Automated Segmentation of Mitochondria in Normal and Drug-treated Cells using U-net

Hsu-Ting Kuo*, Min-Zhi Gao**

Department of Biomedical Engineering, National Taiwan University
Correspondence : * B08508028@ntu.edu.tw, ** B08508020@ntu.edu.tw

Outline

1. Introduction
    a. Background
    b. Goal
    c. Proposed flowchart
2. Methods
    a. Data preprocessing
        i. Labeling
        ii. Augmentation
        iii. Resize
    b. Model
        i. Overview of the proposed convolutional neural network
        ii. Implementation - Pytorch package
        iii. Implementation - U-Net structure
        iv. Details in the model
    c. Automated Cropping
3. Results
    a. Model Performance
        i. Outcome Performance
        ii. Visualization
    b. Mitochondrial Segmentation
4. Discussion
    a. Problems encountered
    b. Future Work
5. Reference
6. Collaboration

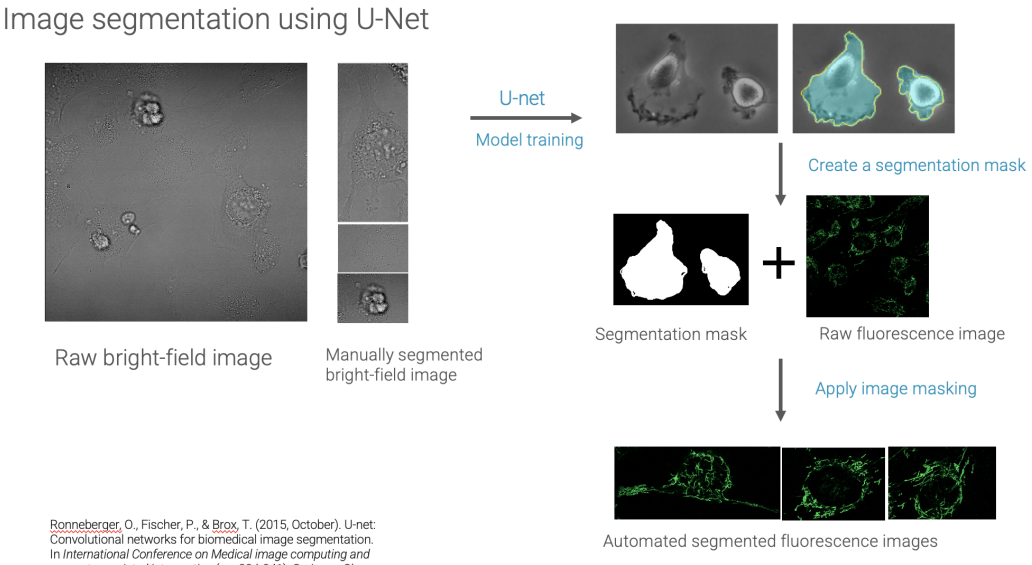
1. Introduction
   a. Background

      Mitochondria is the powerhouse of the cell and mitochondrial dysfunctions could lead to serious diseases. Thanks to the advances in optical imaging techniques, microscopy-based high-content screening for quantification of changes in drug-treated mitochondrial functions has become popular. However, the automated pipeline for mitochondrial image analysis is not yet completed as the existing segmentation methods cannot perform precise segmentation on drug-treated mitochondria.

   b. Goal

      Our goal is to train a U-Net model that can effectively predict the locations and regions of mitochondria from transmitted-light images and generate a binary mask based on the prediction. Using the output binary masks, we can perform mitochondrial segmentation on fluorescence images to acquire images of single mitochondria.

   c. Proposed Flow Chart

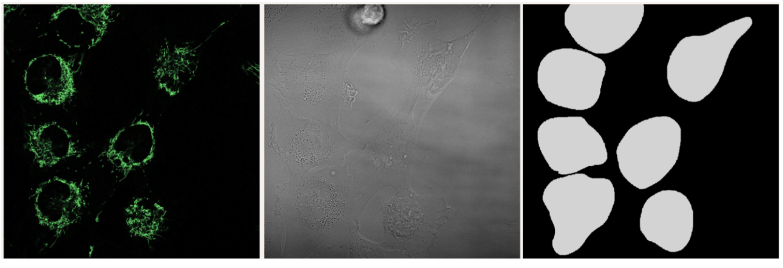


Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.

