CMPT 732: Practices in Visual Computing I

Assignment 2: Image Segmentation

By

Sidharth Singh

301475626

Simon Fraser University

Introduction

In the last few years, several neural networks for image segmentation have been designed with considerable success. One of the most prominent attempts was the U-Net by Ronneberger et al. (2015) (1). The name of the network stems from the symmetric shape of the architecture. As can be seen in figure 1, a contracting path is followed by a symmetric expansive path. In this assignment, you should implement the U-net architecture for cell image data segmentation using PyTorch.

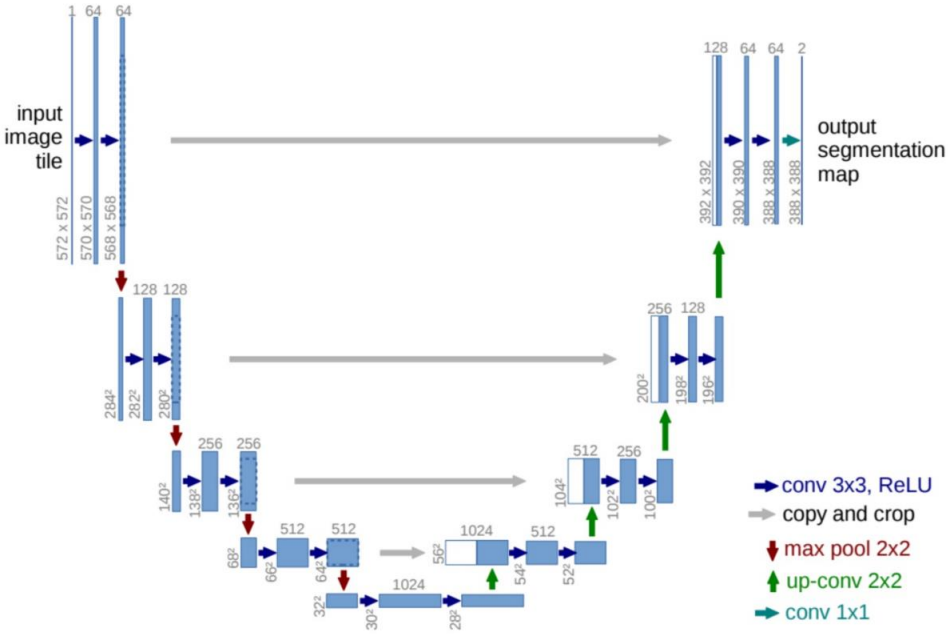

Figure 1: UNet Architecture

Network Architecture

There are two parts to a UNet: the encoder (aka the contractive path) and the decoder (aka the expansive path). The left part of Figure 1 is the contractive part, and the right part is the expansive path.

The contractive path has 2 3x3 convolution layers each with a ReLU. There is a Batch Normalization 2D layer in the implementation of the code. Finally, each such double convolution block is given to a max pool layer for stride 2 and kernel size 2.

Before the layer switches from encoder to decoder there is a bottleneck where the encoder and decoder meet. At this layer we have 1024 features as given by the research paper. Then we start to deconvolve and upscale the layer.

The decoder or the expansive path is a 2x2 Deconvolution layer that upscales the image back. To compensate for the missing feature details and better reconstruction, features from parallel downward

path are concatenated after cropping to the size of the decoder layer. Thus, this layer has double the feature depth of encoding path.

Finally, the output layer has a 1x1 convolution that outputs 2 classes. We are interested in the second class that generated the masks.

Data

A total of 38 images for cell segmentation with their corresponding labels are provided as “BMP” lossless images. They are of 1024x1024 size. For training they were resized to 572x572.

All the images in data are grayscale with 8-bit depth. The labels are just images with 0 and 1 as pixels values for segmentation task. The images take the full 8-bit depth. They were normalized from 0-1 range for better training and also to prevent exploding gradients.

Data Augmentation

Since the quantity of data was quite less, data augmentation for both image and label was required to increase the generalization and improve the accuracy further. 4 non elastic transformations were used to generate data:

1. Vertical Flip
2. Horizontal Flip
3. Random Crop and Resize
4. Rotation

Training

The model works a quite different than general convolution image segmentation networks. It classifies each pixel to the class. To acquire missing boundary information due to convolutions, we need information from encoding path. Hence, we concatenate the features from encoder to the decoder thus doubling the feature depth and then applying convolution on them again.

For the experiment following hyperparameters were used:

1. Learning Rate: 1e-02
2. Optimizer: Adam
3. Weight Decay:
4. Train-Validation split: 0.8
5. Image size: 572x572
6. Training loss: 0.1212

Results

I was able to achieve an accuracy of about 74% across multiple runs. I believe further tuning can push the accuracy beyond 85%. Increase in dataset size and further experimentation with transforms can also improve it.

