CMPT 742 Visual Computing Assignment 2 Report

Name: Jiaqing Hu (Vincent)

SFU ID: 301368526

SFU Mail: jiaqingh@sfu.ca

1. Segmentation Results

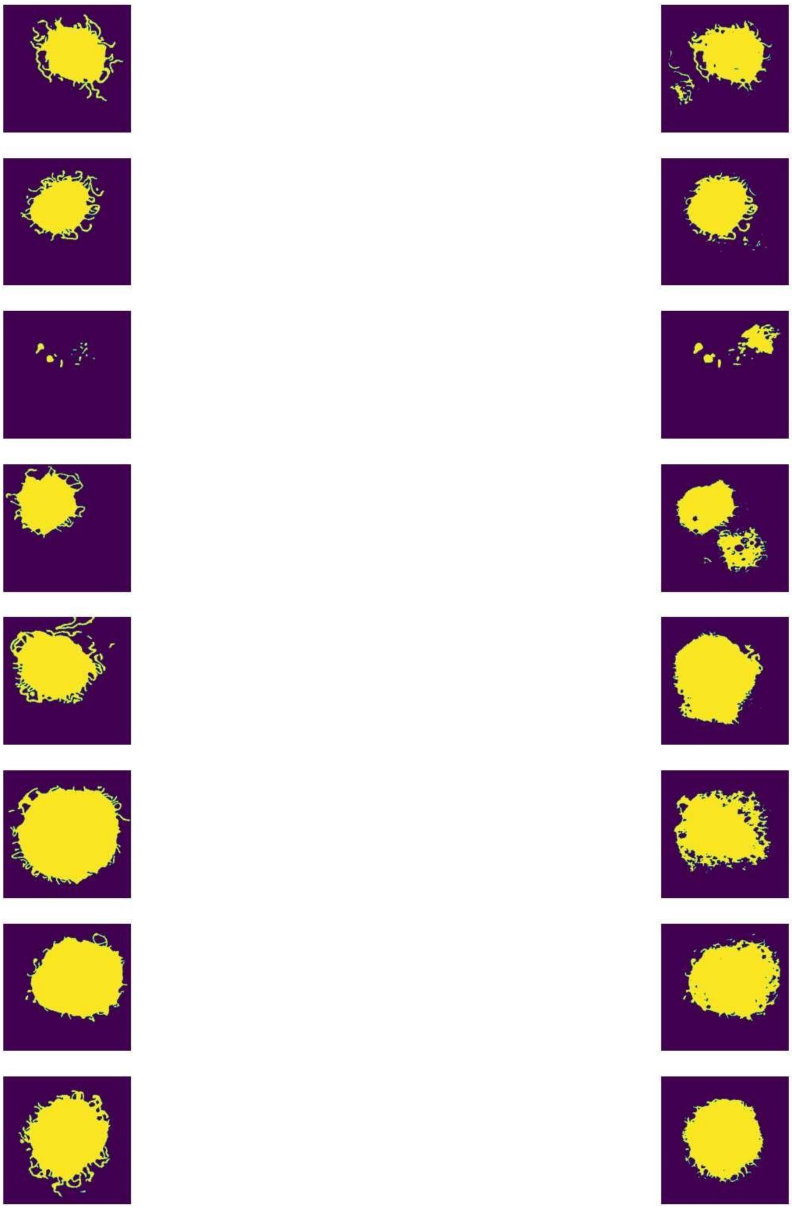

fig 1 result with image size 256

I used ImageSize 572, 256, 128 for training, and 256 image size for final results.

2. Data Augmentations

I implemented all the data augmentations mentioned in Part 5, namely:

a) Horizontal/Vertical flip, using vflip(), hflip() from pytorch

b) Zooming, using resized_crop() from pytorch

c) Rotation, using rotate() from pytorch

d) Apply Gamma correction, using adjust_gamma() from pytorch

e) Apply Elastic Transformation as mentioned in the original paper, using ElasticTransform() from pytorch

3. Unet Structure

```
class UNet(nn.Module):

    def forward(self, x):
        # implement the forward path
        x1, x1_maxpool = self.inc(x)
        x2, x2_maxpool = self.down1(x1_maxpool)
        x3, x3_maxpool = self.down2(x2_maxpool)
        x4, x4_maxpool = self.down3(x3_maxpool)

        x_bot = self.bot(x4_maxpool)

        x = self.up1(x_bot, x4)
        x = self.up2(x, x3)
        x = self.up3(x, x2)
        x = self.up4(x, x1)

        x_out = self.outc(x)

        return F.softmax(x_out, dim= 1)
```

```
class twoConvBlock(nn.Module):
    """Part 1  The Convolutional blocks"""

    # initialize the block
    def __init__(self, input_channel, output_channel):
        super(twoConvBlock, self).__init__()
        self.doubleConvBlock = nn.Sequential(
            nn.Conv2d(input_channel, output_channel, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),  # ReLU
            nn.Conv2d(output_channel, output_channel, kernel_size=3, padding=1),
            nn.BatchNorm2d(output_channel),  # Batch normalization layer
            nn.ReLU(inplace=True),
            nn.Dropout(0.25)
        )
```

I modified the final ouput using a **softmax activation function** as it is a classification problem, sigmoid could be applied as well since there are two classes. By applying the activation function, it helps my model converge faster while becoming more robust. I also implemented a **dropout** to the end of convolution block for a more stable training loss and testing accuracy.

4. Training Parameters

Below is the summary of all parameters, Runtime is in seconds.

Name	Runtime	batch_size	epochs	image_size	learning_rate	accuracy	batch_loss	epoch_loss	test_loss
256_5e-5	30	4	20	256	0.00005	90%	0.1008	0.1175	0.1112
572_1e-7	152	4	20	572	1.00E-07	58%	0.1751	0.1877	0.1700
572	132	4	20	572	0.0001	87%	0.1167	0.1186	0.1161
128	15	4	20	128	0.0001	86%	0.1065	0.1238	0.1146
256	31	4	20	256	0.0001	92%	0.1350	0.1217	0.1010

5. Logging & Graphs

I used wandb for logging all the loss and parameters for each run.

Below is the graph generated by wandb.