BIOMEDICAL SEGMENTATION ON CELL AND BRAIN IMAGES

A Thesis

by

BOWEN LAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Sunil Khatri
Co-Chair of Committee,	Anxiao Jiang
Committee Members,	Peng Li
Head of Department,	Miroslav M. Begovic

December 2019

Major Subject: Computer Engineering

ABSTRACT

The biomedical imaging techniques grow rapidly and output big amount of data quickly in the recent years. But image segmentation, one of the most important and fundamental biomedical data analysis techniques, is still time-consuming for human annotators. Therefore, there is an urgent need for segmentation to be taken by machine automatically. Segmentation is essential for biomedical image analysis and could help researchers to gain further diagnostic insights.

This paper has three topics under biomedical image segmentation scenario. For the first topic, we examine a popular deep learning structure for segmentation task, U-Net, and modify it for our task on bacteria cell images by using boundary label setting and weighted loss function. Compared to the MATLAB segmentation program used before, the new deep learning method improves the performance in terms of object-level evaluation metrics.

For the second topic, we participate into a brain image segmentation challenge which aims for helping neuroscientists to segment the membrane from neurites in order to get the reconstruction of neurites circuit. Data augmentation tricks and multiple loss functions are examined for improving the test performance and finally using combined loss functions can out-perform the original U-Net result in terms of the official ranking metric. A new dice loss is designed to focus more on the hard to segment class.

The third topic is to apply the unsupervised segmentation method which will not be restrained by human labelling speed and effort. This is meaningful under biomedical segmentation scenario where training data with expert labelling is always lacking. Without using any labelled data, the unsupervised method, Double DIP, performs better than the MATLAB program on the semantic level.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Sunil Khatri and Professor Peng Li of the Department of Electrical and Computer Engineering and Professor Anxiao Jiang of the Department of Computer Science and Engineering.

The cell data analyzed for Chapter 2 and 4 was provided by Professor Lanying Zeng of the Department of Biochemistry and Biophysics.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

No outside funding was received for the research and writing of this document.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Background

High-resolution biomedical images can be acquired quickly with advanced biology microscopy imaging techniques in today's world, but analysing such data can be time-consuming for researchers. Because of the need for professionalism of biomedical image annotation, usually only human experts with the related knowledge can give the accurate annotations. And they still need large amount of time to annotate. If the annotation process becomes automatic, less human efforts and lower cost can be achieved, and it can be act as an assisted role to reduce human mistakes. This work focuses on design and improve an automatic deep learning segmentation method on two kinds of biomedical microscopy images, (bacteria) cell images and brain (fruit fly nerve) images. A good automated segmentation method can have a lot of potential applications including counting normal or dead cells, segmentation of desired regions or objects, re-constructing neural connections and using these preliminary results to perform other meaningful analysis such as to classify cancer grades and understand nervous system etc. [5, 6].

Segmentation of biomedical images can be very challenging for computer and sometimes even for human experts [7], because of the various visual characteristics in color, shape and texture in the images which will be shown in next section. On the other hand, a large dataset with ground truth is always preferred for neural network to get best performance. But for the problem of segmentation on biology microscopy images, usually only limited number of manually segmented images are freely available and doing hand labelling of biology image consumes much manpower and time. So the main challenges we have here are the essential complexity of biology images and scarcity of labelled data. For traditional image segmentation algorithms like Otsu's method [8], K-mean clustering [9], Grab Cut [10] and etc, there are mainly two weaknesses explained in [5]. First is it can only be effective for limited types of images and very sensitive to manually set parameters. Second is it probably need semi-automated correcting afterwards which would be a burden for

researchers because of the huge amount and high resolution of biology data.

Fortunately, deep learning becomes dominating in the computer vision for the past few years, as it can usually achieve higher accuracy and quicker speed than traditional method. There is already one branch in deep learning area focusing on segmentation task which need the classification and localization of each pixel in the desired image. This work learns from two fundamental works in this branch which are "fully convolutional network" FCN [2] and U-Net [3] and examines modifying and tuning different parts of these framework to get a good model we want. Such supervised method needs big training data which is hard to get, so we also find an unsupervised method to do semantic segmentation on cell images without the need for labelled data.

1.2 Problems Definitions

This work includes two kinds of biomedical image data, cell and brain. Two deep learning method, supervised and unsupervised structures are examined to tackle different problems of segmentation tasks on the two datasets. Totally three topics are covered, each topic's problem and challenges are defined here .

Cell data are fluorescence microscopy images of E.coli bacteria cells provided by Zeng lab in Texas A&M University's biochemistry/biophysics department. Previously, a MATLAB program is used to do automatic segmentation first and followed by human correction. The program's automatic segmentation results usually have problems which may cost long-time manual correction. It is shown in Figure 1.1 that target cells on the border of the image are missing and surrounding area of cells is identified wrongly as cell instance. Such problem is caused by the inconsistent background fluorescence or color of the image. Another challenge is there are lots of touching cells, especially for some cells undergoing splitting process. These sometimes are expected to be separated as two instances.

The training data of brain image is a set of section images from a serial section Transmission Electron Microscopy (ssTEM) data set of the Drosophila first instar larva ventral nerve cord (VNC) [11]. The expert labels are provided in the format of white for the pixels of segmented neuron cell objects and black for the membranes (borders of neurites) [1] . This dataset looks harder than

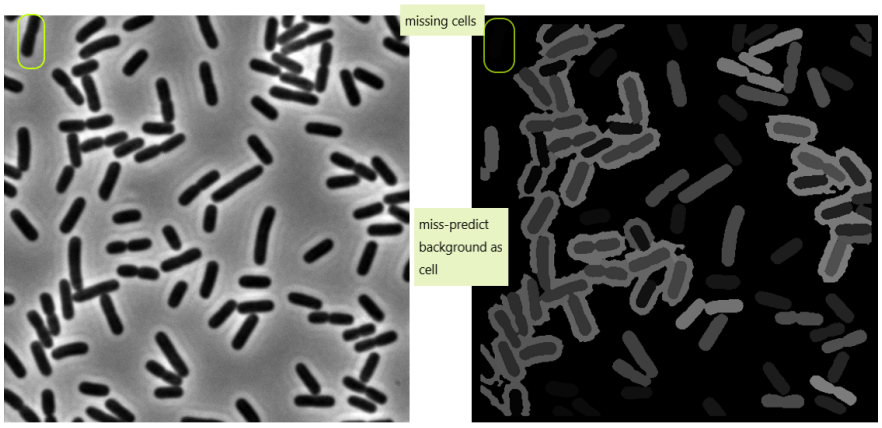


Figure 1.1: left: original image, right: MATLAB program automatic instance segmentation results; Problems in the results: missing few cells; miss-predict the background area surrounding cell where the color is inconsistent

the cell dataset since there are more complexities within the images. It can be seen in Figure 1.2 that there exist fuzzy and low contrast parts and inside compartments similar to the target borders which should be ignored when doing segmentation. This image is a projection of the 2D section, so some of the membranes that are not orthogonal to the cutting plane appear blurry and fuzzy [1].

Only 339 images of cell data and 30 images of brain data can be used for the above supervised segmentation tasks here. Such scale of data is limited for deep learning method which usually requires thousands of images, but this is an inherent problem for biomedical image segmentation. The reason is that getting and annotating biomedical images are more complicated and time-consuming than labelling natural images [12]. Human labelling costs too much time, for the brain dataset we use, one and half days are needed for the labelling of 30 section brain datasets [1]; for the cell dataset, it is not hard to imagine human correction time of thousands of cell images which are the needed amount of data for just one research project in Zeng Lab. Therefore, an unsupervised method which doesn't require any labelled data to give a segmentation result comes to our interest. Here the problem simplifies to: given one original cell image, can we get a mask layer to segment all the cells out. As far as we know, there hasn't been such attempt in cell segmentation

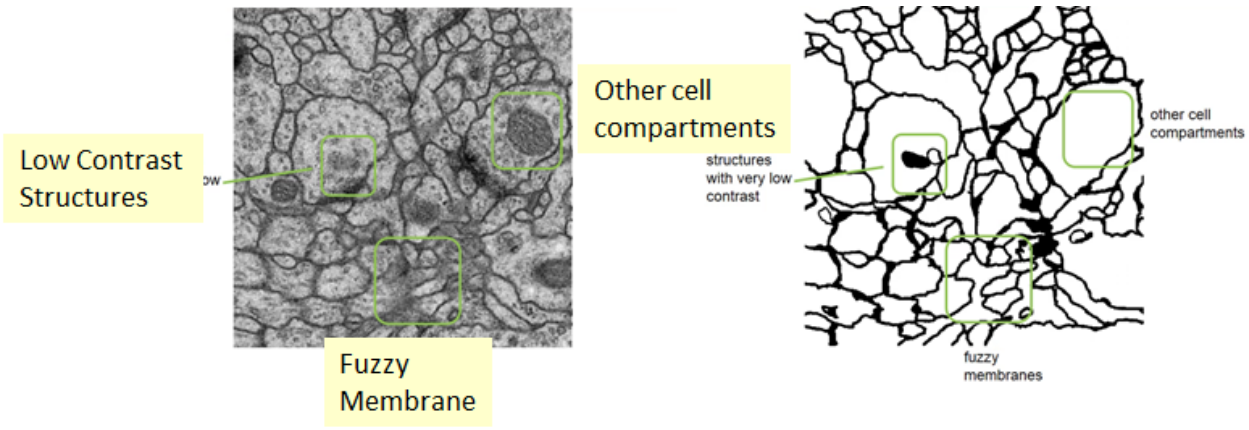


Figure 1.2: left: original image; right: ground truth label; Challenges in brain image: low contrast and fuzzy appearance; compartments inside neuron having similar color characteristic as membrane of neuron. Images adapted from source [1]

area yet.