2. Methods
   a. Data Preprocessing
      i. Labeling:

         To generate ground truth images, we manually convert transmitted light images into binary masks using ImageJ.
         The steps are as follows:

         Compare fluorescence images and transmitted-light (TL) image to locate mitochondria → circle mitochondria in TL images→ ROI manager, add the selected regions → select all the ROI → clear outside → open Properties, change the color of ROI to white → Process, Make Binary with method Otsu → split channels →save the binary mask.

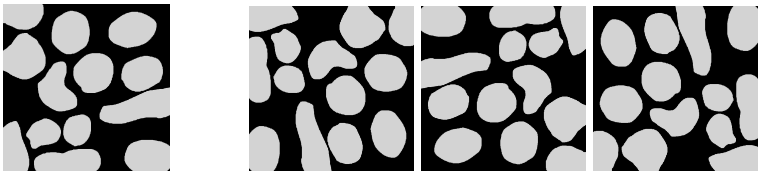


ii. Augmentation:

To solve the problem of insufficient amount of data, we use data augmentation to automatically create new data.

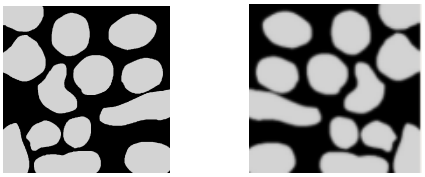
1. Random_Rotation:

We first do a random rotation to the original image (bright-field images), create random new data with rotation of 90 degrees, 180 degrees, and 270 degrees.



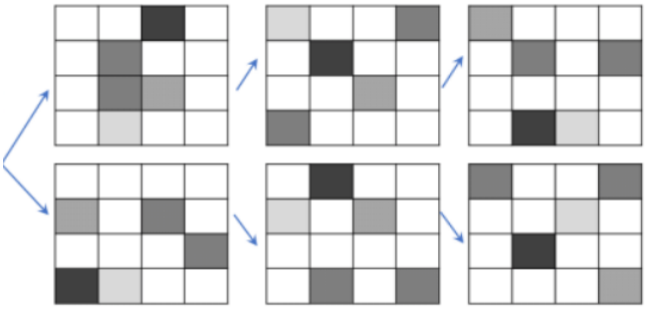
2. Random_Flip:

We also do the vertical flip to the original image.



3. Generated Patches:

To do more data augmentation, we change the kernel size and the stride, do three times unfold, and then change the dimension with PyTorch kit "permute" and "view"



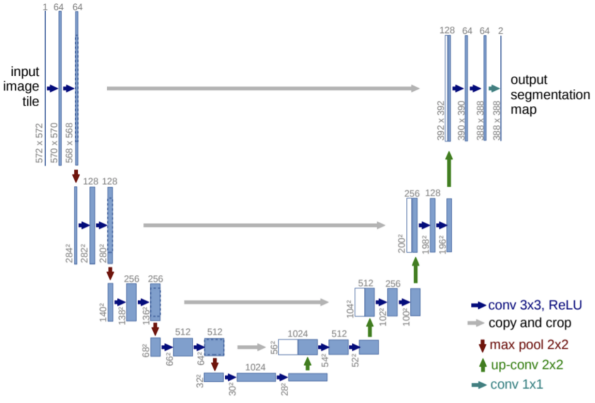
iii. Resize:

To speed up the training process and to meet U-Net's requirements for image sizes, we apply nd. image zoom function to resize input images (1987*1987 → 256*256).

b. Model

i. Overview of the proposed convolutional neural network U-Net:

