Final Project Report

Rui Hu
School of Electrical and Computer Engineering

Introduction

In this project, we are going to use 4 different machine learning approaches to analyze and make inferences on the electron microscopy images of the neuronal structures of Drosophila. The data sets are provided by ISBI Challenge: Segmentation of Neuronal structures of EM stacks[1]. In this project, we use machine learning approaches including SVM, Random Forest, SegNet and U-net to train our model, and then use the best trained model to make inferences on the test set. The best model we obtained is trained by U-net and the Foreground-restricted Rand Scoring after border thinning of our inference results is 0.83, and the Maximal foreground-restricted information theoretic score after thinning: 0.92.
We first introduce the dataset, then we describe the details of the project design. Finally, we use 20 figures in the experiments to show our discoveries and results.

Dataset

The data sets are electron microscopy (EM) stacks of Drosophila first instar larva ventral nerve cord (VNC), containing some noises and small image alignment errors. Each object in the images is labeled manually by the experts. The basic information of the dataset is given in Tabel 1.

Tabel 1. Basic information of the dataset.

	Training set		Testing set
	Training images	Training labels	Testing images
Num of image	30	30	30
Size of image	512x512	512x512	512x512

In the following, we show the original image and the label and provide a simple exploration on them.

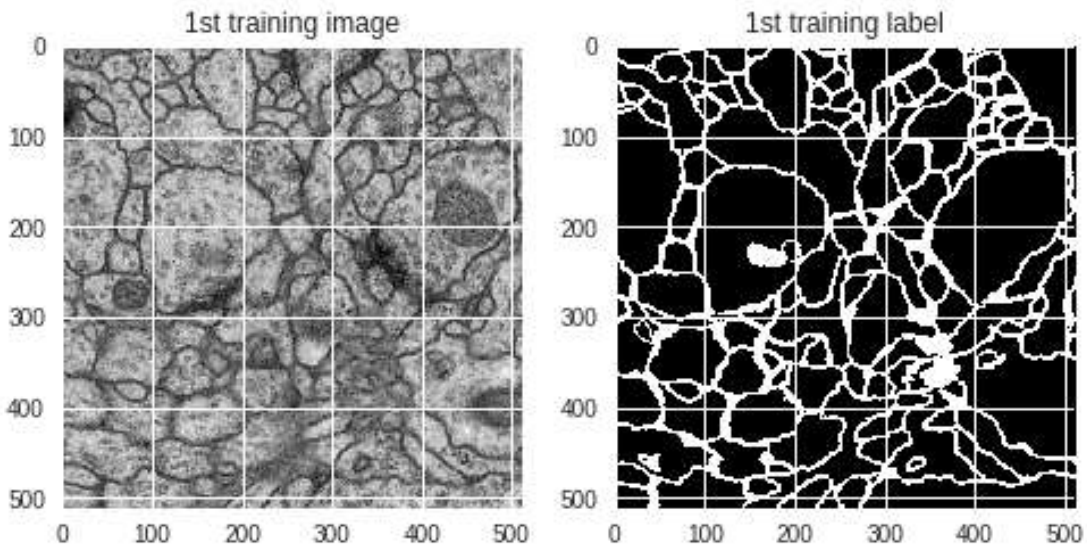

Figure 1. The 1st training data and the corresponding groundtruth(white for the pixels of segmented objects and black for the rest of pixels which correspond mostly to membranes.).

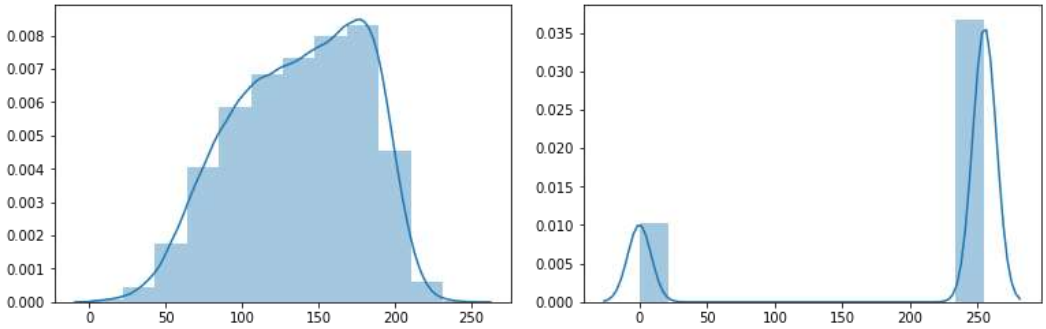

Figure 2. Histogram for pixel matrix of the 1st training image and training label.

We can see that the train images are quite noisy.

Project Design

To find the best model, we are going to try 4 different machine learning approaches, including SVM, Random Forest, U-net and SegNet. For the experiment of each approach, we split the whole process into three phases: Trainng, Testing and Inference. We use the original train dataset for training and testing, and use the test set for inference.

