COMP9517 Computer Vision

Final Report

Prepared by

Han Yang z5140181

Xueyan Lu z5159859

Haixin Chen z5181211

Chengbin Feng z5109259

Tables of Content

1. Introduction

The aim of project is to develop some methods to separate membrane pixels and non-membrane pixels from original dataset. Ground truth label is also given as an instruction to help us assess effects of our methods.

We mainly focus on machine learning based approach SVM plus post-processing, deep learning based approach including U-Net and Deep Residual and also non-learning based method Watershed. The general process includes data augmentation, pre-processing, feature extraction, classification, post-processing and evaluation. We will elaborate these methods and results detailedly in our report. Among all these methods, we find that Deep Residual has best performance.

2. Methods

2.1 Machine Learning based Approach

2.1.1 SVM based Segmentation and Post-processing

Support Vector Machine, SVM, is a popular algorithm in machine learning. Unlike deep learning based approach, this method takes manually selected features as input and try to find the largest margin that can separate pixels into 2 classes (membrane pixel and non-membrane pixel). Therefore, feature selection is a fairly critical task.

We will explain why we choose these features, how SVM works, post-processing method and we will also describe our attempts and experiments on other features or methods before we find the most suitable ones in exploration of methods section. In order to elaborate our method more clearly, we also display a figure to show our flow of method.

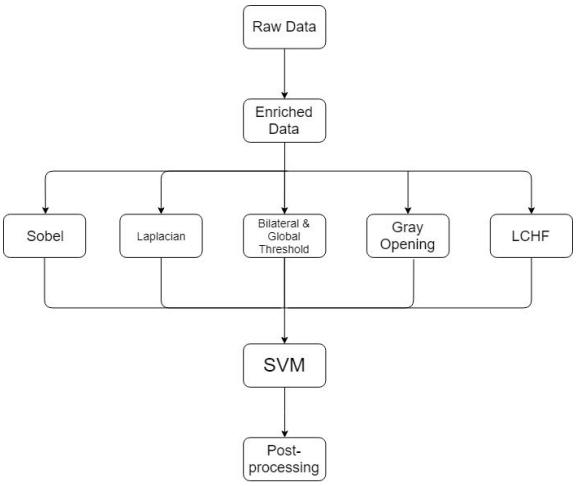

Figure 1: Method Flow of SVM

2.1.1.1 Feature Extraction

For feature selection, we find Sobel, Laplace, global thresholding combined with bilateral filtering, gray opening and LCHF fairly helpful for our task.

Sobel and Laplacian are typical edge detection methods that work with first order derivatives and second order derivatives respectively. In our case, since membrane pixels are most likely edge pixels, Sobel and Laplacian would be great features for SVM to train on.

We also use global thresholding combined with bilateral filtering. The reason why we adopt bilateral filter rather than Gaussian or median is that it can better preserve edge while removing noise. After applying a global threshold, this can help SVM learn which are possible membrane pixels in the first place.

Gray Opening is another feature that focus on smoothening high local maxima. This operation is actually done by a grayscale erosion followed by a grayscale dilation. This feature will help SVM remove sharp noise.

Our last but almost most important feature is local contrast hole filling, short for LCHF, which is a non-learning membrane segmentation method proposed by R. Raju, T Maul and A Bargiela [1] using Matlab. Since implementation on python is a bit different from using Matlab, we tried many parameters so that we can achieve an outcome as good as original algorithm or even better. The algorithm individually includes pre-processing, classification and post-processing.

For the pre-processing part, a median filter is applied in order to remove some noise from image. Initially the filter size is set to (4,4) while we can only set an odd number as filter size to get the median value. After a few trials, we decide the best parameter to be 3 because we find the larger the filter size is, the more possibility that it will blur the margin between dark blocks and membrane, which will affect our later result. Then we apply CLAHE, short for Contrast Limited Adaptive Histogram Equalization, to improve local contrast. Initially the window size is set as 25×25. However, when it comes to our case, clip limit value of 2.0 and moving window size of 15×15 can make membrane pixels much clearer.

For the classification part, a global threshold value 100 is applied on the processed image. At this stage, we got a binary image yet it still contains some dark blocks. Hole Filling is used as post-processing to remove some dark blocks. Unlike instructions in reference, we discard dilation method at this stage since the membrane edges from dilation output is much thinner than actual result. After that, a median filter of size 3 is used to remove noise and finally we got a good result that in most cases is able to separate out membrane pixels but remain some noise and sometimes shallow dark blocks. We think this would be a good beginning for SVM to train.

A sample of our result can be seen as figure 2 below.

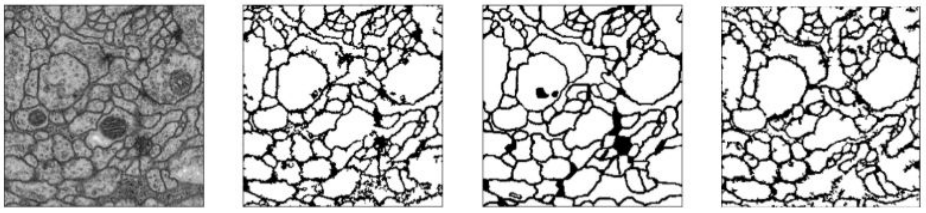

Figure 2: Result of modified LCHF algorithm (second from left) compared to existing ground-truth (second from right) and initial LCHF result from R. Raju, T Maul and A Bargiela[1] (first from right).