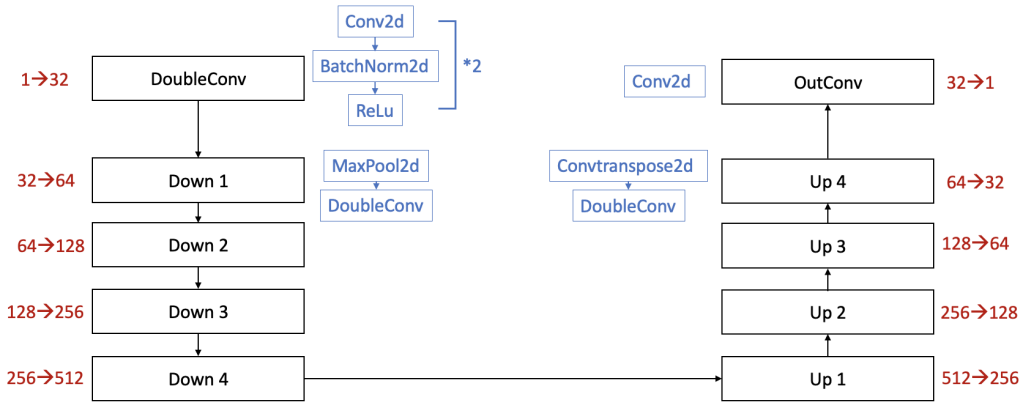
We chose U-Net due to its ability to yield relatively good results on pixel labeling tasks with limited dataset images.



ii. Implementation - Pytorch package:

```
import torch
import torch.nn as nn
import torch.nn.functional as F
```

iii. Implementation - U-Net structure:

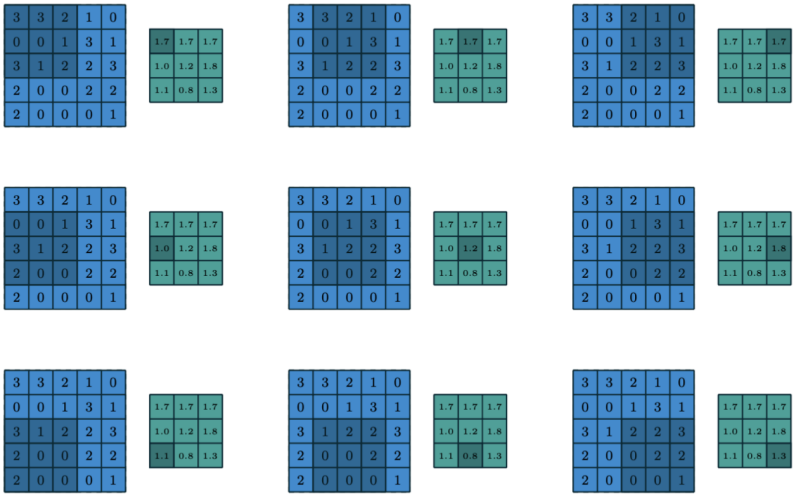



```
class UNet(nn.Module):
    def __init__(self, args, bilinear=False):
        super(UNet, self).__init__()
        self.n_channels = 1  #1
        self.n_classes = 1 #1
        self.bilinear = bilinear

        self.inc = DoubleConv(self.n_channels, 32)
        self.down1 = Down(32, 64)
        self.down2 = Down(64, 128)
        self.down3 = Down(128, 256)
        factor = 2 if bilinear else 1
        self.down4 = Down(256, 512 // factor)
        self.up1 = Up(512, 256 // factor, bilinear)
        self.up2 = Up(256, 128 // factor, bilinear)
        self.up3 = Up(128, 64 // factor, bilinear)
        self.up4 = Up(64, 32, bilinear)
        self.outc = OutConv(32, self.n_classes)

    def forward(self, x):
        x1 = self.inc(x)
        x2 = self.down1(x1)
        x3 = self.down2(x2)
        x4 = self.down3(x3)
        x5 = self.down4(x4)
        x = self.up1(x5, x4)
        x = self.up2(x, x3)
        x = self.up3(x, x2)
        x = self.up4(x, x1)
        logits = self.outc(x)
        return logits
```

iv. Details in the model:

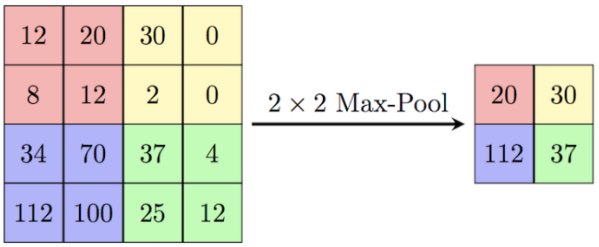
1. Convolution Method:

A convolution allows for weight sharing (reducing the number of effective parameters) and image translation (allowing for the same feature to be detected in different parts of the input space)



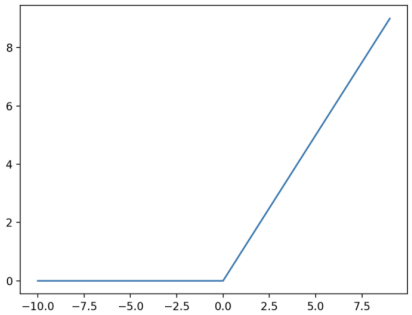
2. Max pooling:

A pooling operation that calculates the maximum value for patches of a feature map, and uses it to create a downsampled (pooled) feature map.



