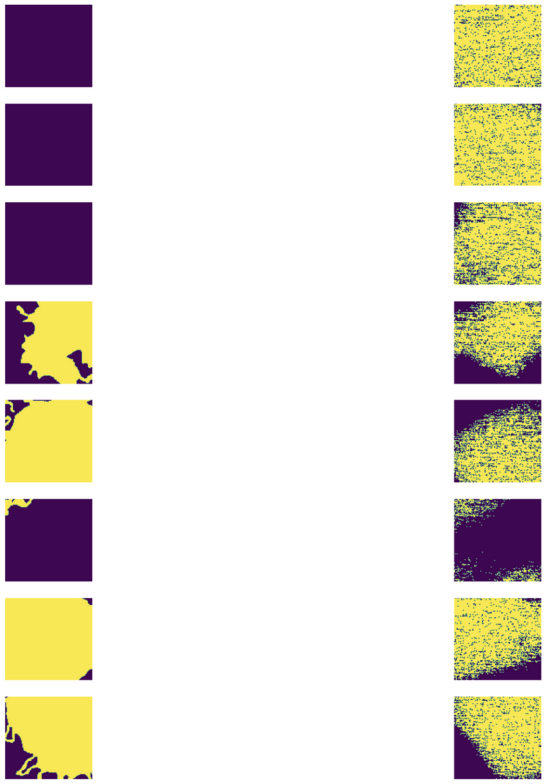
Anurag Parcha - 301558925

CMPT 732: Practices in Visual Computing I

Assignment-2: UNet

1. Report the resulting segmentation of the test images



2. What was the size of the images you used for training? you can rescale your images for faster training and better memory usage. For the final results, use at least 128 × 128 images.