2.1.1.2 SVM

In SVM structure, we combine 5 features mentioned above (Sobel, Laplace, LCHF,) to train SVM classifier. For each input image (512×512), we read it in gray-scale and compute the 5 features for each pixel, then we get the feature matrix (262144×6). The paper[3] said we should choose 50000 pixels to train the model. Considering the huge training time, we randomly select 2000 pixels (half membrane pixels and half non-membrane pixels) as 'X' and also select 2000 corresponding label pixels as 'y' to train the classifier. After completing training, the trained classifier will be stored as a .pkl file, then we can load this file as our model to do the prediction for a new input image.

2.1.1.3 Post-processing

Since result from SVM still contains some dark blocks, we need further post-processing to remove them. The idea of method comes from an article on the Internet [2], which aims to detect multiple light bulbs in an image. In our implementation, we first apply explosion method on our image in order to make membrane pixels connected and then we apply dilation method in order to make lines thinner. After that we can find out all the connected regions using a measure.label method from skimage. For most of the connected regions, they are just noise or isolated dark blocks. Only one or two of these connected regions contain most membrane pixels. Therefore, we can set a threshold value to extract the mask with most number of connected pixels. Although this method may not be able to remove those blocks connected with membrane pixels, it is still strong to remove isolated dark blocks.

2.1.1.4 Exploration of Methods

Previously we experiment on different methods based on predecessors' experience but does not achieve a good outcome as we expected. We also put these attempts here to indicate the process of finding a final solution with good results.

For membrane segmentation, since membrane pixels are most likely located on edges in the images, we consider edge detection as a useful tool to separate out membrane pixels. We come up with using Adaptive Threshold Classification using Edge Strength [4] in the first place.

ADTES method in essay [4] is applied after attribute opening function. The attribute opening function will remove the noise of raw images and to combine the similar part which has similar intensity into one region. However attribute opening in python is different from matlab. it more likely to combine the pixels into super pixels, therefore there are gaps in black color between two super pixels.if we apply this function before the edge detection, these gao will be much more obvious than boundary.

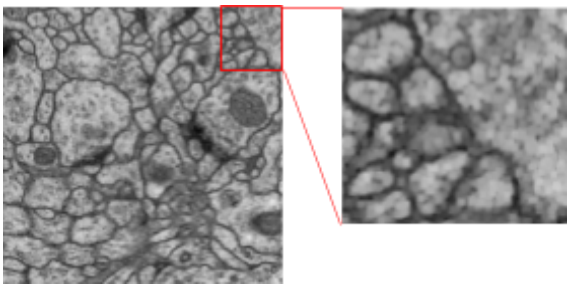

Figure 3: the left image is raw image processed by gray opening function, zoom that as the right image show, there are gaps between super pixels.

To avoid this defect of gray opening in Python, we tried median filter to eliminate the noise. Because ADTES is an edge strength based function, therefore we need to remove the noise but keep the membrane structure clear. Gaussian filter and mean filter will blur the membrane boundary by their neighbour pixels. Compare with that median filter can give a more clear membrane boundary to detect, this filter can keep the intensity value of raw image.

We write ADTES function in python with same as the paper [3] by using a different adaptive thresholding function and sobel image, and set the Wm to 33, but the result is not good as the essay show. As the images below show, the sobel image cannot give us a perfect edge of membrane structure. and the image looks totally black that the mean intensity will be too low, but intensity of some part of membrane structure is even greater than local or global mean value. In contrast, gradient image give us a better result, that the mean intensity is brighter, however, not large enough, hence when applying ADTES with gradient the middle part of membrane is even white. The result is not applicable to be a feature in SVM.

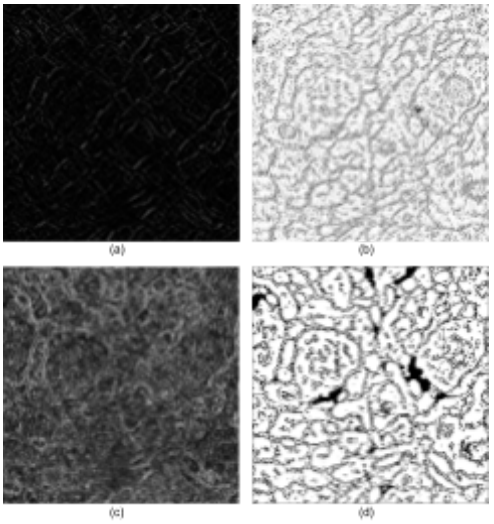

Figure 4: (a) image is the sobel image of raw image. (b) image is apply ADTES use sobel detection. (c) image is gradient image of raw image. (d) image is apply ADTES use gradient edge detection.

Nucleus is also a main obstacle for membrane segmentation since some of them are so large that common filtering method cannot remove dark blocks and edge detection method may recognize pixels on edge of nucleus as membrane pixels. Before we find out the final method, we experiment on different methods. One example is an in-built function SimpleBlobDetector from OpenCV, however it is not easy to use. When we try different parameters, we find it hard to include all dark blocks yet exclude some of membrane pixels at the same time since some membrane pixels also looks like dark blocks. To be specific, in order to map more nucleus with elongated shape, we set minimal Convexity and minimal Inertia Ratio to a slightly lower value, however those membrane pixels are also recognized as nucleus pixels as shown in figure below. In this case, SimpleBlobDetector might not be able to separate out dark blocks with irregular shapes.