1.3 Related Works

The MATLAB based software Zeng Lab used to do segmentation is called Schnitzcells which is developed specifically for bacteria and has been instrumental in analyzing E.coli and B. subtilis movies [13]. It helps to get segmentation for bacteria cell and do quantitative analysis of living cells. It has an useful GUI for researchers to view the automatic segmentation results and edit the problem manually. Human correction is very time-consuming which may takes hours for few images, so this segmentation method is not scalable with the increasing amount of data generated in the lab. Moreover, the automatic segmentation results are sometimes very bad which causes a lot of post-processing time for human researchers to correct them back.

Another related method is to use deep learning, convolutional neural networks (CNNs) to do semantic segmentation tasks in recent years. Since CNNs are first designed to do classification task with fully connected layer in the end, Ciresan et al. adopted a sliding window based method to perform segmentation by pixel classification on each patch around the center pixel [14]. The fully convolutional network without any fully connected layers is proposed by Long et al. in 2015 to perform end-to-end full image segmentation, and it surpassed all the previous approaches in an

elegant and efficient way [2]. FCN type network have also shown great promise in doing biomedical image segmentation[15], and most of the credits go to the U-Net [3]. U-Net and U-Net like models have been successfully used in segmenting biomedical images of neuronal structures [3], skin lesion [16], etc. The structure of U-Net is based on FCN, using an encoder network with a decoder network improved from the naive upsampling path of FCN. Furthermore, the corresponding same size layers of the encoder and decoder are connected by skip connections to be concatenated, prior to the max pooling layer and subsequent to a upsampling operation respectively [17].

1.4 Organization of Thesis

1. Section I Introduction: This section gives background and definition of the problem we learned and introduces related works to show the state of the biomedical segmentation field. The challenges of the problems lead to following three topics. Our main goal is to improve the performance compared to existing backbone and get meaningful observation of the specific way for improvements.

2. Section II Supervised Learning for Cell Segmentation: first overview the current standard deep learning method U-Net and based on the baseline framework, modify it for the task of cell segmentation to have better performance than MATLAB program.

3. Section III Supervised Learning for Brain Segmentation: simple modification on U-Net structure, data augmentation tricks and several loss functions are studied to improve performance compared to the original U-Net.

4. Section IV Unsupervised Learning for Cell Segmentation: applied an unsupervised deep learning framework on cell semantic segmentation, doing metric comparison with MATLAB results on a semantic level.

5. Section V Conclusions and Future Works: we will conclude the observation and contributions for each of the topic and propose possible directions for the next step of segmentation on such data, especially for the current naive unsupervised method.

2. SUPERVISED LEARNING FOR CELL SEGMENTATION

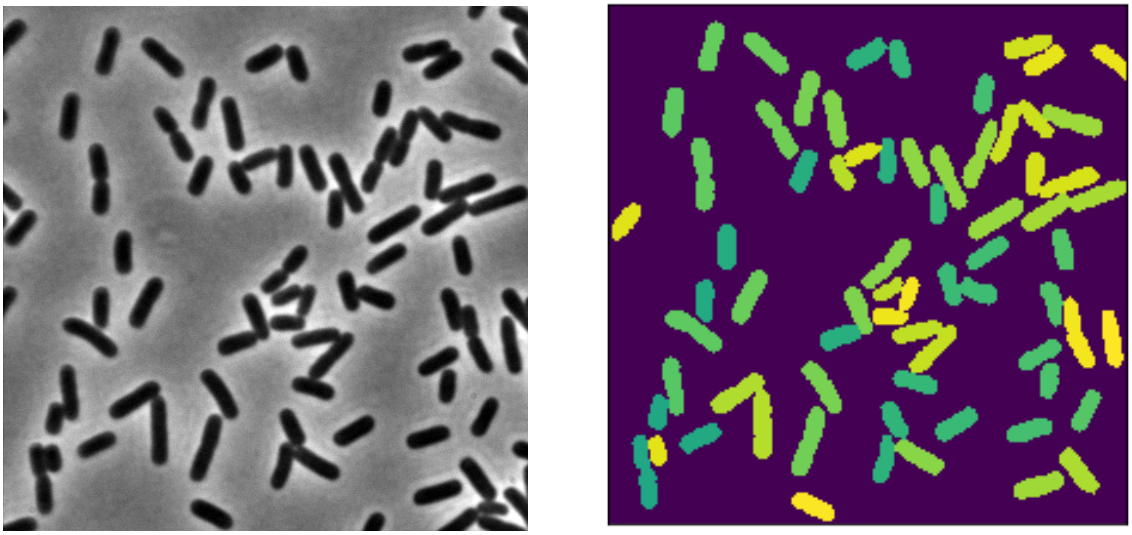
The bacteria cell datasets we do experiments on are all from Zeng Lab of Biochemistry-and-Biophysics department in Texas A&M University. They labelled the cell image data with Schnitz-cells program [13] introduced in first chapter and perform human correction on the raw annotations of the traditional segmentation algorithm. As they described the segmentation process of using the program, automatic segmentation takes about one minute per image and precisely labelling one cell image could take hours. So our objective is to design an automatic image segmentation tool for them based on the deep learning structure, U-Net [3] and achieve better performance as well as quick speed on the provided cell type data. The raw data we got is in mat file format, python script is written to extract the annotation matrices out and save them corresponding to the original cell images in one directory named "FISH" (Fluorescence In Situ Hybridization) which is the name of their imaging technique. The first 48 images come with raw MATLAB automatic segmentation without human correction, so these 48 images will be held out as validation set to compare the performance of our designed method to the MATLAB program. Remaining 291 images are used as our training set. Average number of cells on one image is 65 and the area of cells range from 600 to 1500 which doesn't include any tiny target. An example of the given data is shown in Figure 2.1. In human annotation, each cell is labelled with a different classes so shown in different color.

2.1 Overview of Deep Learning Structures

More detailed background of the deep learning structure, U-Net, we used for this task are presented here.

2.1.1 Fully Convolutional Network

The image segmentation branch of computer vision using deep learning begins to prosper after this end-to-end approach, Fully Convolutional Network (FCN) first introduced in 2015 [2]. The authors replace all the fully connected layers in the well-known architectures of that time by convolutional layers to make the network have the ability to output an image but not one dimensional



(a) original image

(b) human annotation

Figure 2.1: One example of cell images data

vector. In order to learn image features on different levels, convolutional neural network uses downsampling operation, like the maxpooling layer, to get larger receptive fields for convolutional layers. This works very well for the task like classification because network can learn the abstract information of what is in the image but such downsize operation damages the location information within the image data. For segmentation task, the location information is important and need to be restored back after downsampling. So the network need to learn how to restore the downsized feature maps back to the same size of original input. The reverse of downsampling operation, upsampling operation, is introduced to increase the size of feature map [2]. Upsampling layers will interpolate the input feature map by the way of duplicate the nearest pixel or biliear interpolation to increase the size by normally a scale of two. An advanced upsampling layer is later devised, commonly called deconvolution layer [18], which has learnable filters to do the equivalent opposite operation to convolution with strides. Generally, the downsampling path is seen as an encoder and the upsampling path is seen as a decoder, and this way of design gives the network same size for the input and output, so that it can be trained using a pixel-wise loss with the ground truth label. Compared to pixel predictor design [14] using sliding-window approach, FCN is more elegant and

efficient, also with better performance. An illustration of FCN design is in Figure 2.2

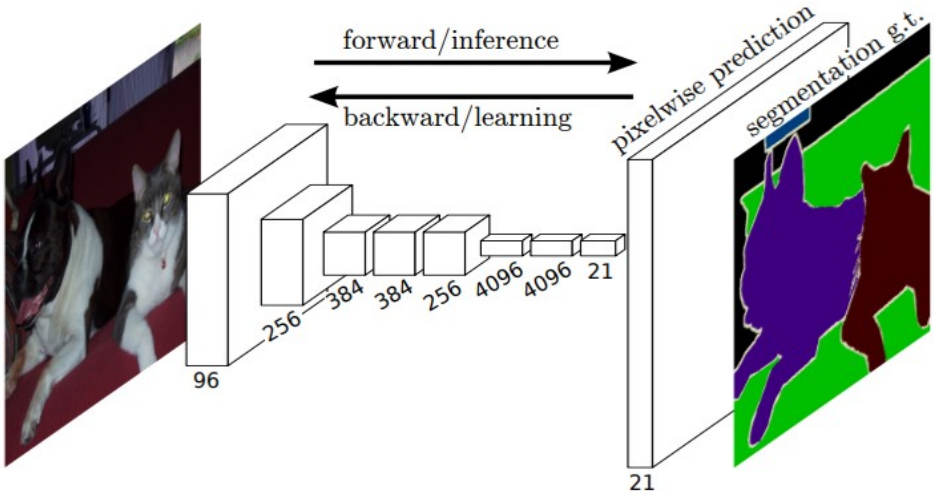


Figure 2.2: Fully Convolution Network output pixel-wise prediction Illustration. Image reprinted from [2]

2.1.2 U-Net Structure

Based on FCN network architecture, U-Net improves the network design with a symmetric structure and skip connection which is now one of the most popular method in the biomedical image segmentation field [3]. It can be seen in Figure 2.3 that the overall shape of the structure looks like a big letter 'U' and that is why this design is called U-Net. It consists of a contraction path on the left and an expansion path on the right, corresponding to encoder and decoder. Contraction path consists of consecutive blocks of two times of 3×3 convolutional layers and 2×2 max pooling layer [19]. This can help to extract more advanced features but it also reduce the size of feature maps. This can be intuitively understand as the network learns more of what is in the image from advanced information but loses information of where the object locates in the image. Expansion path consists of consecutive blocks of 2×2 deconvolutional layer and two times of 3×3 convolutional layers to recover back to the size of segmentation map. Skip connections are added in the network between the contraction path and expansion path to combine same-size fea-

ture maps. This skip connection design means to help upsampling path better recover the location information from the earlier levels in contraction path. Such skip connection mechanism also inspired the famous residual networks [20] which uses skip connection to make the network deeper a lot. At the end this network design, 1×1 convolutional layer is to reduce feature map channel number from 64 to the label channel number which depends on how the task is defined.