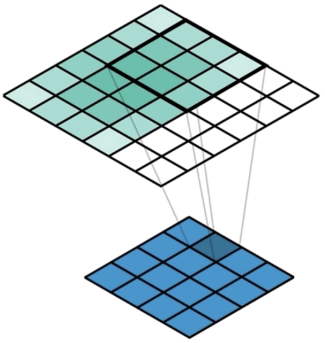
3. ReLu (Rectified Linear Units):

A type of activation function that is linear in the positive dimension, but zero in the negative dimension.



4. Convtranspose2d:

The transpose (**gradient**) of conv2d : expand grid-size (height and width) from previous layer.

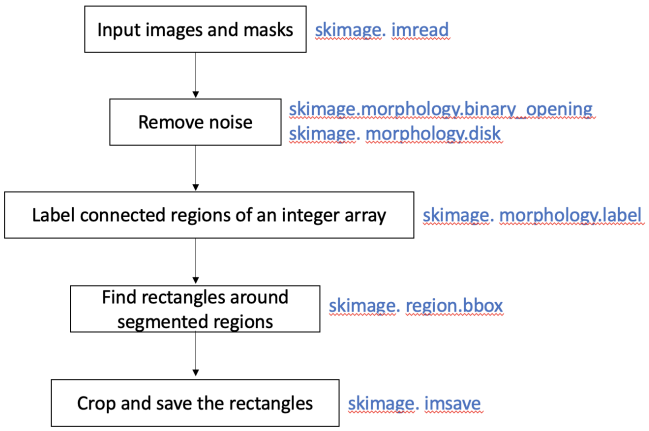


c. Automated Cropping

i. Implementation - Python libraries

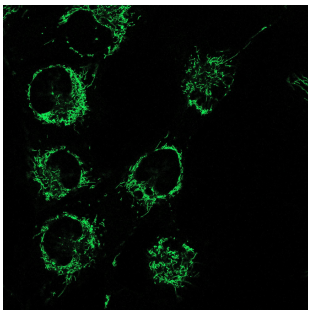
1. skimage

ii. Workflow

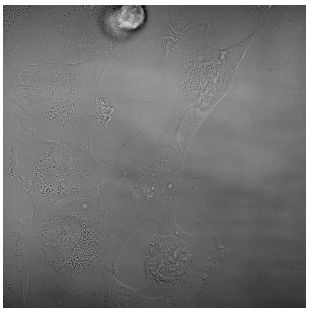


3. Results
    a. Model Performance
        i. Outcome Performance
            1. Initial Parameter
                a. Epoch: 100
                b. Batch size:
                    i. Test: 32
                    ii. Train: 12
            2. Outcome
                a. Train_mean_iou: 0.55162
                b. Test_mean_iou: 0.616704
        ii. Visualization
            1. Drug: Oligomycin / Dose: 1μM