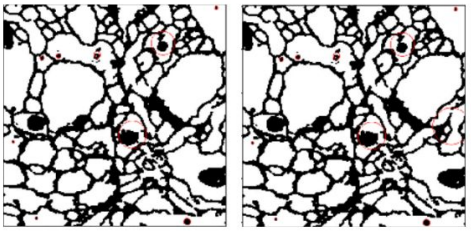

Figure 5: An example of a slightly lower threshold value(second from left) might mistakenly recognize membrane pixels as nucleus.

2.2 Deep Learning based Approach

2.2.1 U-Net

U-Net perform well on medical image segmentation. Most medical images, including the images in our project have several attributes, are 1) simple semantics, 2) have a settled structure. The structure is simple that the skip connection and U shape structure will be useful for this project. Both of downsample and upsample in U-Net execute for 4 times, use skip connection at same stage to copy and crop. This operation can guarantee that the result fuse low-level feature, and at same time fuse the feature in different scale. 4 times upsampling also give a more precise boundary information.

2.2.1.1 Understanding about the Network

U-Net, is a Fully-Convolutional-Network model, using skip connection to extract the abstractive features. It has good performance in solving visual recognition problems, so we choose it as our deep learning based method. The whole structure contains two main phases, one is construction phase, a traditional flow of some convolutions and down-sampling(max pooling), which is used to extract different levels of features of input image. The other one is the symmetric expensive phase, combines features from construction phase and the results of deconvolution to reconstruct the original image from these features, it generally involves up sampling from low resolution to high resolution.

2.2.1.2 Flow Descriptions

We implemented two versions of U-Net network with different parameters. The first version (unpadded) is strictly based on the paper [4], there are 5 kinds of operations through the whole network. 1) overlap-tile strategy. It is used to preprocess the initial input image with padding 30 pixels in each border (512×512 -> 572×572), these pixels are extrapolated by mirroring. 2) convolution operation. For example, in the first convolution layer, we have the input image of size 572×572×1, 64 filters of size 3×3×1 (1 is the channel of gray image), padding is 0 and stride is 1, and the active function is ReLU. Hence the output is 570×570×64, then in the second convolution layer, we do the convolution with same parameters except the size of filter has been changed to 3×3×64. 3) max pooling. We use 2×2 block to select the maximum pixel value as the value of pooled feature map, which reduces the sensitivity of small changes in the location of features. In addition, it reduces the size of border by half, which can also promote the computing efficiency significantly and can help to focus on higher level features on the following convolution layers. 4) up-convolution. We combine up-sampling and convolution to do the deconvolution, in this step, the size of image will be enlarged and the resolution can be recovered to a higher level. 5) copy and crop. Following each two

convolutions, we do the centerCrop and concatenate with corresponding step in expansive phase to ensure more previourse features can be merged, which can recover more information precisely such as edges.

The second version (padded) is similar to the first one, we just redesign the value parameter of padding and do not need to do centerCrop to do the concatenation, since after padding, the size of image will not be changed anymore. Hence, the input and output images are in the same size 512*512. The final result of U-Net is based on the second version, because the output image is in size 388*388, which is not suitable to do evaluation with label image.

2.2.2 Deep Residual Network

Compared with U-Net, Deep residual network is obviously deeper. According to He, Kaiming and Jian Sun [5], deeper network with smaller filter size would have better performance than shallower networks with larger filter size in the context of same time complexity. However, in another paper from He et al. [6], simply stacking more layers might lead to degradation problem that is rapidly dropping of training accuracy. Deep residual network is proposed to not only address this problem but also show higher accuracy. In our study, we would like to experiment on this network and expect to see a better result.

2.2.2.1 Understanding about the Network

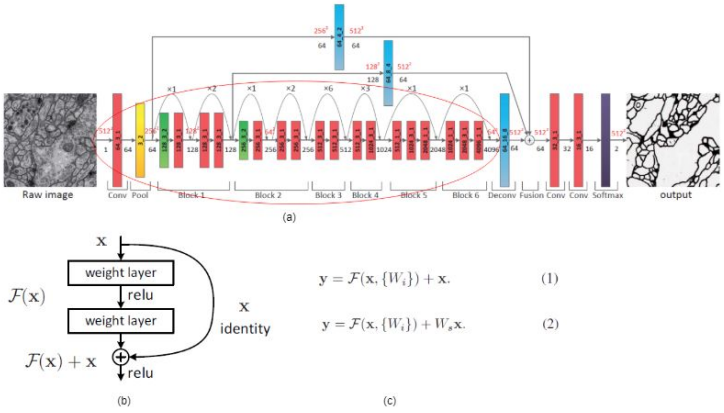

Figure 6: (a) A complete structure of a deep residual network from [7] where the part encircled by red lines is the contracting path with similar structure from resnet38. (b) The inner structure of a residual unit from [6]. (c) The computation equation of a residual unit from [6].

Based on [7], deep residual also contain a contracting path and an expansive path that is similar to U-Net. However the structure of contracting path is based on resnet38, so later we researched on [8] and it shows each block denotes a different residual unit with inner shortcut connections. If we do not apply on shortcut connections, then our network would no longer be a residual network. He et al. [6,9] showed an instruction on how shortcut connections work in their paper and the basic structure and computation of a residual unit is also shown in figure below. Every block in Figure 6(a) adopts a residual structure shown in (b). To be specific, suppose input and output of a residual unit are x and y respectively, every layer except the last layer exploits a relu activation function. After making an addition on identity, another relu activation is applied. Sometimes when there is variations on depth of

convolution layer or output size, a linear projection, denoted Ws, is needed to apply on input of unit. This operation can be summarized in a equation shown as figure 6(c).

