*



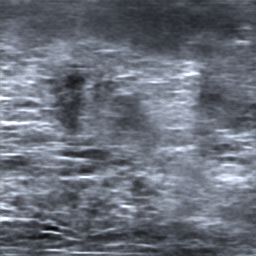
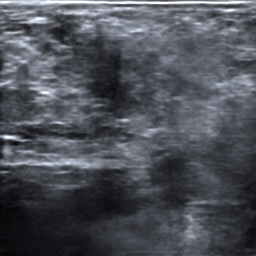
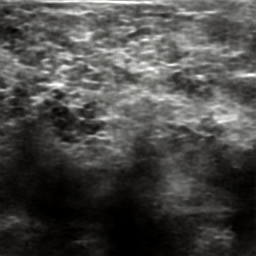
*Fig 5: Actual (top) vs embedder generated (bottom) of training VAE embedder on USG set*



*Fig 6: Samples of the diffusion generated images after training diffusion on the USG VAE*



*Fig 7: Actual (top) vs embedder generated (bottom) of training VAE embedder on BUSI set*



*Fig 8: Samples of the diffusion generated images after training diffusion on the BUSI VAE*

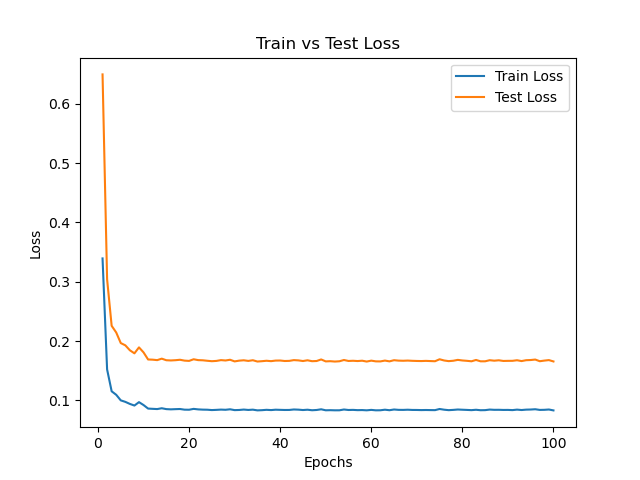
**Semi-Supervised Training:**

Lastly we use the labeled images and 1000 fake images to retrain the deeplab v3 from the beginning of this project and see if there is a performance difference. We use the unlabeled images ina semi-supervised training method called “mean-teacher”. The mean-teacher method uses pseudo labels on unlabeled data to encourage smoother outputs on masks of the labeled images.

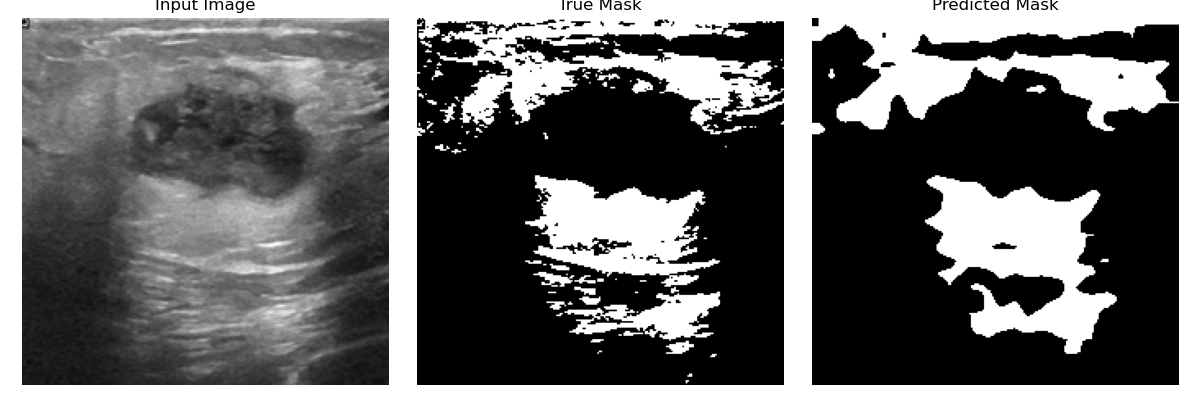
There was no hyperparameter tuning, but preliminary evaluation metrics show little performance increase, likely a non-significant increase from fully supervised learning. The metrics can be seen in table 3 below.