The original U-Net structure downsizes the input image 4 times in the contraction path and then upsampled them back. Although skip connections are used between the layers of same size from, we have doubt that the learning ability of such depths may not be fully utilized for our tasks, namely, cell and brain segmentation. Discussion for how to choose the depth of the U-Net is in the next two chapters.

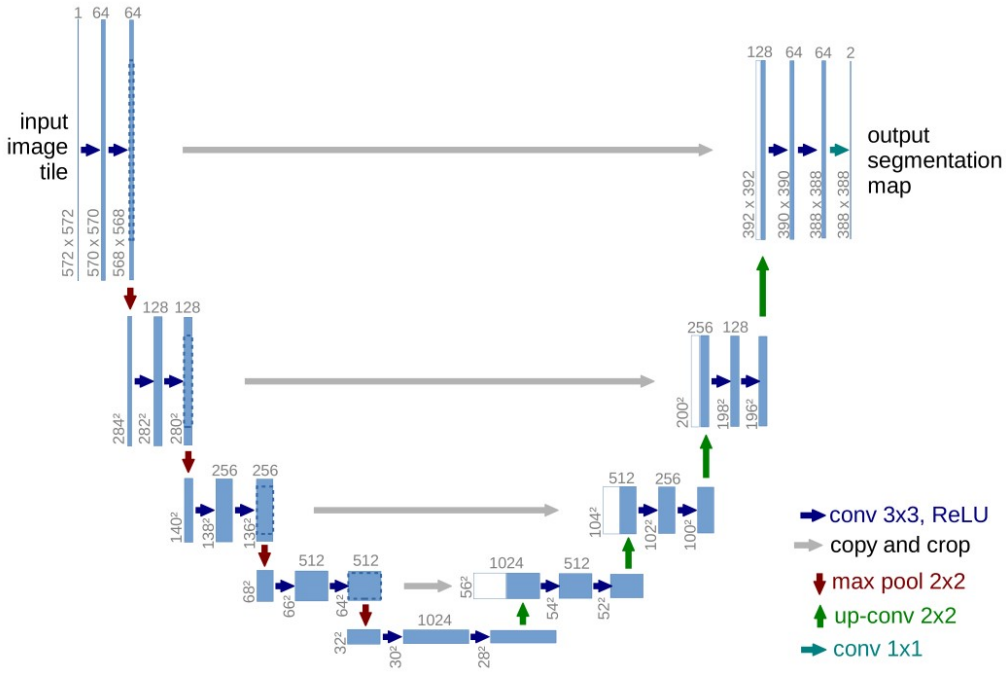


Figure 2.3: Original U-Net architecture. Image reprinted from [3]

2.1.3 Network Depth

As we downsized the original cell image, it can be seen by eyes that the information for cell shapes, boundaries and relation between adjacent cells loses a lot, after downsize three times, each time by a scale of 2. The number of pixels for one cell, especially on the border of one cell, are very few and this could not bring useful information into the whole network training. It should be mentioned that the border of the cell is one of the most important information for the network to discriminates different cell objects in our task. So a shallow depth of U-Net structure which contains only 2 maxpooling layers is chosen and we do comparison experiments with U-Net of depth 3 to show that deeper network here is not able to perform better but instead a little worse which verify our doubt that some learning capability is wasted.

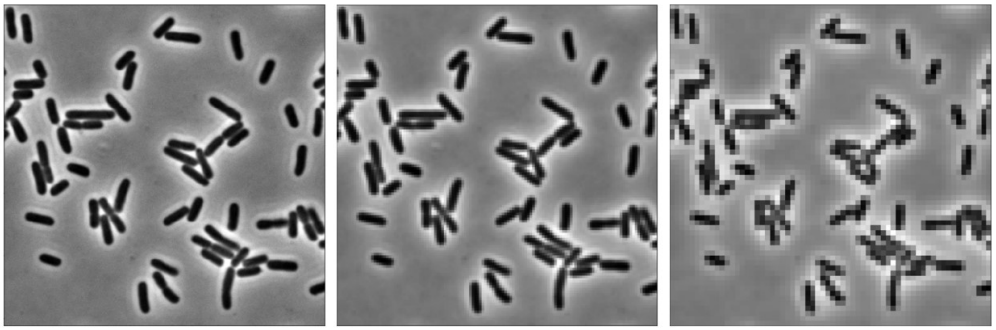


Figure 2.4: left: original image 512x512 resolution; middle: 128x128 resolution; right: 64x64 resolution

2.2 Label channel addition and corresponding weighted Loss Function

2.2.1 cell, background and boundary annotations

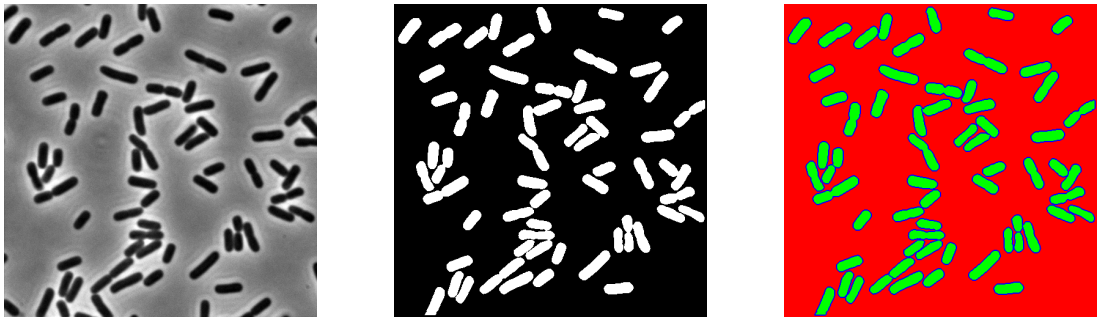
For our cell segmentation task, the wanted output is the instance segmentation of cell images which requires different classes assigned to different cells, but U-Net is designed as a semantic segmentation network which means it can only assign the same class to the same category, so all the cells will be classified into one class without different indexes for them. The first thing we need

to do is to transform this problem from instance segmentation to semantic segmentation suited for U-Net. The raw annotations with different numbers representing different cells are transformed into binary labels, 0 for background and 1 for all cells shown in the middle picture of Figure 2.5. This can be easily transformed back to instance label by using labelling algorithm [21] in python skimage package under the condition that predicted cells have no merging region. Binary cross entropy loss is used to do the training for this label setting. Loss function for each pixel is shown in Equation 2.1. p and p' are the probability of prediction and ground truth respectively. The overall loss value is the average or summation of all pixel loss values. However, such configuration with only two classes, background and cell, gives us too many merger errors since one of the challenges in the cell image is that there are many adjacent and touching cells. It can be seen in Figure 2.6, the middle picture is the output of binary label setting, the cell cluster on the middle region is not separated out correctly, all with the same label number so same color for them.

$$\text{binary cross entropy} = \begin{cases} -\log\left(p'\right), p = 1 \\ -\log\left(1 - p'\right), p = 0 \end{cases} \tag{2.1}$$

The information of how to separate between cells should be given into the network explicitly. Inspired by the 1st place solution of 2018 data science bowl [22] and U-Net paper[3], a boundary channel is added along with background, cell interior channels. The surrounding pixels around the cells are defined to be boundary, having value 1 in the boundary channel, all the pixels of cells inside boundary are set to value 1 in cell channel. The background class is the remaining pixels outside of boundary. So our label becomes a 3 channel image, each channel represent background cell and boundary respectively. An image illustration is given in Figure 2.5 with rgb colors for background, cell and boundary respectively. Loss function is changed into multi-class version of cross entropy loss shown in equation 2.2, M is 3 for three classes label setting. After using separated channels for three different classes, many merger errors are corrected. It can be seen in the right picture of Figure 2.6, on the up left middle part of the results, a cluster of merged cells in the middle becomes separated clearly.

$$\text{multi-class cross entropy} = \sum_{c=1}^{M} p_c \log\left(p'_c\right) \tag{2.2}$$



(a) (b) (c)

Figure 2.5: (a) original image (b) binary label (c) 3 channel label (adding boundary class)

2.2.2 Weighted Loss Function for Unbalance Problem

Adding a background channel ameliorates the merger error, however, after doing a statistic analysis, the number of merger errors is still more than MATLAB results. So how the cross entropy loss function works are examined. This loss calculates difference on each pixel between the network output and the ground truth annotation. On each channel, a binary representation is set