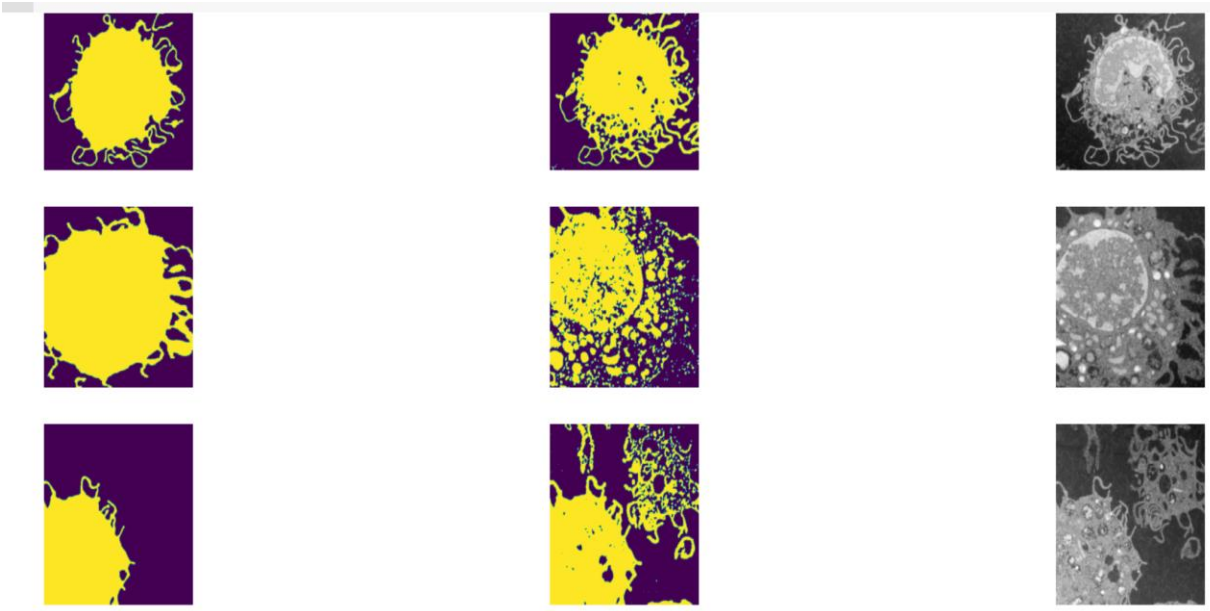

Figure 2 Label, Generated Mask, Image

Figure 3 Training Loss

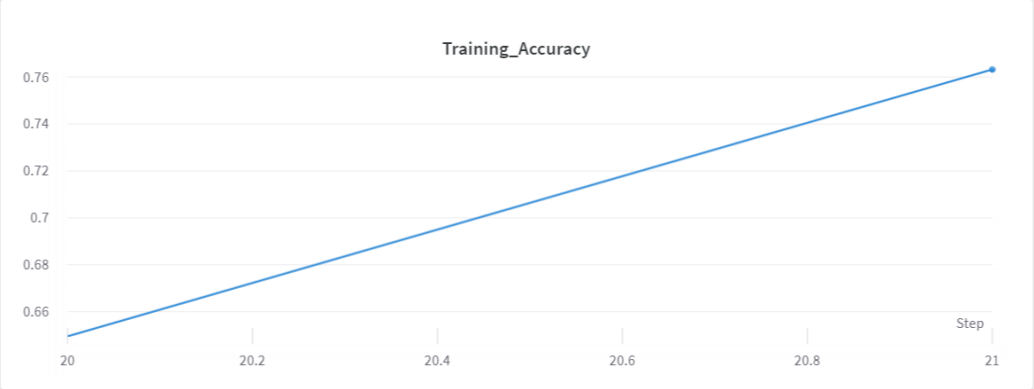

Figure 4 Training Accuracy

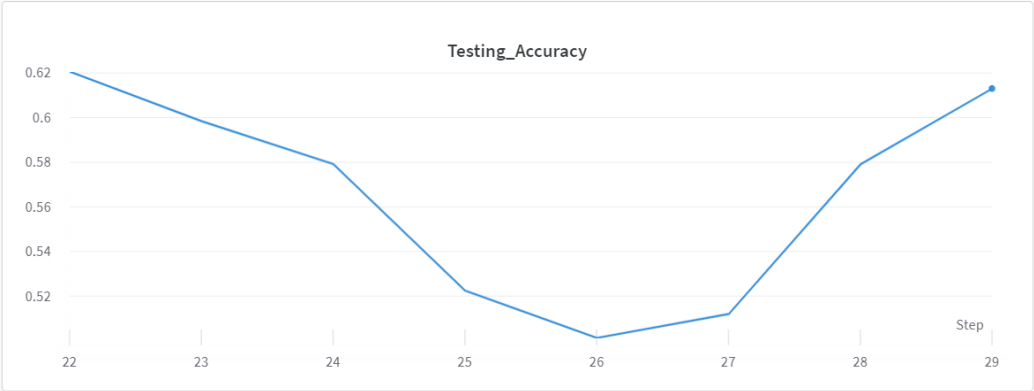

Figure 5 Testing Accuracy

The above metrics were generated and logged using wandb api on Google Colab.