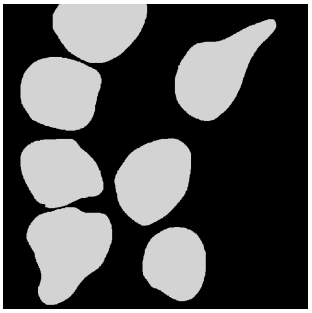
Fluorescent



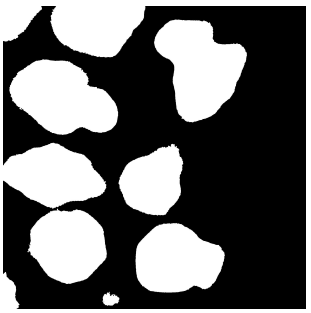
Bright Field



Manually Segment

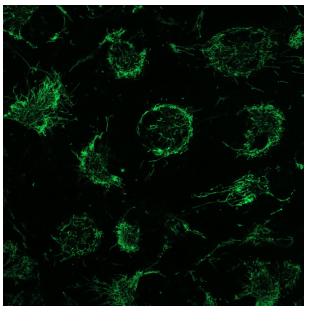


Automated Segment

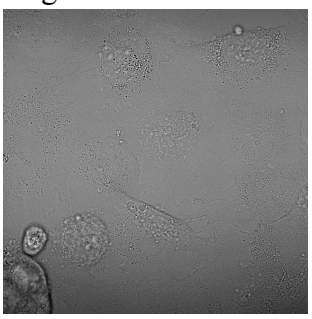


            2. Drug: FCCP / Dose: 0.5μM

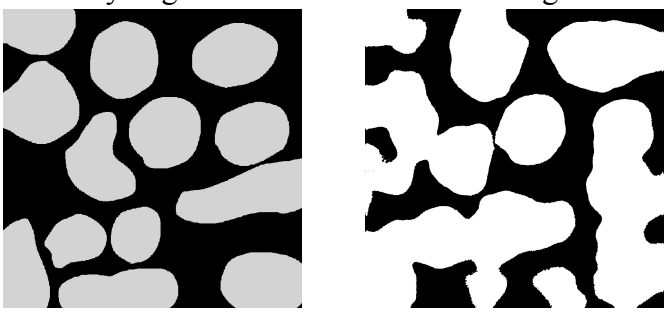
Fluorescent



Bright Field

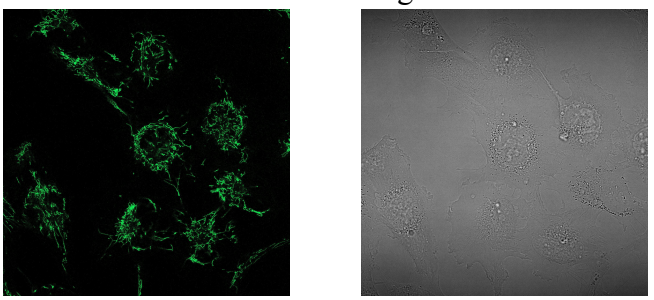


Manually Segment Automated Segment

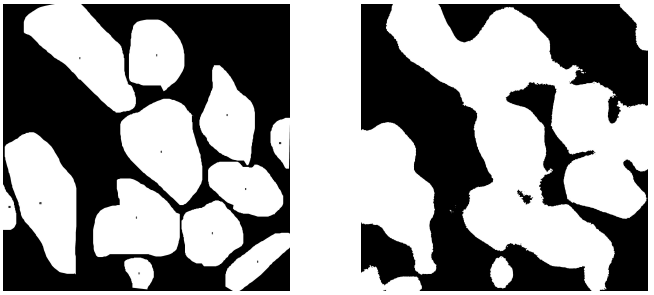


3. Drug: AntimycinA / Dose: 2μM

Fluorescent Bright Field



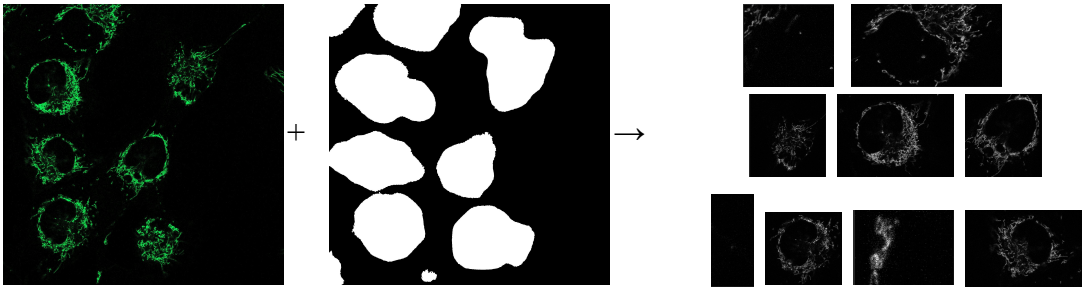
Manually Segment Automated Segment



4. Mitochondrial Segmentation

Drug: Oligomycin / Dose: 1μM

Original fluorescent image Output binary mask Image of single mitochondria



4. Discussion

a. Problems Encountered

i. Format

The first problem we met is to read in the image, changing from the original format of the open source (png.) into our image format (tif.)