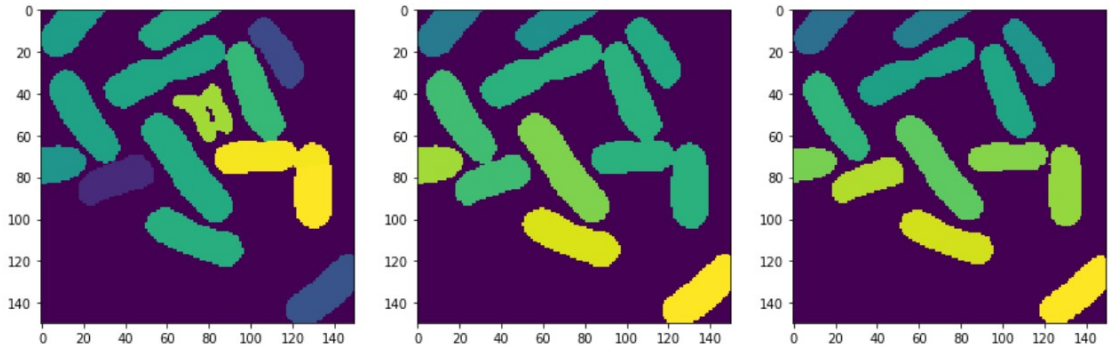


Figure 2.6: Compare MATLAB results, binary label results and 3 channels label results, from left to right respectively

for whether this pixel is background, cell, boundary or not. For the added boundary channel, the numbers of 0 and 1 are highly unbalanced since the boundaries are only one or two pixel width around cells.So the positive pixels which meaning it belongs to boundary are much less than the negative pixels in the boundary channel. And this causes the network tending to predict more negative pixels corresponds to value 0 which means not boundary, because this is an easier way to get a lower loss. A good solution to cancel out this unbalance problem is to add more weights onto the boundary channel. For the 3 channels of the output, a weight is multiplied on it after calculating the multi-class cross entropy loss. We first add weight 10 onto the boundary channel and achieved good improvements. A result sample is shown in Figure 2.7, using weighted loss can separate better than both no weight version or matlab result. Then the weight ratio for three channel is chosen as 1:3:15 according to the reverse ratio between the number of pixels of three different classes. This is intuitive so succeeding experiments with different ratios are performed to see how to choose the weight for best performance. We finally find out a trad-off relation between split and merger errors controlled by different weights.

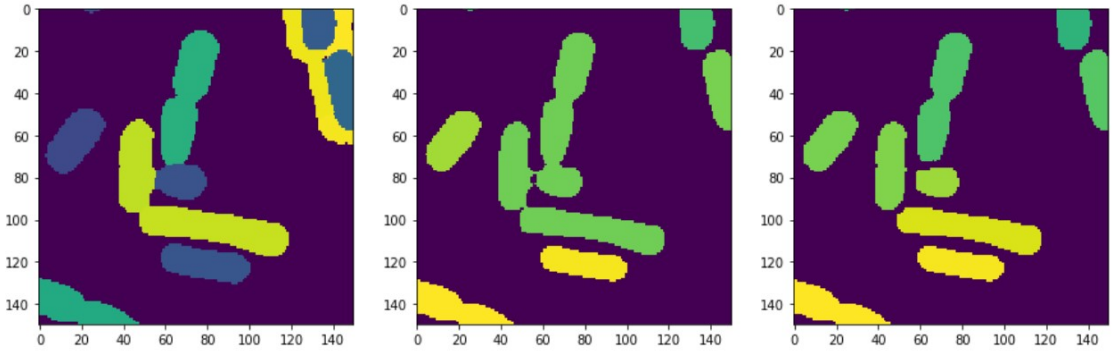


Figure 2.7: Zoomed in comparison after giving more weight to boundary channel

2.2.3 Training

When performing the experiments, several hyper parameters are chosen as below. Random crop of original images can give more training samples to ameliorate the problem of lacking data.

Another way for this problem is using some data augmentation techniques, flip, rotation and adding Gaussian noise are used to train the rotation invariance of the network for biomedical data. Each epoch will be trained 300 steps with batch size of 16 so totally 4800 samples can be seen by the network in each epoch. Early stop monitor the validation loss and if it doesn't decrease for 10 epochs, the network will be stopped immediately before the 200 training epochs. Learning rate scheduler also monitors validation loss, if it doesn't decrease for 3 epochs, the learning rate will be multiplied by the factor. Tensorboard callback is used to monitor the overall training performance, precision and accuracy for each channel and overall categorical accuracy are recorded into the tensorboard interface. When doing prediction, the whole image will always be used as input on validation set.

- random crop size: 256
- batch size: 16
- training epochs: 200
- step per epoch: 300
- early stop patience: 10
- reduce learning rate factor: 0.2

2.3 Evaluation Metrics and Results Analysis

Apart from pixel accuracy, the object-level accuracy is relatively more important for medical research, because further biomedical research is performed on the individual level. Intersection of Union (IoU) match, one of the common object level metrics is chosen as the evaluation metrics for our task. The equation of IoU is given below.

$$\mathrm{IoU}(A, B) = \frac{A \cap B}{A \cup B} \tag{2.3}$$

Here A and B means ground truth area and predicting area. When the overlapping between A and B is more than a threshold, this prediction, B, is called a IoU match. For example, at the threshold of 0.7, one predicted area is considered a match with the ground truth object if their IoU is greater than 70%. The thresholds are a range of discrete value, at each value, the average of higher-level metrics for the validation set are calculated and shown in table 2.1. We calculate on the threshold values range from 0.5 to 0.9 with a step size of 0.05 same as in [23]. Then three basic metrics, true positive (TP), false positive (FP) and false negative (FN), can be defined as one and only one match, extra prediction match no ground truth area, missed ground truth area with no prediction match respectively based on IoU [23]. These three values are used to calculate the precision, recall and F1 score on an object level. Another important aspect we cared about are the split and merger errors which can also be defined by IoU match, split meaning one ground truth object matches more than one predictions and mergers are one predict area matches more than one ground truth region.

In table 2.1, binary label setting which does not includes boundary information is the worst. After adding boundary channel, the performance is improved a lot and becomes better than MATLAB results in several metrics except Recall. Finally, using a higher weight on the unbalanced boundary channel gives us the best result, all better than MATLAB results. So this results prove our experiments to be reasonable and meaningful. It is probably used to replace the previous MATLAB program to do the automatic segmentation task on the cell images for Zeng Lab.

Experiment Name	MATLAB	Binary Label	1:1:1	1:1:10
F1 score	0.873	0.551	0.874	0.941
IoU score	0.810	0.707	0.875	0.892
Precision	0.806	0.660	0.901	0.929
Recall	0.959	0.480	0.850	0.960

Table 2.1: Metrics Comparison between MATLAB and different label and loss settings results. 1:1:1 and 1:1:10 are the weight of loss function when using 3 channel label setting

2.3.1 Trade-off between split and merger errors

After doing experiments with different weight ratio settings shown in table 2.2 and 2.3, a pattern shows that fixing weight on the first two classes, higher weight on the boundary class leads to higher split errors along with less merger errors. Also more extra cells, (higher false positive rate) are related to higher split errors, same for missing cells corresponding to merger errors. As we discussed with the student in Zeng Lab, the weight should be a chosen hyper-parameter depending on the personal criteria of the researcher. Different people may have different need of split and merge performance of their biomedical research.

Experiment Name	1:3:5	1:3:10	1:3:15	1:3:30
Splits	9	27	45	90
Mergers	30	12	6	4
Extra	160	173	212	237
Miss	130	98	97	130

Table 2.2: Trade-Off between Splits and Mergers errors when giving different weights on 3 channels label, fix 1 and 3 as weights for background and cell classes, increase weight for boundary class

Experiment Name	1:5:10	1:5:15	1:5:20
Splits	10	20	34
Mergers	28	20	11
Extra	157	179	196
Miss	127	106	102

Table 2.3: Trade-Off between Splits and Mergers errors when giving different weights on 3 channels label, fix 1 and 5 as weights for background and cell classes, increase weight for boundary class

3. SUPERVISED LEARNING FOR BRAIN SEGMENTATION

The brain data we work on is from the first international competition on 2D segmentation of electron microscopic (EM) images of the fly brain [1]. The importance of electron microscopy (EM) imaging technique is described in [1]. This technique has advanced the research on synapses and other subcellular structures. In 2012, the connectivity within elegans nervous system is reconstructed on serial EM technique[24]. However, since human labelling on such images are too time-consuming, there is an urgent need for the development of an automatic machine aid program for the analysis of EM images. An example showing how time-consuming this task is is in [25], a reconstructed area of a mouse retina costs 20,000 hours of experts' time but this area is only enough to encompass the smallest types of retinal neurons. In other word, without the automation or semi-automation, the reconstruction process of EM images will require much more of human effort and this will be the biggest bottleneck for the research of this field.

This competition aims for appealing machine learning experts to the task of image segmentation which can further help brain reconstruction connectomics. Anyone can submit and test his predict result to the online server, and the result will get scored compared with human expert annotations hidden from participants. We do experiments on the training data they provided with human expert label and submit the best configuration of our experiments to the test server.

An example of the brain images is shown in Figure 3.1 which are total 30 consecutive images (512×512 pixels). The goal is to assign each pixel to either value "0" for belonging to the membrane (boundary) between neurons, or value "1" for belonging to the interior of the boundary, resulting into a binary image with "black" for neurites boundaries and "white" for all neurites. The right colored image in Figure 3.1 is the next step by giving each closed boundary region a different class shown in different colors. These instance segmentation sections can be further connected by merging same segmentation class in the z-direction of 30 consecutive images so that a 3D reconstruction can be built in the end. 3D reconstruction of neurites helps neuroscientists to study the structure of nerve system.