| Dataset | USG | BUSI |
| --- | --- | --- |
| Iou | 0.6424 | 0.6900 |
| F1-Score | 0.7752 | 0.8041 |

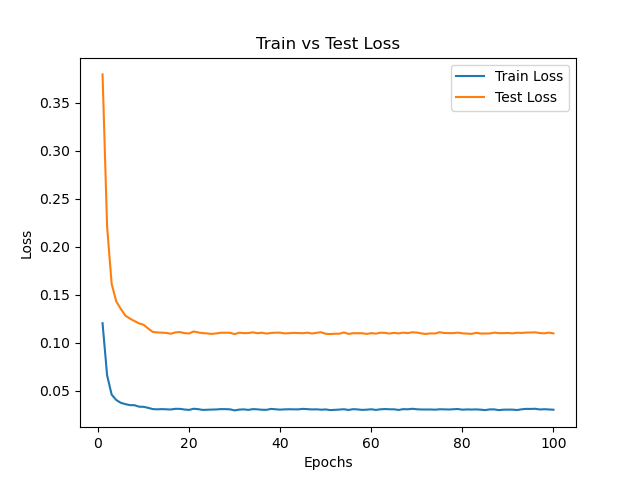
*Table 3: Metrics for evaluating the performance of semi-supervised training deeplab v3 model on the two datasets.*



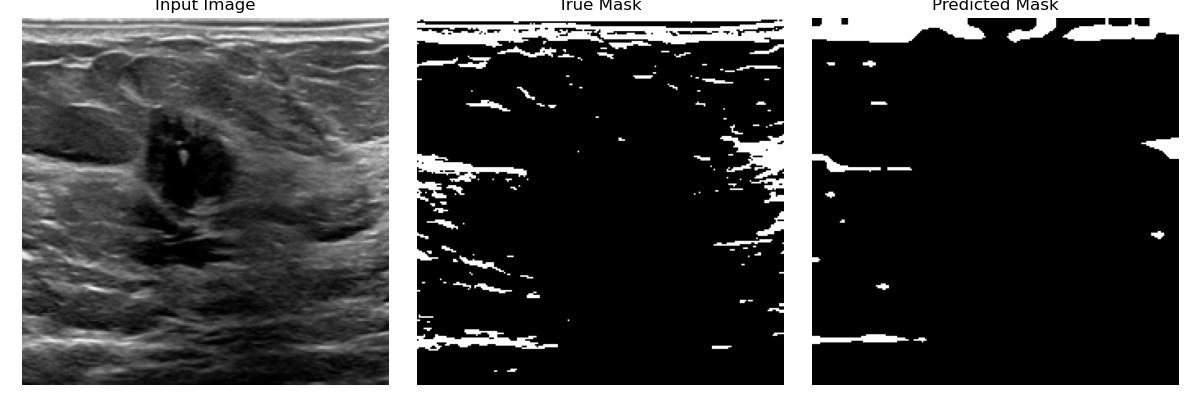
*Fig 9: Test and Loss curve of training deeplab v3 using the mean-teacher method on the USG set and the diffused fakes.*



*Fig 10: Failed comparison of the actual (center) vs generated (right) masks for the USG mean teacher model.*



*Fig 11: Test and Loss curve of training deeplab v3 using the mean-teacher method on the BUSI set and the diffused fakes.*



*Fig 12: Failed comparison of the actual (center) vs generated (right) masks for the BUSI mean teacher model.*

**Conclusion:**

The performance from the supervised and semi-supervised training was too similar to regard any sort of improvement or detriment to training the deeplab v3 model on image segmentation breast ultrasound tumor. However, there are many places along the pipeline that have room for optimization. For example: we tried removing all the obscure blurry ultrasounds from the USG set to train the embedder and performance did not increase significantly. Hyperparameters can be tuned finer for the embedder, diffusion, and the semi supervised method. Other semi-supervised methods can also be tried such as co-training.

In summary this project was more of an excellent exercise in pipelining deep learning technologies and learning the use of infrastructure surrounding it.