2.2.2.2 Flow Descriptions

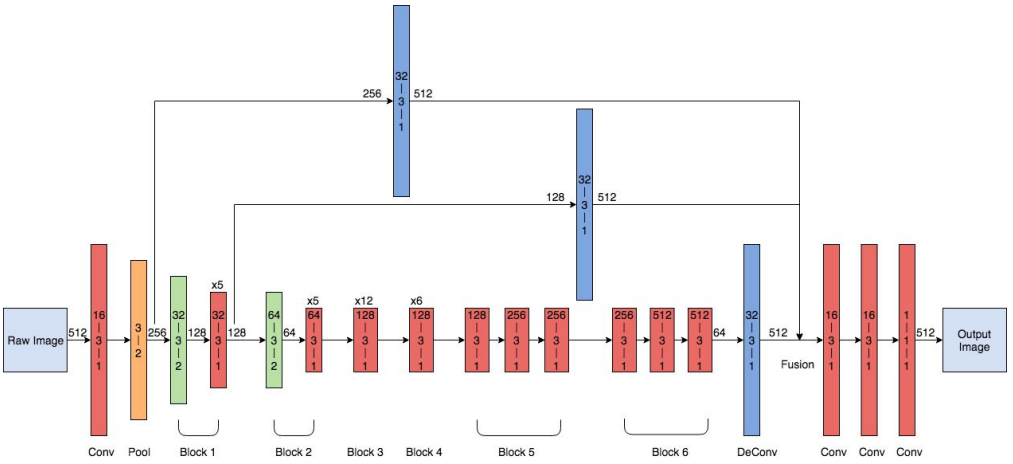

Figure 7

Our structure is inspired from paper [7] but adjust the value of parameters in convolution layers. The overview of our structure shown as Figure 7, which contains two main phases like U-Net. One is traditional convolution phases, deeper than U-Net, the another is deconvolution phase. In the first convolution layer, we use 16 filters of size 3×3×1 and 1 stride to convolve the 512×512 input image with same padding. After that we implement a max-pooling operation with 3×3 block and 2 strides, which reduces the size of each border by half. Following are two blocks of several convolution layers with 2 strides, which are used to do the downsampling. Then is the successive convolution layers with padding, it is deeper than U-Net and can extract more complex features. Next is deconvolution phase, we also use up-sampling to resize the previous feature maps from 64×64, 128×128 and 256×256 into 512×512 maps separately, and fuse them together to get the feature map of initial image size, so-called summation-based skip connections [7], different with U-Net. Finally, two convolution layers and dropout with p is 0.5 are implemented to avoid overfitting.

2.3 Watershed based Approach

2.3.1 The quality and method description

In this study, we have tried some traditional approaches in this image analysis project. In order to extract the information necessary for solving an image problem, three main steps consisting of preprocessing, data reduction and feature analysis should be implemented in the traditional approaches. As the knowledge told in the lecture, preprocessing removes noise and eliminates irrelevant information. Data reduction extracts features for the analysis process while the extracted features are examined and evaluated for their use in the application.

During the taste of this project, we have implemented only one threshold to process the image.

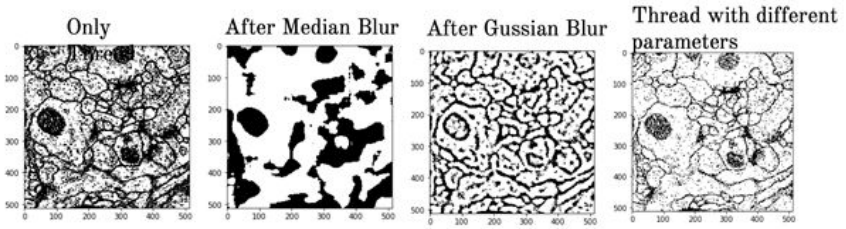

Figure 8

However, the result can be easily told from easily it is so bad. This is because it is very hard to extract each individual coin from the image by using tradition image processing methods such as thresholding and contour detection.

In order to improve the accuracy of the result, therefore, we have combined filter, threshold and watershed together to process the image in our tradition approach.

Watershed is an efficient segmentation to detect an extract objects in images that are touching and/or overlapping.[10] Due to many cells in the images are overlapping and some cells appear in more than one section. Therefore, combing filter, threshold and watershed is set as our one of choice to deal with this project.

2.3.2 Appropriate level of method complexity

Since the traditional method have not implemented learning algorithms, the calculation including space and time complexity of the method is much lower than deep learning approach. The complexity part for this method is how you understand the filters and threshold. And finding a good threshold value is a big challenge.

2.3.3 Correct logic of method design and reasonable design choices

About choosing filter to reduce the noise, from the definition of median filter and Gaussian filter from opencv docs, we can know that median filter run through each element of the signal and replace each pixel with the median of its neighboring pixels while Gaussian filter is done by convolving each point in the input array with a Gaussian kernel and then summing them all to produce the output array. Since the median filtering can preserves the edges while removing noise, we assume the median filtering would be a better option in this method.

Talking about the threshold, the simple thresholding is to assigned one value if the pixel value is greater than a threshold value. However, this value is very hard to find. And sometime, it may not be good in all the conditions where the image has different lighting conditions in different areas. In this case, adaptive threshold works better since the algorithm can calculate the threshold for a small region of the image. Instead of using an arbitrary value for threshold value, otsu binarization can calculate a threshold value from image histogram which seems a better choice.