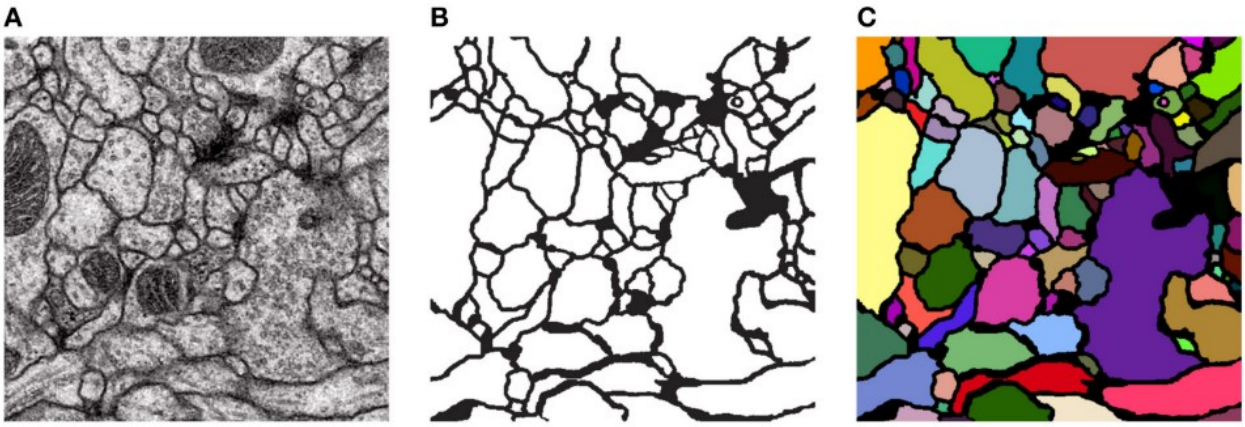


Figure 3.1: Image from [1]: A is the original image, B is the ground truth boundary label, C is the instance segmentation into neurite cross-sections

3.1 Network Structures

U-Net [3] was one of the participants in this challenge at that time. Since the online test server of this competition is still running today, we can do experiments with U-Net as baseline model to study how different modifications influence the overall network performance on this task.

First thing we think about is still the depth of the network. As shown in the previous chapter, U-Net uses 4 times downsampling operations but that could be a waste of computation resources without learning effective information. A comparison of resized brain images is shown in Figure 3.2, after 3 times of downsampling by a scale of 2, the image becomes blurry and most importantly, some small but complex part of boundaries of the neurites in black color are losing too much information. This loss of information will damage the prediction performance on those small intricate region. Even with learned deconvolution layer and skip connections, lost information can not be ensured to recover back. So we assume depth more than 3 cannot introduce more useful information into the network and chose 3 as our network depth. There will be test results in the following result analysis section showing that shallow network perform even better than deeper network using 4 as the depth.

Compared to the cell images in the last topic, the brain images are more complicated and irregular. There are more compartments of fly nerve than bacteria cell and all neuron cells are clustered

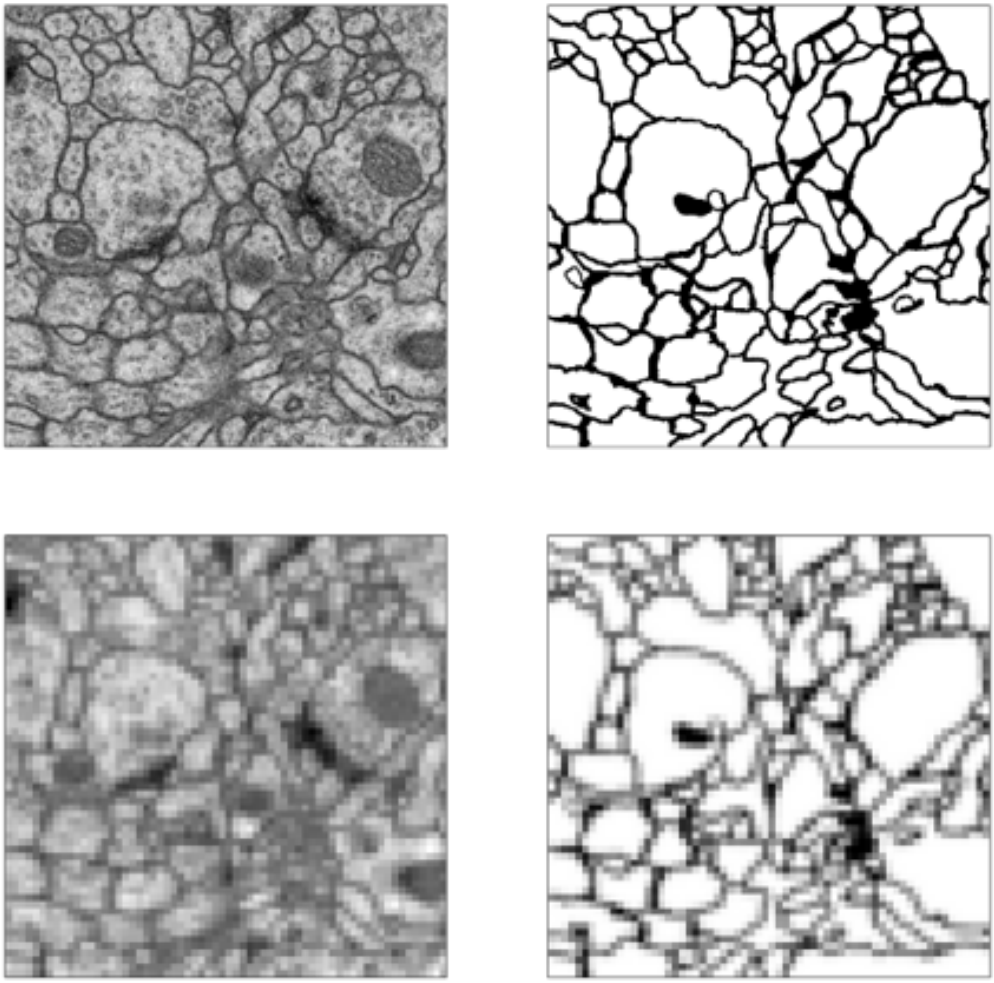


Figure 3.2: Original data compared with data downsized 3 times, each time by a scale of 2

together without any regular patterns. So on the expansion path which is for recovering the features localization information back, more network capability corresponding to larger number of channels is added to help the network learn the complicated detailed information of boundaries. The channel number of convolutional layers in expansion path are doubled and this improves the performance on the test set. Dropout layers are added for the lowest three blocks of the U-Net to overcome the problem of overfitting on small training set. If adding more dropout layers to the higher blocks of the U-Net, it will cause a degeneration of the network performance.

3.2 Data Augmentation

Three hundreds of cell images are already short for the deep learning training, only 30 images are provided for brain. And one third of them are selected out as the validation set to help us determine the training performance when doing experiments. So intensive data augmentation tricks are used on this task. Apart from the first step of data augmentation including simple flip and rotation operations, we use uniform and Gaussian noises addition, brightness change and deformation as the second step. In the second step, these transformations are not performed on the ground truth labels but only on the original images. Theoretically, these transformations let the network see more possible situations of the brain images and it can be seen in the result analysis section that such data augmentation tricks improve the overall performance. Deformation using here is to give displacements to the coordinates of the image, and map the pixels of the original image corresponding to the deformed new coordinates onto a regular coordinate . The displaced coordinates is used to find the corresponding pixel value in the original input to be mapped to the output image. The value of these found displaced pixel value is determined by spline interpolation if it doesn't fit into another pixel position, in the order of output coordinates. This mapping function is provided in python scipy package. An example of such transformation is shown in Figure 3.3. Different sigma and alpha (standard deviation and scaling factor) setting for Gaussian filter lead to different intensity of the deformation. The left picture uses a larger setting than the right picture in Figrue 3.3.

3.3 Loss Functions

In the previous chapter for cell segmentation, weighted cross entropy loss is a pixel-level loss and used to overcome the problem of unbalance ratio between boundary and background classes. The same problem appears here for brain segmentation with smaller area of boundary and larger area of neuron cell area. So weighted cross entropy loss should also be used here and the weight is determined by the reverse ratio between the boundaries and neurites areas which is 4 to 1. The equation 3.1 of the weighted cross entropy of each pixel is shown below. p is the label for ground

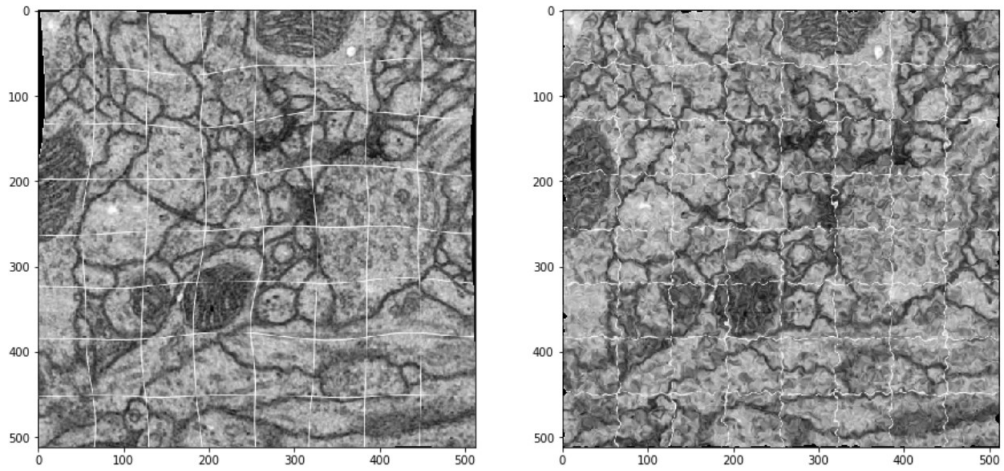


Figure 3.3: Illustration of deformations with coordinate lines showing what changes are applied; left: with large std(standard deviation) of Gaussian filter; right: with small std(standard deviation) of Gaussian filter

truth pixels and p' is the predicted probability. The overall loss value is the summation or average of all the pixel losses. Since this is a harder topic compared to cell segmentation, we decided to examine different loss functions and implemented them to study their influence on the network performance. Compared to modifying network structures, having a well designed loss function is more explainable. Below are the overviews of three more loss functions we examined.