ii. Small dataset

Due to the data deficiency, we use the three methods mentioned above to do data augmentation.

iii. Dimension

The original model provides a structure with the format (3, 256, 256), the 3 means RGB three-channel, 256x256 means length and width. However, the images we obtained are 256x256. Therefore, we lift the dimension up to (1,256,256) to make our images fit the model architecture.

iv. Loss function

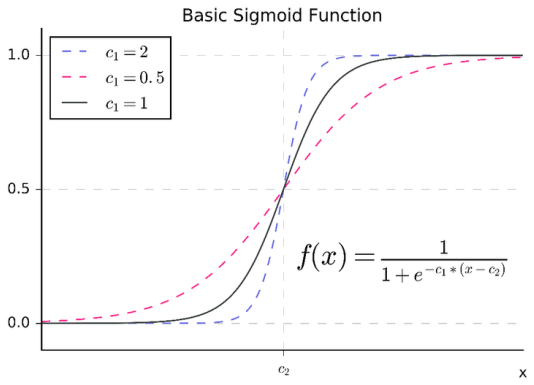
The model checks the performance with Cross-Entropy Loss Function, which is usually suitable for the data that will be split into more than three classes. There's another loss function called the Binary Cross Entropy that much more fits our images, which only classifies the data into two classes (0 or 1). Thus, we set the Binary Cross Entropy to amend this problem.

v. Batch size

Enlarging the batch size means feeding more data for training at one time. Owing to the smaller difference between each batch when using larger batch size, the gradient will become more smooth and stable when training. Therefore, we increase the batch size for a better outcome.

vi. Binary Optimization

Implementation of the sigmoid function is to make the outcome fit 0 or 1 more, which is more appropriate for our binary images. Also, we do the rounding to make the outcome split to 0 or 1.



b. Future work

i. Data amount

From the outcome that the performance of testing is better than training, we presume that this result comes from the data deficient. So the most fundamental solution is to increase the amount of data to improve the performance.

ii. Precision improvement of manual segmentation

When doing the manual segmentation, we noticed that there's a lot of difference in the definition of mitochondria contour

between different people. Also, when dealing with the image with drugs applied, we can hardly segment the mitochondria precisely. Thus, the precision improvement of manual segmentation will be a method to optimize the performance.

iii. Convolution layer

The model we use now owns 4 layers of convolution, and we think that the increasing of convolution layers may improve the performance.

iv. Model

The performance of this project is much better than we expected, even better than manual segmentation. We also found that the situation of ambiguous segmentation is normal for the sack of drug applied, which causes the overlapping of different mitochondria. We presumed that the U-Net model had already reached its limit for doing mitochondria segmentation. Thus, we may try another model to meet the precise segmentation of mitochondria.

5. Reference

a. Shen, K. (2018, June 20). *Effect of batch size on training dynamics - Mini Distill*.Medium.https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e

b. *torchvision.transforms — Torchvision 0.11.0 documentation*. (2017b). PyTorch. Retrieved January 2022, from https://pytorch.org/vision/stable/transforms.html

c. Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.

d. Lefebvre, Austin E. Y. T. (2021, August 19). Michelle Digman. Nature. https://www.nature.com/articles/s41592-021-01234-z?error=cookies_not_supported&code=473c581f-21a7-4198-ae2a-ce4fb546f0aa

e. Bruijne, D. M., Cattin, P. C., Cotin, S., Padoy, N., Speidel, S., Zheng, Y., & Essert, C. (2021). Medical Image Computing and Computer Assisted Intervention – MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, . . . Part VI (Lecture Notes in Computer Science) (1st ed. 2021 ed.). Springer.

f. M. (2021). *GitHub - milesial/Pytorch-UNet: PyTorch implementation of the U-Net for image semantic segmentation with high quality images*. GitHub. https://github.com/milesial/Pytorch-UNet

6. Collaboration

Tasks	Contributors
Data labeling first version	Min-Zhi Gao
Data labeling second version	Hsu-Ting Kuo, Min-Zhi Gao

Split_data.py, Data.py, Test.py	Hsu-Ting Kuo
Train.py, Model.py	Hsu-Ting Kuo, Min-Zhi Gao
Automated cropping	Hsu-Ting Kuo
Results organization	Min-Zhi Gao
Final Project Slide	Min-Zhi Gao