Figure 9

Last step, a marker-based watershed algorithm will be implemented in the method. The algorithm starts to fill every isolated (local minima) valley with different colored water. With the rising of water, water in different valleys begin to merge. And in the same time, barriers will be built until all the peaks are under water which can return the final result. The maker-based watershed algorithm can specific where to merge and where not. And we can label the region which we are sure of being the foreground or foreground. This is very helpful in improve intra-section neuron segmentation accuracy. In summary of this method, the filter can reduce the noise. The threshold can give us a better result after convert the images with varying illumination in different regions. And watershed can successfully segment overlapping cells by labelling every region.

3. **Experimental setup**

3.1 Data Augmentation

Data augmentation is important in learning based approach especially when dataset is not large enough. Training on a limited dataset will limit generalization of classifier because the classifier may overfit on existed data. Even classifier has a good performance on existing data, the accuracy would decrease when it comes to testing on unknown data.

In our case, initial training dataset contains only 30 images with size 512×512, which means we do need data enrichment before we start our learning based method. In order to avoid overfitting, we apply elastic transformation with different parameters to enlarge our dataset. After augmentation, the number of samples in our training dataset has increased to 300. An example of our augmentation result can be seen in figure 10 below.

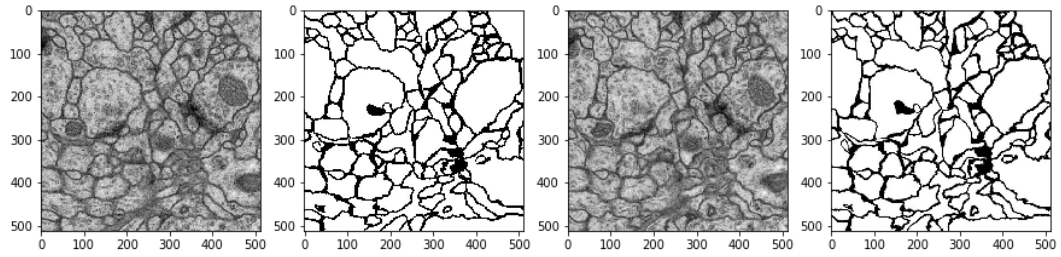

Figure 10: An example of data enchiment by exploiting elastic transformation. First and second from left is initial image and label. Third and fourth from left is a variation of initial image and label using elastic transformation.

3.2 Evaluation Methods

3.2.1 Cross Validation

Typically, we would split our dataset into training folder and testing folder. However, when dataset is in limited size, the effect of classifier would be determined by how we split the dataset. Cross validation is an important evaluation method to help us assess whether our learning based method is overfitting or not.

As for our machine learning based method, we ues 120 images from our augmented dataset combined with the initial 30 raw images as the whole dataset. Before we split our dataset into 5 folds,

we also shuffle our data. For each fold, we select 120 images from dataset as train dataset and always use the 30 raw image as test dataset. Finally, after all the training is done, we use the trained classifiers to test on our initial raw dataset.

As for our deeping learning based method, including U-Net and Deep Residual, since our dataset is enlarged to 300 images, we firstly divide our dataset into 10 folds and each fold contains 3 initial raw images and corresponding labels. Then for each fold, we use the other 27 raw images and corresponding augmented images, which is 270 images in total, to do the training. After training for each fold, we use the initial excluded raw images to the test the corresponding classifier.

3.2.2 Evaluation Metrics

We mainly use two kinds of evaluation metrics that are validation accuracy and loss, V^{Rand} and V^{Info}. Validation accuracy and loss is an in-built evaluation in learning based method which gives us a first insight of how our learning based method develops. Accuracy is to compute the number of pixels segmented correctly divided by all pixels. For loss function we used binary cross entropy which is usually used in cases with binary classes.

V^{Rand} and V^{Info} are required in our task. According to ImageJ [11], V^{Rand} and V^{Info} are better metrics to evaluate the performance of our membrane segmentation because previous scoring system in ISBI-2012 is not robust enough to deal with variations in the widths of neurite borders. This metric is to thin the borders of membrane to a width of one pixel and then calculate foreground-restricted rand scoring and information theoretic scoring.

In our case, we would compare all 30 predicted images with corresponding ground truth labels one by one. Then we would calculate the average value of V^{Rand} and V^{Info} for each method in order to better evaluate quality of our methods.

4. Results

4.1 SVM plus post-processing

Table 1: Result of SVM

	Accuracy	Precision	Recall	V^{Rand}	V^{Info}	Run Time
SVM	0.63	0.65	0.78	0.43	0.60	83min/fold

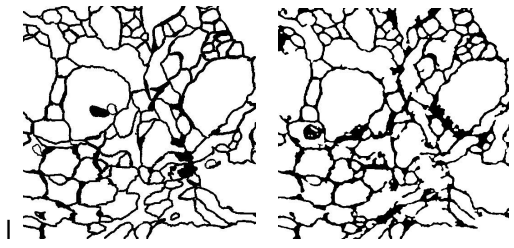

Figure 11: Left is label image. Right is predicted image.

The results of SVM are shown as the table above, the accuracy, precision, recall, V^{Rand} and V^{Info} are all not high, as you can see from the predict image, there still are many noises along membrane, and

it also has mis-labeled pixels on dark blobs. One reason of that maybe the features we extract are not representative enough, which is where we need to improve in the future, maybe we should do deeper researches on the structure of membrane and non-membrane to find more of their unique characteristics as features to train the SVM model. Another reason maybe the training samples are not large enough, since we take into account that training process is too time-consuming, we just choose 30 raw images and 120 augmented images as our dataset. For each fold of cross-validation, the size of training dataset is 120 (randomly selected in 150 images), it is big enough to train for SVM model, because it takes almost 1.5 hours to do training for only one fold, which is inefficient.