$$\mathrm{WCE}(p, p') = -(p \log(p') + 4(1-p) \log(1-p')) \tag{3.1}$$

3.3.1 Focal Loss

Focal loss is an upgrade of cross entropy loss by adding exponential down-weight terms which can suppress the easy classified samples, so that to make the network training on the hard classified samples. In the equation 3.2 of focal loss of each pixel, $(1-p')^{\gamma}$ and $(p')^{\gamma}$ are the exponential down-weight terms. For example, when the true value of the pixel $p = 1$, the prediction of this pixel $p' = 0.97$, then this is an easy pixel and the down-weight term is very small for it, but if the prediction $p' = 0.3$, then this is a hard pixel with relatively higher factor.

$$FocalLoss = \begin{cases} -(1-p')^{\gamma}\log p' & p = 1 \\ -(p')^{\gamma}\log(1-p') & p = 0 \end{cases} \tag{3.2}$$

3.3.2 Object-Level Loss

Cross entropy loss family acts on the pixel level, but pixel accuracy doesn't directly correlated with the segmentation performance which can be measured by object-level metric IoU in the last chapter. This gives us the direction to examine object overlapping level losses for the network to directly optimize for object level metrics. There are two popular object level losses which are derived from two object level coefficients which can measure the similarity between two segmentation results. The equations for these two coefficients are given below which are the dice coefficient and the Tversky Index. A and B represents ground truth and prediction area respectively. The intersection of A and B is the true positive. $|A - B|$ and $|B - A|$ are false negative error and false positive error respectively. So α and β can be put in front of them depending on the user's perspective of which error is more important. When these two are both 0.5, tversky index becomes the dice coefficient. The losses derived from these coefficients are simply 1 - them, because we want the loss as small as possible, but coefficients as large as possible which meaning more overlapping area.

$$\text{Dice Coefficient(A, B)} = 2\frac{|A \cap B|}{|A| + |B|} \tag{3.3}$$

$$\text{Tversky Index(A, B)} = \frac{|A \cap B|}{|A \cap B| + \alpha|A - B| + \beta|B - A|} \tag{3.4}$$

3.3.3 Upgrade Dice Function

In this task, there are two classes, neuron and boundary. We think the difficulties to segment these two classes are different, segmenting the boundary should be harder because the boundary class takes less areas and there are more ambiguities. For the cross entropy loss, we adopted

weights to give different importance to different classes. For dice loss, we find a way to give different importance of different classes which is inspired by [26]. Adding different power terms for the loss functions of two classes, equation shown below in 3.5. We can plot the curve of dice coefficient to dice loss to the power of γ, shown in the Figure 3.4. It can be seen that when dice coefficient is high around 0.8, using a small γ makes the loss curve to have higher rate of change, conversely if using a gamma bigger than one. So when the training coming to the point of a coefficient more than 0.8, we use the small gamma to give more rate of change to the difficult class, boundary and large gamma for the other class. The reason we focus on coefficient more than 0.8 is that we observe the dice coefficient performance can be quickly trained to 0.8 and slow down afterwards.

$$\text{Upgrade Dice Loss } = (1 - \mathrm{DC}_0)^{\gamma_1} + (1 - \mathrm{DC}_0)^{\gamma_2} \tag{3.5}$$

3.4 Training

When performing the experiments, several hyper parameters are chosen as below. Random crops of image size 256 x 256 from the whole image with 512 x 512 pixels are used as input. Extensive data augmentation tricks performed on it, normally no data augmentation will be used when doing experiments on validation set because real-time data augmentation seems to be 5 times slower than no augmentation and longer training steps are needed for data augmentation. Each epoch will be trained 500 steps with batch size of 16. Early stop callback monitors the validation loss and if it does not decrease for 30 epochs,the network will be stopped immediately before the total 100 training epochs. Learning rate scheduler also monitors validation loss, if it does not decrease for 5 epochs, the learning rate will be divided by a scale of 2. Tensorboard callback is used to monitor the overall training performance, dice and jaccard scores as well as loss values are recorded into the tensorboard interface.

- random crop size: 256
- train valid split: first 10 as validation set, rest as train set

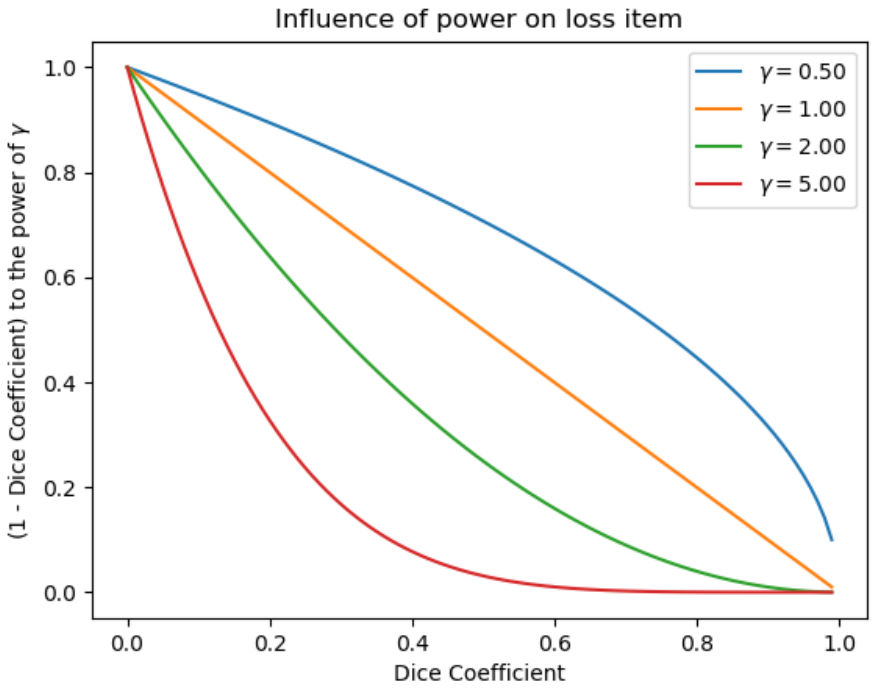


Figure 3.4: Curve of dice coefficient and power of dice loss (1 - dice coefficient) with different gamma value

- training epochs: 100
- steps per epoch: 500
- early stop patience: 30
- learning rate range: 1e-4 to 1e-6

3.5 Evaluation Metrics and Results Analysis

In the paper of brain competition [1], several metrics are examined thoroughly from theoretical and empirical perspectives. They choose the metrics based on the standard that it should indicate its potential utility in practical applications which can be helping human experts to segment images by just having small mistakes of the algorithm results [27]. Pixel error doesn't correlated to the good segmentation prediction result. Therefore, apart from the naive pixel error, rand score[28]

and variation of information [29, 30] are chosen to be the metrics for judging qualitatively of the object-level segmentation and rand score is official ranking score of the challenge. Since the scripts for calculating theses metrics are already given by the challenge, simple intuitive introductions for them are summarized here:

- pixel error: difference between the pixels of original label and the result
- warping error: the number of splits and mergers required to obtain the desired segmentation [31]
- Rand Score: measure the similarity of two segmentations [28]
- Information Theoretic Score: using mutual information and entropy to measure the similarity of two segmentations by considering split and merge errors [29]

Experiment Names	Rand Score	Information Theoretic score
unet4	0.963	0.981
unet3 symmetric	0.966	0.984
unet3 expansion	0.971	0.985
unet3 dropout5	0.955	0.982
unet3 data aug	0.973	0.986

Table 3.1: Compare the performance on test set for different structure modification, using depth 3 U-Net with 3 dropout layers and deeper expansion path is the best network structure; data augmentation can give small improvement

Performance measured by these four metrics for different structures we modified are shown in table 3.1. U-Net with depth 4 is a little worse than U-Net with depth 3. Adding more convolutional channels improves the performance by 0.05 on rand score. Using more than three dropout layers damages the overall performance a lot. Data augmentation only gives a small improvements for the best structure. For the following results of different loss functions, all experiments are performed on the best structure we found so far which is the unet3 with more expansion convolutional layers and using 3 dropout layers.

In table 3.2, the performance of four loss functions shows that there are not much difference between them, tversky and focal loss functions, only better than the normal cross entropy and dice loss by around 0.05 in the rand score. But in the next table 3.3, combined the pixel-level loss and object-level loss together give us satisfying improvements which is around 0.1 in the rand score. Two of them, weighted cross entropy combined with Tversky loss and focal loss combined with dice loss give the best validation performance and the later one gives the best test performance which is 0.979 for the rand score, and this test score improves the original U-Net performance we re-implemented by a margin of 0.13 on the test server. This makes us the top 25 in the leader-board of this challenge. Considering the top methods usually use some post-processing methods to improve their deep learning results, our result is satisfying.

Losses	wce	focal	dice	Tversky
Pixel Error(10e-3)	66.6	60.9	61.4	65.9
Warping Error(10e-6)	1530	1474	1567	1577
Rand score	0.945	0.947	0.946	0.949
Information Theoretic score	0.977	0.976	0.975	0.975

Table 3.2: Compare the performance on validation set for four losses: the performances of four different losses are similar

Combination of losses	wce+dice	wce+Tversky	focal+dice	focal+Tversky
Pixel Error(10e-3)	60.1	60.1	60	62.2
Warping Error(10e-6)	1400	1350	1445	1520
Rand score	0.95	0.957	0.957	0.94
Information Theoretic score	0.977	0.978	0.977	0.975

Table 3.3: Compare the performance on validation set for combined losses: combined losses like add weighted cross entropy and dice loss together can give good improvement

In table 3.4 for the upgrade dice loss function, different choices of γ are tested. The performance shows that using 0.5 for the loss term of class 0 is better than using a higher value as we

wanted. The reason is in Figure 3.5, when the dice coefficient is bigger than 0.8, a small γ gives the loss term higher rate of change, and big γ gives the loss term small rate of change. And this is what we assume to suppress the easy to classify neuron cell class and give more weight to the boundary class. Then we fix the value of γ_1 to be 0 and increase the value of γ_2 from 0.5 to 10, the best pair we finally have is 0.5 and 2 for γ_1 and γ_2 respectively.