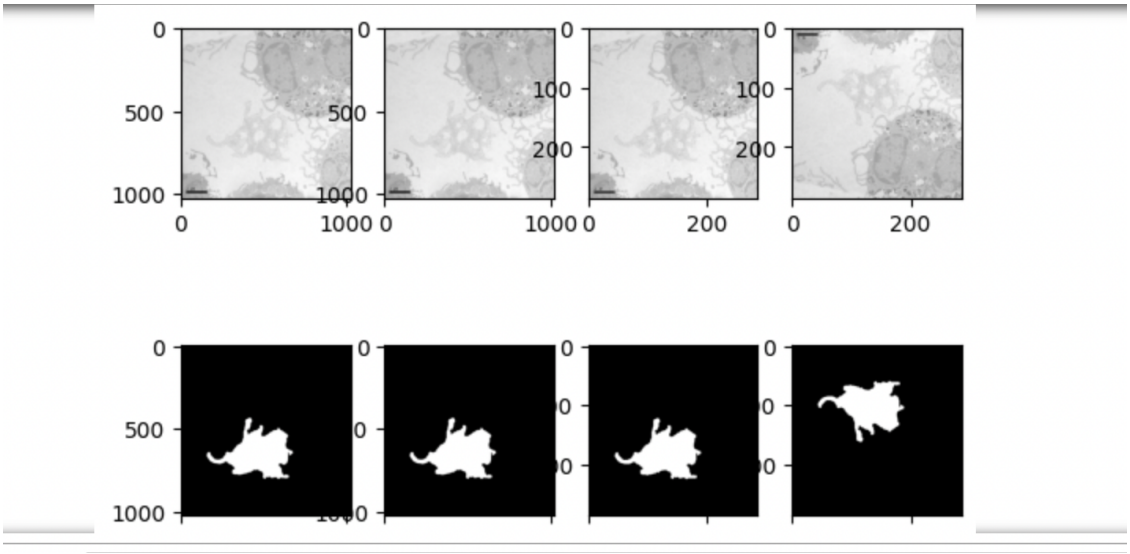
   H x W = 286 x 286

3. Which data augmentation strategies did you use?

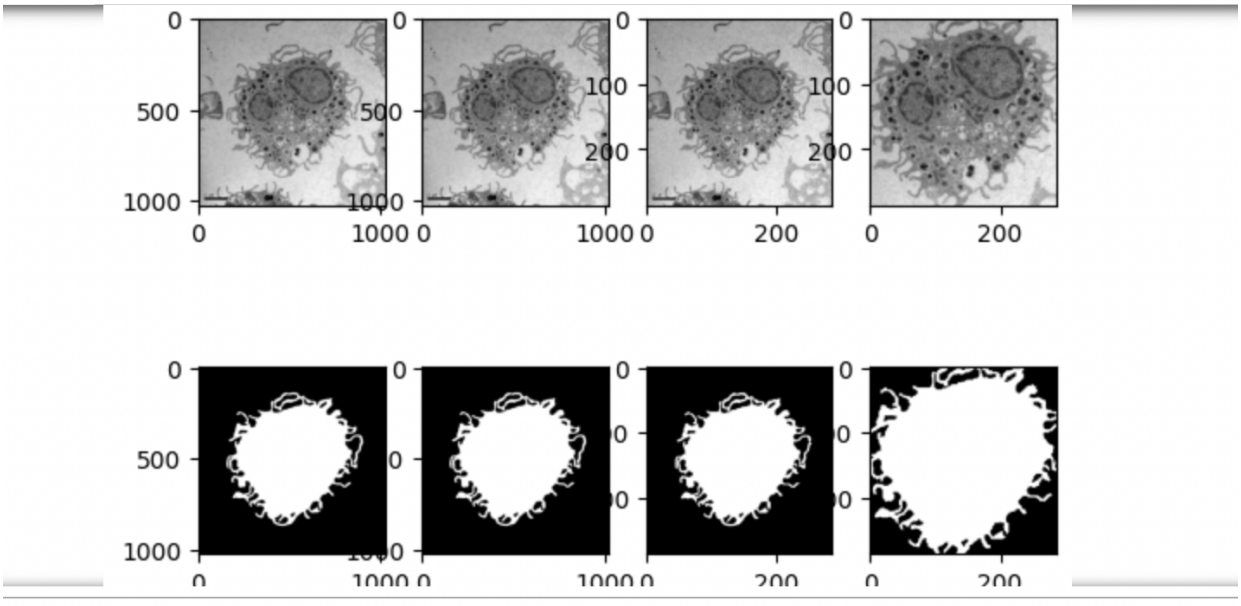
   The data augmentation was done dynamically in each training epoch. The data augmentation was only applied to the training dataset and not the

validation or test data set. Image and Labels were both augmented. The data augmentation techniques used were:

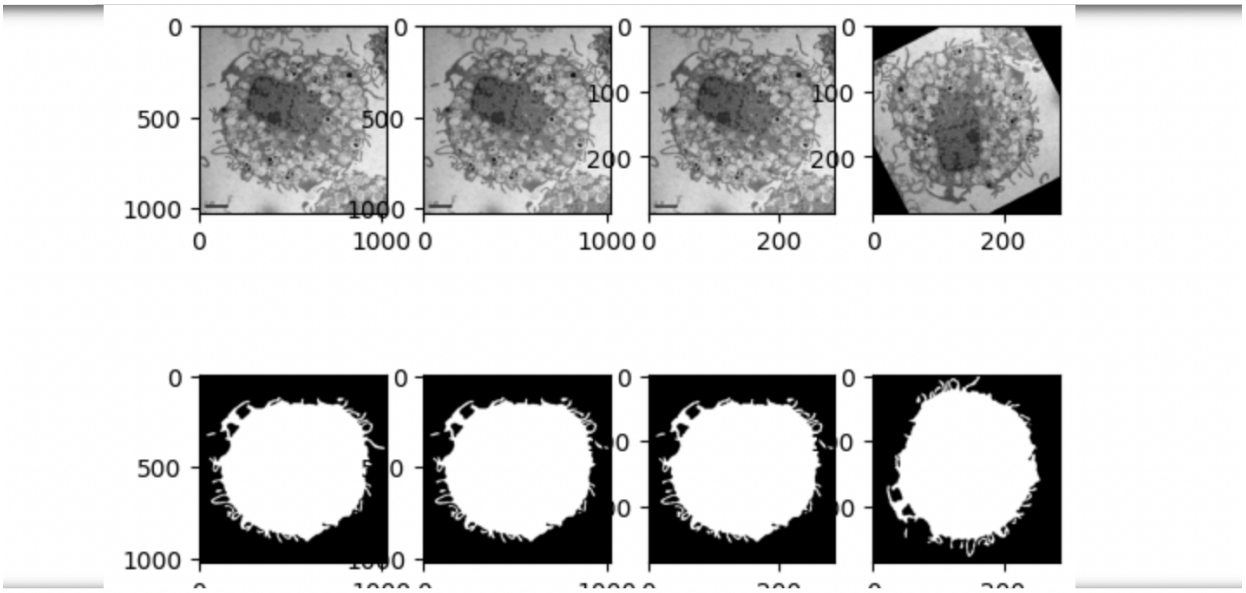
a. Horizontal / Vertical Flip



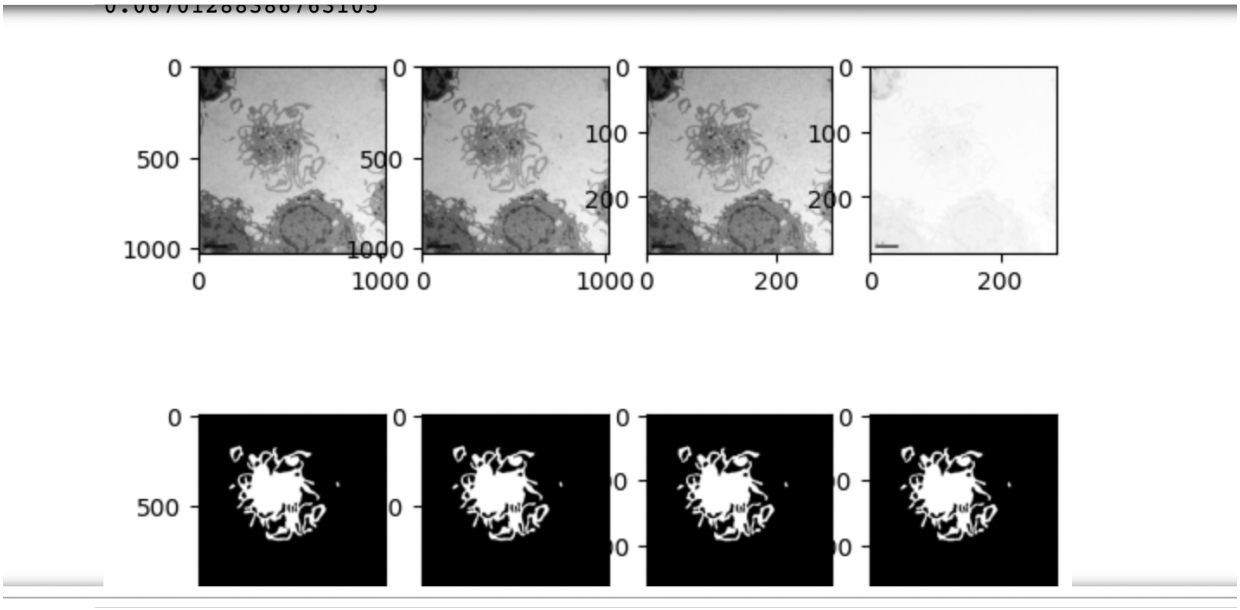
b. Random Center Zooming / Crop and Resize



c. Random Rotation between 0-360 degrees



d. Gamma Adjustment



4. Did you deviate from the original architecture? If so mention the changes.

I have implemented the UNet as per the paper and have not changed its architecture. For data augmentation i am augmenting images and labels per epoch instead of doing it before the training. I have read that this is

better for most cases. I have also only augmented the training dataset and not the validation/ test set.