4.2 U-Net

Our U-Net implementation runs on Colaboratory with GPU. In order to avoid overfitting, we use data augmentation techniques to enrich our dataset to 300 images and do cross validation by 10 folds. After applying different parameters on our code, we got a series of results with different evaluation metrics. Both accuracy and loss are done by an in-built evaluation function of Keras model with pre-separated testing dataset after the whole training process for each fold. V^{Rand} and V^{Info} are done offline using FIJI after generating all 30 predicted images from Colab. After we evaluate these results, we take the average value to present an overall result in the following table.

Table 2: A comparison on different parameters on U-NET

initial learning rate: 1e-4	Batchsize = 1	Batchsize = 2	Batchsize = 4	Batchsize = 5
	Epoch = 240	Epoch = 240	Epoch = 240	Epoch = 240
Accuracy	0.87498	0.89675	0.9230	0.90693
Loss	0.32126	0.28979	0.2050	0.26767
V^{Rand}	0.70368	0.80694	0.89425	0.85397
V^{Info}	0.85530	0.84601	0.90812	0.85755
Run Time	5min/fold	16min/fold	48min/fold	80min/fold

Based on previous experience, we set our initial learning rate as 1e-4 and epoch as 240 and we adopt a callback mechanism that can reduce initial learning rate by after having no better performance on train loss value after some epochs. Since Colab GPU has limited memory, our code collapse when batch size is set over 5.

As we can see in the table, the performance is obviously not good enough when batch size is smaller. We think the reason might be number of training samples is not sufficient enough. However, it takes more time to run when batch size is larger. Under comparison on performance, we take batch size of 4 as our choice. In order to achieve better result, we also apply a further post-processing on predicted image.

The basic idea of post-processing is simply applying a bilateral filter and then a global threshold function. This can be useful to remove some stubborn but shallow blocks. Although the predicted

result from U-Net is quite similar to ground truth, however, it does not have a good result on evaluation section using FIJI even when we try on different format of label image like TIFF or JPG. We think it might be due to the mechanism of V^{Rand} and V^{Info} since it would thin the edges into 1-pixel width. This is why we further apply a post-processing in U-Net.

Here is also a comparison on predicted image and ground truth when batch size is 4.

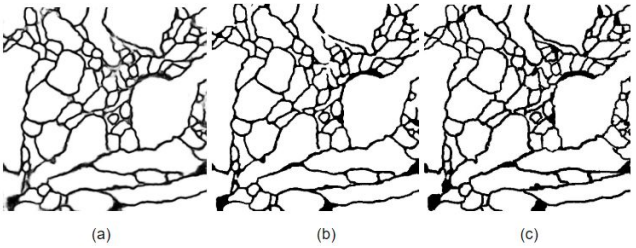

Figure 12: A comparison between output and ground truth for U-Net. (a) Direct output from U-Net model. (b) Image after applying post-processing on (a). (c) Ground truth image.

4.3 Deep Residual Network

We use colab to do this experiment, and run our code by GPU. We use data augmentation to add more images and labels to train our network, total 300 images. We found that the accuracy of result is increased with the growing of batchsize. We split 10% of them to do the validation test. For cross validation we split the raw images into 10 folders. and take rest of the data as train data of the network. We adjust batch size to find the optimal parameter. We use the evaluation function in Keras to test the loss and accuracy in cross validation, and use Fiji to test the V^{Rand} and V^{Info} value.

We use the mean value of our cross validation for each metrics. The result is show as below.

Table 3: A comparison on different parameters on Deep Residual

	Batchsize = 1	Batchsize = 2	Batchsize = 3	Batchsize = 4
	Epoch = 80	Epoch = 80	Epoch = 80	Epoch = 80
Accuracy	0.9174	0.9228	0.9498	0.9653
Loss	0.2134	0.1979	0.1168	0.0826
V^{Rand}	0.7280	0.7230	0.8060	0.8295
V^{Info}	0.9031	0.9085	0.9229	0.9631
Run Time	120min/fold	113min/fold	120min/fold	129min/fold

From the table above shows that the accuracy increasing with the growth of batch size. On colab, our code cannot run with the batch size greater than 4. When the batch size comes to 4, we get the largest accuracy. The running time also increase slightly with the growth of batch size. We think the accuracy increase because when the batch size becomes larger, the total training data volume will be larger, that can promote the accuracy efficiently.

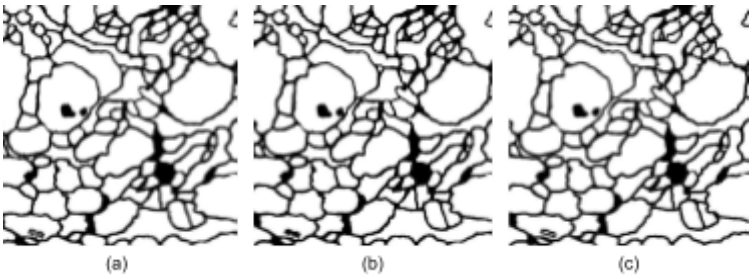

Figure 13: A comparison between output and ground truth for deep contextual residual network. (a) Predict result by deep contextual residual network. (b) Post-processed image of (a). (c) The ground truth

The predict result be processed by same post-processing function as discussed above in U-Net. The predict result is quite clear, however, there are still have some area in gray color, hence we need to classify it whether an membrane part. the accuracy also be enhanced by post-processing.

4.4 Watershed

After applying the above methods, at the beginning the output images are so dark. In order to get a better light of each images, we apply one more threshold after watershed. And V^{Rand} and V^{Info} is like following.