Two Gamma for upgrade dice loss	5;0.5	1;1	0.5;5	5;5	0.5;0.5
Pixel Error(10e-3)	60.4	61.4	61.5	75.5	62.3
Warping Error(10e-6)	1502	1567	1523	1803	1445
Rand score	0.93	0.946	0.951	0.902	0.945
Information Theoretic score	0.974	0.975	0.976	0.969	0.975

Table 3.4: Compare the results using different gamma pairs for upgrade dice loss: use small gamma for boundary class 0 and big gamma for neuron class 1 give the best performance as we wanted

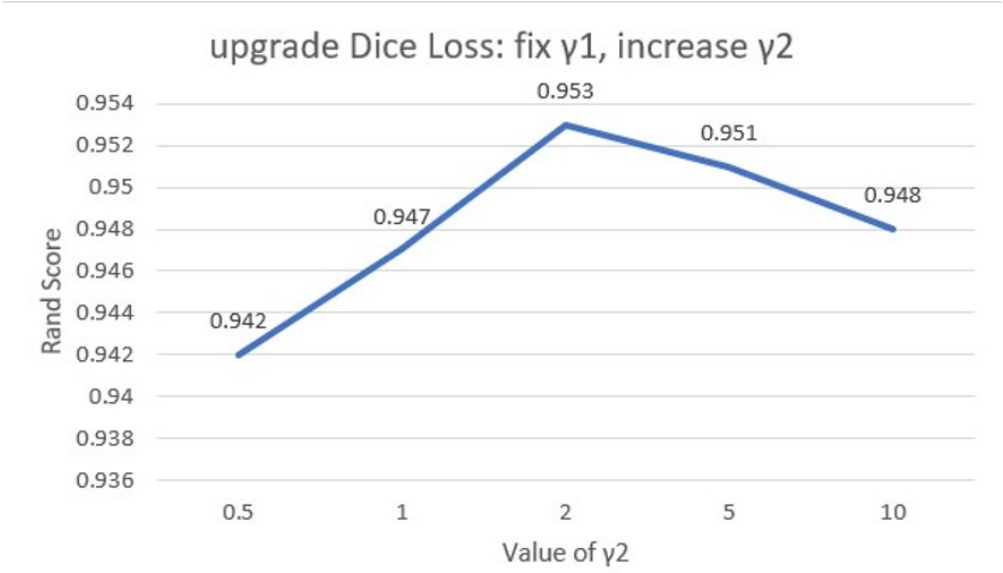


Figure 3.5: Fix the gamma 1 as 0.5, increase the value of gamma 2. Best gamma pair for this task is 0.5;2

4. UNSUPERVISED LEARNING ON CELL IMAGES

4.1 Overview of Unsupervised Method

The researchers in the Weizmann Institute of Science, Israel have recently published a new network structure named "Double-DIP" which uses deep learning to separate the wanted and unwanted parts from one image with no need of any labelled training data [4]. This method, "Double-DIP", is based on "DIP" , Deep Image Prior [32] published earlier at CVPR 2018. DIP aims for image restoration tasks like image denoising, super-resolution, in-painting, etc. It supposes that part of the capability of deep learning resides in the network structure besides the big training data. Specifically, such ability is like having a "prior" of the specific task. For our human, prior can be defined more intuitively as our beliefs if we do not have full information, e.g. In the case of denoising, a prior of a noisy image is what we think the natural version should look like. For the unsupervised method, the prior means that the network tends to learn some low-level statistics of the input image and such low-level statistics are what makes the image natural in human's perspective. Previously, insufficient training data causes overfitting problem and data augmentation is needed to help the network learn more general information. For DIP, since no labelled data is used, the network is not going to overfit to the noisy image easily, it tends to restore to the natural form of the image first.

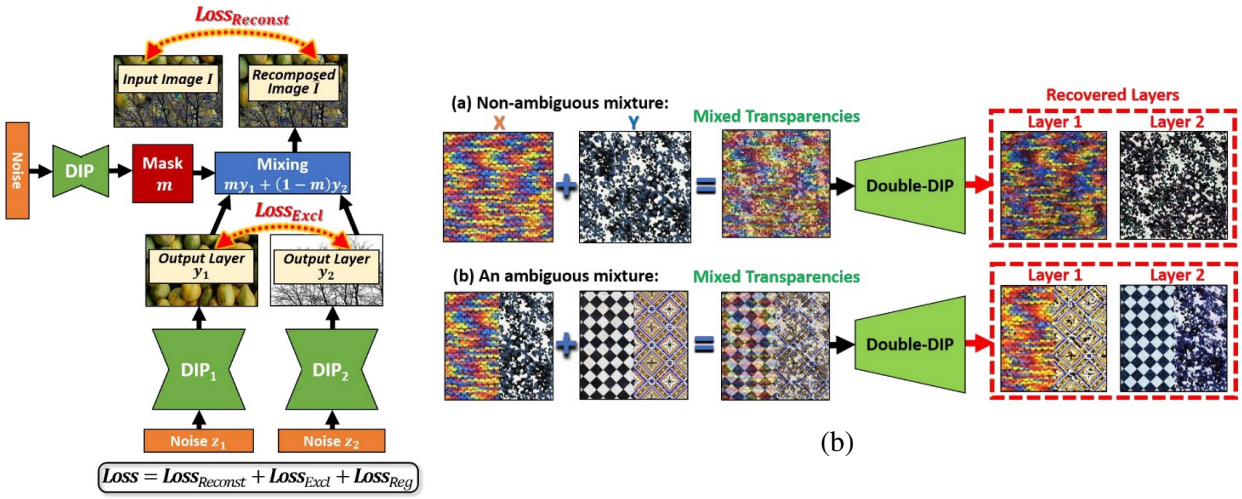
The authors of DIP use random noise as the input and original image as the wanted output to see how well the network can generate the natural form they expect. For example, given a foggy city image as original image, the network is expected to output a translucent image of city but not further overfit to the foggy original image. This paper also compare the optimization process of training the network on natural image, image with noise and random noise. Normally, it is likely to assume that noisy image is the easiest one to be generated back. However, the experiment shows that the loss converges much faster for the natural image. This is a prove of CNNs have this "prior" which is actually a bias towards the natural form of one image. This allows us to use a CNNs as a

decoder for generating natural images given compromised images.

Based on DIP, [4] proposed a layer decomposition framework by using two parallel DIP networks. One outputs the foreground layer and the other outputs background layer, combining these two layers together with a mask to reconstruct an image which should be trained to be similar to the original image. This capability stems from the fact that the internal statistics of a mixture of layers is usually more complex than the statistics of each of its individual components [4], and DIP tends to learn the easier one. So these two networks output relative simple separated layers but not original image containing mixed layers.

But layer decomposition faces a challenge when there are multiple textures mixed in one image, which is common for natural images. In the upper part of image 4.1b, different simple textures can be successfully separated, but in the lower picture, when the layer contains more than one textures, the separation becomes undetermined and causes the layer decomposition results to be ambiguous. Such ambiguities are also within our cell images, especially in the background layer. In the paper, the authors propose to use an initial hint to constrain the network to learn specific texture area in the beginning stage of the training. They used a crude image saliency [33] for the two networks, namely shown in Figure 4.1, DIP1 is restrained to train only on the salient image regions, whereas DIP2 is restrained on the non-salient image regions, for the first few thousands of training iterations.

Our work applys Double DIP method on the cell data, cell regions are foreground layer with value 0, color black, and remaining regions are the background with value 1, color white. We use a simple pipeline to generate the hints and use Double DIP network to train 6000 iterations for each image, hint relaxed after 3000 iterations. Loss of reconstruction and loss of regularization of mask are used in our application. These loss are L1 loss which measures the mean absolute error or two terms. PSNR (peak signal-to-noise ratio) is used during our training process to measure the quality of reconstruction. The signal in this case is the original image, and the noise is the reconstructed output. If PSNR is more than 30, then normally human can perceive the reconstructed image is quite similar to the original image. During the training time, PSNR is calculated between the



(a)

Figure 4.1: (a) Double Deep Image Prior Network Structure, decomposing one image into two layers; (b) The ambiguity within one layer confuses Double DIP network. Image reprinted from [4]

original image and network reconstructed output and when it is over 30, the network will be early stopped.

4.2 Hint generation

Simple and automatic generation of hint for the cell images are designed. The mean value of an image is used as the threshold to separate it into a binary image. It can be seen in the left image of Figure 4.2 that there are lots of background area around cell, so an erosion operation is performed using python skimage package. One sample of our hint is in the right part of the image. Double DIP has the ability to remove the remaining background part, and based on the shrink cell area as hint to restore the real cell part back.