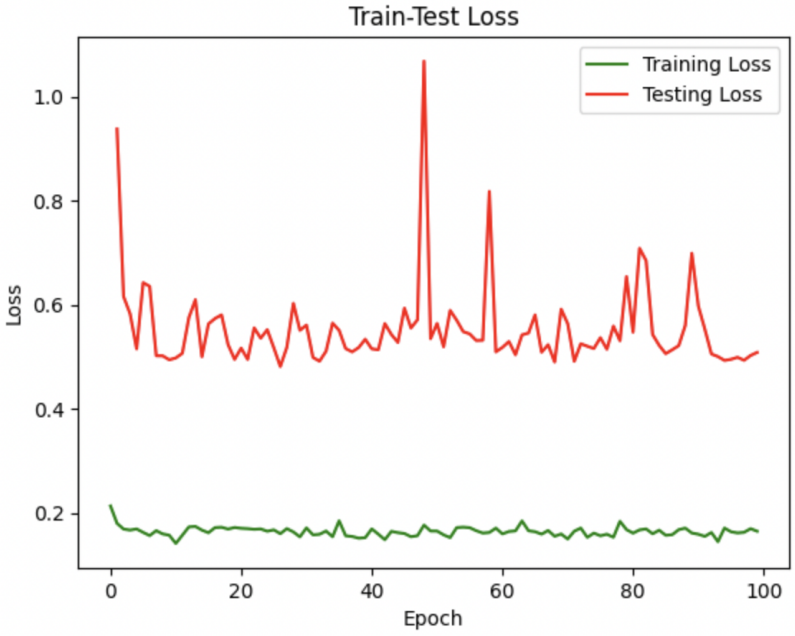
5. How many epochs did you train? provide plots for the train-test losses.

   I have trained mostly for 100 epochs. But I have tried smaller epochs such as 20 and 25. The model learns slowly overtime which I suspect is because of the learning rate and weight decay parameters that I have chosen.

   Epochs are calculated based on the dataset size and variety firstly, since the size of the image dataset is small, we have to run more epochs to compensate for the lack of training data. This is so that the model learns the structure of the data.

   Secondly, learning rate and weight decay also have a role in the decision of number of epochs for training. If the learning rate is too low, the model will converge to a good accuracy very slowly which means that we need more epochs. However is the learning rate is too high, the model will quickly overshoot the minima and the loss will keep bouncing around. Weight decay helps to regularize the generated weights so that the bouncing is reduced and helps generalize the weights to the training dataset.

   Train-Test Loss – learning rate = 1e-2 – epochs = 100



Train-Test Loss - learning rate = 1e-4 - Epochs = 20

Therefore 1e-4 is a better learning rate than 1e-2.

6. What batch size and learning rate did you use?

   Batch size = 4

   Learning rates = 1e-2, 1e-3, 3e-4, 1e-4

7. How long did the training take?

   For 100 epochs with lr =1e-4 -> 30 mins

   For 20-25 epochs with lr=1e-4 -> 5 - 8 mins

8. Bonus - Plotting hyperparameters, weights and biases using wandb

Inference:

Best Accuracy reached with:

learning rate = 1e-4

epochs = 100

batch size = 4

weight decay = 0.0005

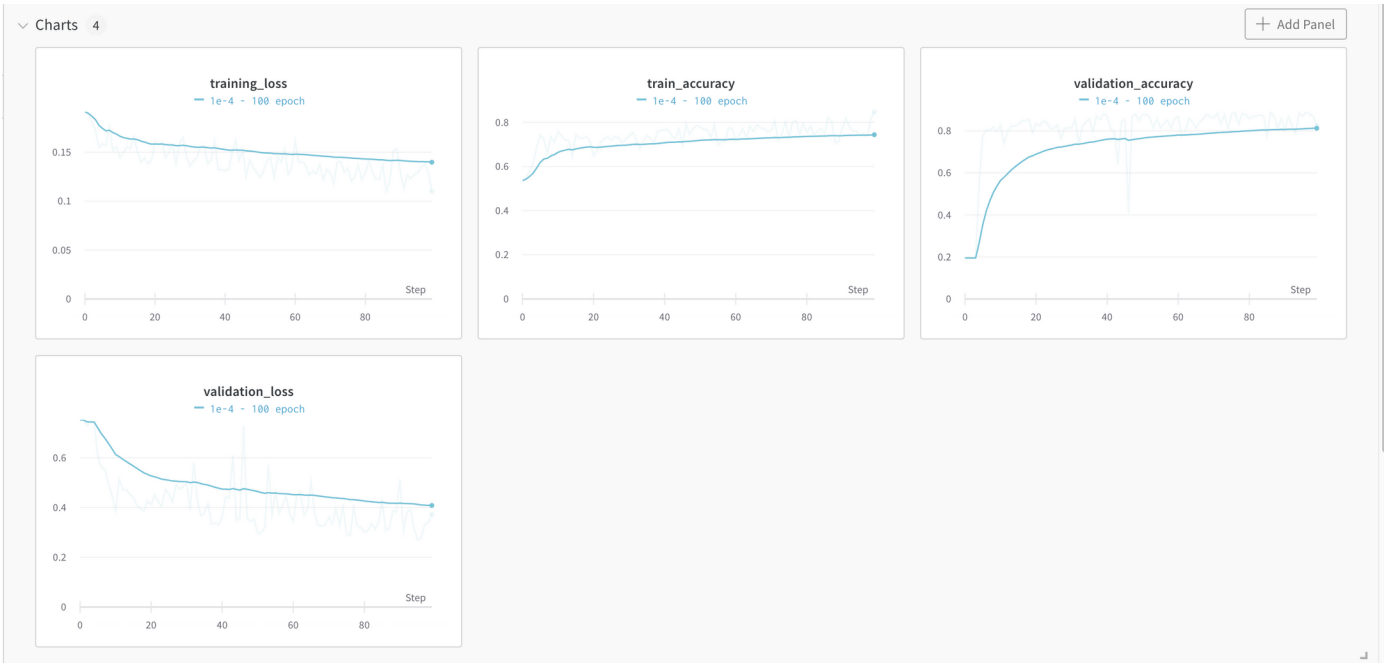
Validation Accuracy = 0.8475

Anurag Parcha - 301558925

Tracking from wandb

Plot of above scenario

After applying smoothing to each chart over 100 epochs, the below is gathered.



Running the same for 20 Epochs instead gives the below plot.

Waiting for W&B process to finish... **(success).**

Run history:

train_accuracy	▁▃▅▆▇▇▇▇▇▇▇▇▇▇▇▇▇▇▇▇
training_loss	▇▆▅▄▃▂▂▂▂▂▂▂▂▂▂▂▂▂▂▁
validation_accuracy	▁▃▅▆▇▇▇▇▇▇▇▇▇▇▇▇▇▇▇▇
validation_loss	▁▂▂▂▂▂▂▂▂▂▂▂▂▂▂▂▂▂▂▁

Run summary:

train_accuracy	0.76719
training_loss	0.15675
validation_accuracy	0.84634
validation_loss	0.44442

Synced **logical-snowflake-177**: https://wandb.ai/an-u-rag/unet-image-segmentation/runs/7fzuipi7
Synced 6 W&B file(s), 0 media file(s), 0 artifact file(s) and 0 other file(s)

Find logs at: `./wandb/run-20221102_152250-7fzuipi7/logs`