Table 4

Vrand	Vinfo
0.3821	0.3247

The method can label each region of the cell but the noise within each cell is still visible. Although we have tried many parameters and threshold methods, the results improve little, but it is still far away from the results generated by deep learning.

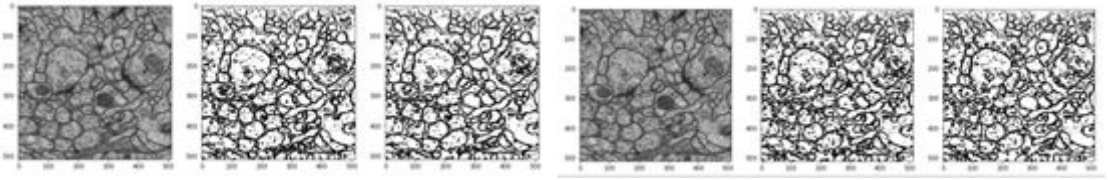

Figure 14:OTSU threshold and Gaussian adaptive Threshold (first from left) Simple Threshold and Gaussian adaptive Threshold(Second from left)

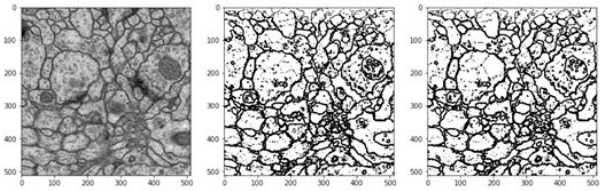

Figure 15: Original image(First from left) Median filter(second from left) Gaussian filter(first from right)

Although Otsu binarization can calculate a threshold value from image histogram which seems a better choice, we still find the simple threshold can generate a better result.

Look the third image, we comparing the results generated by median filter and gaussian filter. Image generated with gaussian filter has less noise.

After trying several combinations of traditional methods in watershed approach, we find Gaussian filter works better than blur filter in removing the noise. This might because the Gaussian filter alone will blur edges and reduce contrast.

By comparing different threshold in the above steps, we choose simple threshold first with 127 as a second input parameter and Gaussian adaptive threshold as the last steps to output the results. Since adaptive threshold can calculate the threshold for a small region of the image which can be good to segment cell which is usually overlapping. And since the image has varying illumination, the result do can improve a lot which can tell from eyes. Gaussian adaptive threshold calculates the weighted sum of neighborhood values where weights are a gaussian window which perform better than mean adaptive threshold whose threshold value is the mean of neighborhood area. This is because the images have varying illumination.

Finding a good parameter in this method takes time, we have tried many times in putting parameters manually to get a better result. And the combination of different thresholds and watershed method also have been tried many times in getting a better result. Fortunately, without learning approach, the running time of this method does not cost too much.

5. Discussion

In this part, we will first elaborate advantages and disadvantages of each method and comparison on different methods. Then we will present a final solution to our whole project.

The good thing for Watershed is the calculation complexity is very little for this method which cost less than 10 minutes for completing the entire iteration. While the noise cannot be fully removed from the output label and finding a good threshold value seems a challenge.

Since there is no uniform criterion for image segmentation, traditional image segmentation methods can only be applied to specific applications. However, SVM based method can utilise various features of the target images to classify the pixels of image precisely, which is a more general approach for image segmentation. It is critical to find good representative features to help SVM learn which pixels are membrane and which are not, otherwise, the performance would be so poor.

The main disadvantage of SVM is the inefficient training process, especially when you have massive observation samples, SVM is not the best choice for image processing. In addition, its segmentation performance is not very good compared with deep learning based methods, the reason for that may be the depth and representativeness of features we find are far inferior to the features extracted by multiple convolution layers of deep learning.

As biomedical image segmentation usually do not have a large dataset, the underlying features are fairly important. Unlike machine learning based methods, deep learning approach extracts features automatically. For example, information on edges is important for membrane segmentation, but when it comes to other cases, color or contour might be the most important. Then SVM needs to redesign its features while deep learning methods have no worries about this. Nevertheless, there are still problems with deep learning methods like overfitting and vanishing gradient.

U-Net is initially designed for biomedical image segmentation. The wonderful parts are downsampling, upsampling and skip connection. While downsampling avoids overfitting, upsampling can recover information loss in previous steps. Skip connection can avoid vanishing gradient problem.
However, the initial U-Net structure proposed by [4] is using overlap-tile strategy. An input size of 572×572 after extrapolating the image by mirror can only get an output size of 388×388. Moreover, the predicted image from U-Net classifier does not map the initial ground truth even after applying center crop, which might decrease universality. A comparison between results coming from U-Net and cropped ground truth image can be seen in figure 16 below. Therefore, many people are using same padding in the whole network in order to get a same size of output.

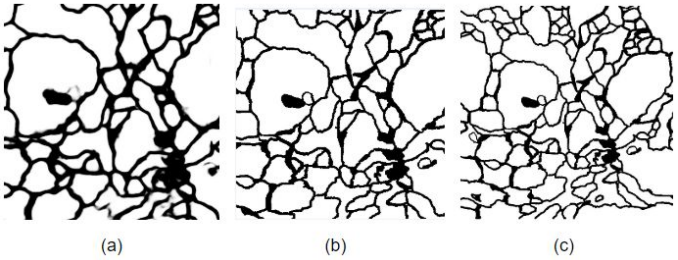

Figure 16: A comparison between predicted image from U-Net and cropped ground truth image. (a) Image predicted by U-Net with size 388×388. (b) Ground truth image with size 388×388 after applying center crop. (c) Initial ground truth image.