4.3 Results Example and Analysis

An example of MATLAB and Double DIP segmentation in the semantic form is shown in Figure 4.3. There is no miss-classified background area in Double DIP result and the cell shape looks better and detailed. Also the problem of missing cells do not appeared in Double DIP results. Three introduced metrics are used here to give a quantitative comparison. Pixel accuracy means the

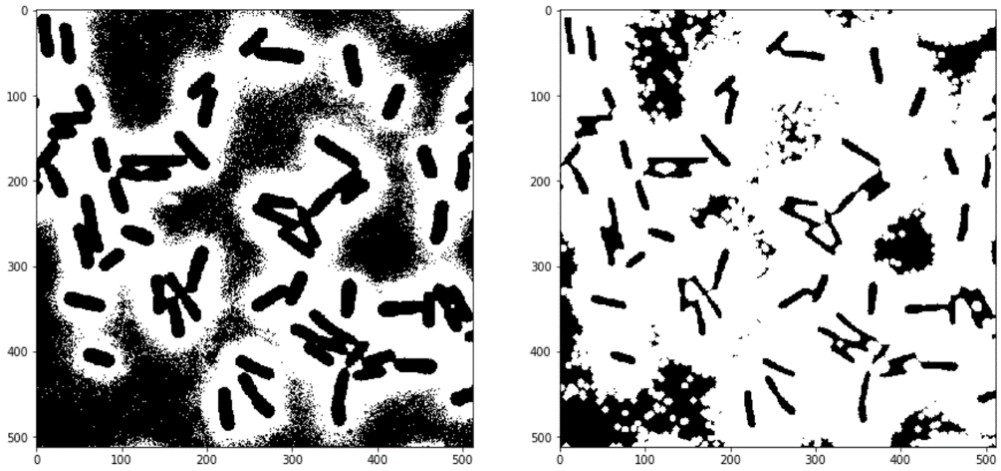


Figure 4.2: Illustrations of cell image hint generation: left is thresholded by the mean value of the original image; right is afterwards erosion

Figure 4.3: Comparison between MATLAB automatic segmentation on the middle and Double DIP segmentation on the right

ratio of correct classified pixel of the whole image, IoU score and Dice coefficient are introduced in the previous chapters. Double DIP outperformed MATLAB results on all three metrics. So as a semantic segmentation tool, Double DIP performs quite well.

Metrics	MATLAB	DOUBLE DIP
Pixel accuracy	0.94	0.97
IoU score	0.76	0.84
Dice Coefficient	0.86	0.91

Table 4.1: Metrics Comparison between MATLAB and Double DIP results

The touching part between adjacent cells make it impossible to separate them using connected component labelling algorithm. Also boundary information cannot be added to the network by modifying the label setting because we do not have label in the unsupervised method. As we discussed with student in Zeng Lab, such results can be read into their MATLAB program and they can do the following manual correction.

5. CONCLUSION AND FUTURE WORKS

Our works are on three different topics: For the supervised learning on cell images, we modified U-Net by using 3 channel boundary and weighted loss function for the specific segmentation task on cell images and designed a read-to-use segmentation program for our collaborator, Zeng Lab. The object-level performance of our designed program is better than the old program they used previously; For the topic of supervised learning on brain images, modifications on structure and data input are made. And multiple loss functions are examined, combining them give us the best test performance which are better than the original U-Net method. We also design an upgrade dice loss function which can give different importance onto different classes; For the third topic, a new unsupervised method is applied on the cell images and achieved better semantic segmentation performance. Unsupervised method could be one of the possible solutions to deal with scarcity of ground truth data in the biomedical field and help enhance the performance of other supervised method.

The possible future direction for cell segmentation is that for more complicated scenario of clustering cells, we may need to upgrade the network by modifying and adding other components into the U-Net framework to have better separation and localization perfomance. For the brain segmentation, there are more complicated challenges in recent years, it could be our future direction to participate in these challenges and learn more about brain image segmentation. For the unsupervised method, it has the potential to be stronger. Currently the Double DIP method only works on relatively simple dataset, it could be a future direction to find a way to add more information into the training process, like teaching it how to segment the boundary of the cells so that we can get instance segmentation results directly.

REFERENCES

[1] I. Arganda-Carreras, S. C. Turaga, D. R. Berger, D. Cireşan, A. Giusti, L. M. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. M. Buhmann, *et al.*, "Crowdsourcing the creation of image segmentation algorithms for connectomics," *Frontiers in neuroanatomy*, vol. 9, p. 142, 2015.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[4] Y. Gandelsman, A. Shocher, and M. Irani, ""' double-dip": Unsupervised image decomposition via coupled deep-image-priors," *arXiv preprint arXiv:1812.00467*, 2018.

[5] Y. Cui, G. Zhang, Z. Liu, Z. Xiong, and J. Hu, "A deep learning algorithm for one-step contour aware nuclei segmentation of histopathological images," *arXiv preprint arXiv:1803.02786*, 2018.

[6] P. H. Li, L. F. Lindsey, M. Januszewski, Z. Zheng, A. S. Bates, I. Taisz, M. Tyka, M. Nichols, F. Li, E. Perlman, *et al.*, "Automated reconstruction of a serial-section em drosophila brain with flood-filling networks and local realignment," *bioRxiv*, p. 605634, 2019.

[7] P. Naylor, M. Laé, F. Reyal, and T. Walter, "Nuclei segmentation in histopathology images using deep neural networks," in *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pp. 933–936, IEEE, 2017.

[8] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[9] P. Filipczuk, M. Kowal, and A. Obuchowicz, "Automatic breast cancer diagnosis based on k-means clustering and adaptive thresholding hybrid segmentation," in *Image processing and communications challenges 3*, pp. 295–302, Springer, 2011.

[10] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," in *ACM transactions on graphics (TOG)*, vol. 23, pp. 309–314, ACM, 2004.

[11] I. Arganda-Carreras, S. C. Turaga, D. R. Berger, D. Cireşan, A. Giusti, L. M. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. M. Buhmann, *et al.*, "Crowdsourcing the creation of image segmentation algorithms for connectomics," *Frontiers in neuroanatomy*, vol. 9, p. 142, 2015.

[12] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical image analysis*, vol. 42, pp. 60–88, 2017.

[13] J. W. Young, J. C. Locke, A. Altinok, N. Rosenfeld, T. Bacarian, P. S. Swain, E. Mjolsness, and M. B. Elowitz, "Measuring single-cell gene expression dynamics in bacteria using fluorescence time-lapse microscopy," *Nature protocols*, vol. 7, no. 1, p. 80, 2012.

[14] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in neural information processing systems*, pp. 2843–2851, 2012.

[15] S. M. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, and M. K. Khan, "Medical image analysis using convolutional neural networks: a review," *Journal of medical systems*, vol. 42, no. 11, p. 226, 2018.

[16] B. S. Lin, K. Michael, S. Kalra, and H. R. Tizhoosh, "Skin lesion segmentation: U-nets versus clustering," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–7, IEEE, 2017.

[17] N. Ibtehaz and M. S. Rahman, "Multiresunet: Rethinking the u-net architecture for multi-modal biomedical image segmentation," *arXiv preprint arXiv:1902.04049*, 2019.

[18] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, pp. 1520–1528, 2015.

[19] Y. Guo, W. Huang, Y. Chen, and S. Tu, "Regularize network skip connections by gating mechanisms for electron microscopy image segmentation," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 868–873, IEEE, 2019.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[21] K. Wu, E. Otoo, and A. Shoshani, "Optimizing connected component labeling algorithms," in *Medical Imaging 2005: Image Processing*, vol. 5747, pp. 1965–1976, International Society for Optics and Photonics, 2005.

[22] "2018 data science bowl 1st place solution," *https://www.kaggle.com/c/data-science-bowl-2018/discussion/54741*, 2018.

[23] J. C. Caicedo, J. Roth, A. Goodman, T. Becker, K. W. Karhohs, M. Broisin, M. Csaba, C. McQuin, S. Singh, F. Theis, *et al.*, "Evaluation of deep learning strategies for nucleus segmentation in fluorescence images," *BioRxiv*, p. 335216, 2019.

[24] T. A. Jarrell, Y. Wang, A. E. Bloniarz, C. A. Brittin, M. Xu, J. N. Thomson, D. G. Albertson, D. H. Hall, and S. W. Emmons, "The connectome of a decision-making neural network," *Science*, vol. 337, no. 6093, pp. 437–444, 2012.

[25] M. Helmstaedter, K. L. Briggman, S. C. Turaga, V. Jain, H. S. Seung, and W. Denk, "Connectomic reconstruction of the inner plexiform layer in the mouse retina," *Nature*, vol. 500, no. 7461, p. 168, 2013.

[26] N. Abraham and N. M. Khan, "A novel focal tversky loss function with improved attention u-net for lesion segmentation," in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pp. 683–687, IEEE, 2019.

[27] J. S. Kim, M. J. Greene, A. Zlateski, K. Lee, M. Richardson, S. C. Turaga, M. Purcaro, M. Balkam, A. Robinson, B. F. Behabadi, *et al.*, "Space–time wiring specificity supports direction selectivity in the retina," *Nature*, vol. 509, no. 7500, p. 331, 2014.

[28] K. Briggman, W. Denk, S. Seung, M. N. Helmstaedter, and S. C. Turaga, "Maximin affinity learning of image segmentation," in *Advances in Neural Information Processing Systems*, pp. 1865–1873, 2009.

[29] M. Meilă, "Comparing clusterings—an information based distance," *Journal of multivariate analysis*, vol. 98, no. 5, pp. 873–895, 2007.

[30] J. Nunez-Iglesias, R. Kennedy, T. Parag, J. Shi, and D. B. Chklovskii, "Machine learning of hierarchical clustering to segment 2d and 3d images," *PloS one*, vol. 8, no. 8, p. e71715, 2013.

[31] A. Fakhry, H. Peng, and S. Ji, "Deep models for brain em image segmentation: novel insights and improved performance," *Bioinformatics*, vol. 32, no. 15, pp. 2352–2358, 2016.

[32] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9446–9454, 2018.

[33] S. Goferman, L. Zelnik-Manor, and A. Tal, "Context-aware saliency detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 10, pp. 1915–1926, 2011.