Normally, the accuracy of deep network cannot be high, because of vanishing gradient problem. However deep residual network can avoid this problem by apply the shortcut for each residual unit. Original version of deep residual network has several characteristics, first, the number of parameters for whole network is controlled. Second, it has obvious hierarchy with much more convolution layer than U-Net, that the number of feature images increasing with the the layer level deeper, which can guarantee the output feature can carry information as much as possible. Third, it use convolution layer rather than max pooling layer for downsampling to extract the feature, this can enhance the transmission efficiency.

The deep contextual residual network not only have these characteristics, but also add two skip-connection layer to incorporate the global information from higher layer and local cues from lower layer, that can optimize the result to a higher accuracy. Furthermore, in deep contextual residual network, ReLU is replaced by ELU, these two are similar, when x is greater than 0, both of them will be the same linear function can mitigate the vanishing gradient, but when x is negative ReLU function is 0 which means this part is not activated, in contrast ELU can give negative output which means it have ability to wake the noise. Thanks for this trait, to reach same accuracy, our Residual network with ReLU have to do 2000 steps per epoch and cost 8 hours, but with ELU just need around 300 steps per epoch and only takes 1 hour.

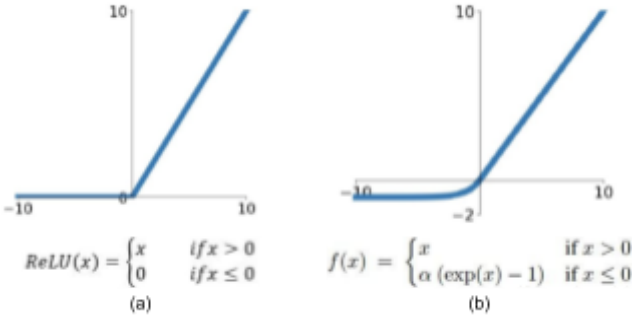

figure 17: Activation function comparison
(a) is ReLU activation function. (b) is eLU activation function

Compare with Machine Learning method and traditional method, both of two types of deep learning segmentation method perform better. Deep learning do not need do the pre-processing with the training data. The network can use convolution layers to extract the feature by itself. The accuracy of deep learning segmentations are much better than Machine Learning based segmentation and traditional method.

U-Net and deep contextual network are both deep learning segmentation, both of them have pros and cons. For these two method, from the result of our experiment on colab, the best batch size of these two is 4. With the same batch size, U-Net is the winner on training time, just takes 48 minutes to training the dataset for a folder, but the deep contextual network takes around 2 hours to train the network. However the accuracy of deep contextual residual network is better. Even we set the batch size to 1, the accuracy, is better than U-Net.

6. Conclusions

In our study, we know learning-based methods have better performance than traditional methods and deep residual network is the best one among our 4 approaches. Even if we have already got well performances in image segmentation, there is still insufficiency in our project. As further improvement, we might need to try more augmentation methods and train our learning based methods with larger training dataset for U-Net and Deep Residual. To improve the performance of SVM, we need to focus more on the principle of image structure fields to have a deeper understanding on the characteristics of different class pixels to extract more representative features to train our model. What's more, maybe we can apply various kernel functions in the future to learn which one can achieve better performance on classifying membrane and non-membrane pixels.

References

1. R. Raju, T Maul and A Bargiela, “Local Contrast Hole Filling Algorithm”,IEEE Symposium on Computer Applications & Industrial Electronics, 2014.
2. Rosebrock, A. (2016). Detecting multiple bright spots in an image with Python and OpenCV - PyImageSearch. [online] PyImageSearch. Available at: https://www.pyimagesearch.com/2016/10/31/detecting-multiple-bright-spots-in-an-image-with-python-and-opencv/ [Accessed 4 Aug. 2019].
3. Xiao Tan and Changming Sun. Membrane extraction using two-step classification and post-processing.InProc. of ISBI 2012 EM Segmentation Challenge.
4. O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” inProc. Med. ImageComput. Comput.-Assisted Intervention, 2015, pp. 234–241.
5. He and J. Sun. Convolutional neural networks at constrained timecost. InCVPR, 2015.
6. He, Kaiming, et al. "Deep residual learning for image recognition." arXiv preprint arXiv:1512.03385 (2015).
7. C. Xiao, J. Liu, X. Chen, H. Han, C. Shu, and Q. Xie, “Deep contextual residual network for electron microscopy image segmentation in connectomics,” in Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on. IEEE, 2018, pp. 378–381.
8. Z. Wu, C. Shen, and A. van den Hengel.Wider ordeeper: Revisiting the resnet model for visual recognition.arXiv:1611.10080, 2016.
9. K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deepresidual networks,”arXiv:1603.05027, 2016.
10. T. Liu, M. Seyedhosseini, M. Ellisman, and T. Tasdizen, “Watershed merge forest classification for electron microscopy imagestack segmentation,” in International Conference on Computer Vision,vol. 2013, 2013, p. 4069.
11. ImageJ. (2019). Segmentation evaluation after border thinning - Script. [online] Available at: https://imagej.net/Segmentation_evaluation_after_border_thinning_-_Script [Accessed 9 Aug. 2019].

Appendix

Student Name	Student zID	Methods	Result
Han Yang	z5140181	Two Steps,U-Net,Residual	
Xueyan Lu	z5159859	Two Steps,U-Net,Residual	
Haixin Chen	z5181211	Two Steps,U-Net,Residual	
Chengbin Feng	z5109259	Watershed	