For SVM and Random Forest, we first extract the features of images to generate the training data, and then use them to fit and test the models. In the inference, we extract the features with respect to each pixel of the test images and then use our trained model to make predictions for each pixel.

U-net and SegNet are two popular Deep Nerual Network (DNN) approaches in image segmentation. The convolution layers are embeded in these two models, so the beforehand feature extraction is unneccessary and both the inputs and outputs of these two models are images. Since the original images is too large for us to train the model (due to the limited computation capacity), we extract small patches from the original images and labels, and use these patches to fit the model. In the

inference, we also extract the patches from the input images, but in the end we need to construct the ouputs of inference into a complete groundtruth for the test set.

The evaluation metrics for the inference results provided by the ISBI Challenge include the Foreground-restricted Rand Scoring after border thinning and Foreground-restricted Information Theoretic Scoring after border thinning. The details of these two metrics are available on the website of this ISBI Chanllenge.

SVM and Random Forest

Training

Feature Extraction

We first extract features from the original image stacks. Based on the existing feature extraction approaches, we mainly consider three types of features: Spectral (RGB) values, Texture feature and Haralick texture features. Since our data are grayscale images, the 3-dimensional spectral of each pixel will have the same value. The texture feature represents the local binary pattern of each pixel. The Haralick texture features includes angular second moment, constract, correslation, sum of square(variance), inverse difference moment, sum average , sum variance, sum entropy and entropy. Both local binary pattern and Haralick texture features extract the global texture charateristics of an image by looking at each pixel. The feature vector is summarized in Table 2.

Tabel 2. Feature vector for training.

• RGB • Local binary pattern • Angular second moment • Constract • Correlation • Variance	• Inverse difference moment • Sum average • Sum variance • Sum entropy • Entropy

In the experiment, we randomly select a fraction of pixels (1000 for each image) from the original train images and compute the feature vector with respect to these pixels, and then extract the corresponding labels.

Fitting models

After generating the new training data, we use 80% of them to fit the model. In the experiment, we directly utilize the mudule of support vector machine and random forest classifier in the library of Scikit-learn.

Testing

We use the rest of the training data (20%) to test the perfromance of models. Specifically, we evaluate the performance in accuracy, precision, recall and F1 score.

Hyperparameter tuning

Here, we summarize all of the hyperparameters of these two models to be tuned in Tabel 3.

Tabel 3. Hyperparameters to be tuned.

SVM
• C: regularization patameter; • kernel: 'linear', 'poly', 'rbf', sigmoid, ...; • degree: degree of 'poly' kernel; • gamma: kernel coefficient for 'rbf', 'poly' and 'sigmoid';
Random Forest
• n_estimators: num of trees in the forest; • criterion: function to measure the quality of a split, 'gini' or 'entropy'; • max_depth: max depth of the tree; • min_samples_split: min num of samples to split an internal node; • min_samples_leaf: min num of samples to be at a leaf node; • max_features: num of features to consider when looking for the best split; • bootstrap: method for sampling data points(with or without replacement).

We will choose the models with best testing performance as the final models to do the inference.

Inference

After tuning the hyperparameter, we use the model with best testing performance to do the inference on test dataset. Since we need to predict the label of each pixel, all pixels of the input images will be used to compute the features.

SegNet and U-net

Training

We use 90% of the original training data for training. We extract randomly 600 small patches with size 96x96 from the training data and use 80% of them for training and 20% of them for validation.

Build a SegNet and U-net

We build the model of SegNet and U-net using following structures.

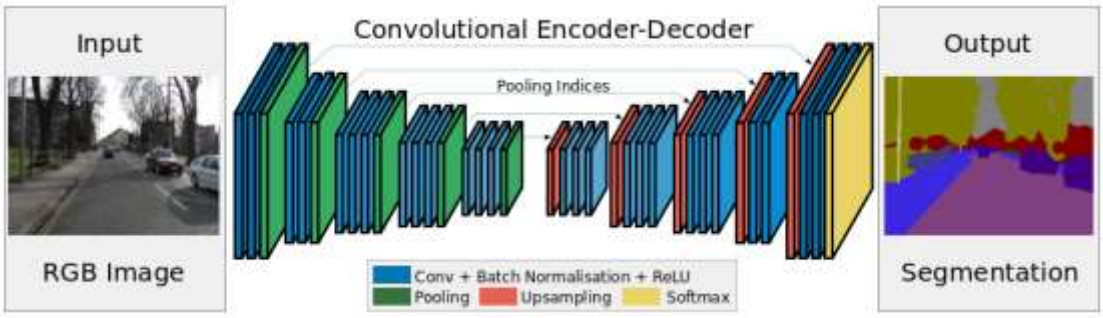

Figure 1. Structure of SegNet[2].

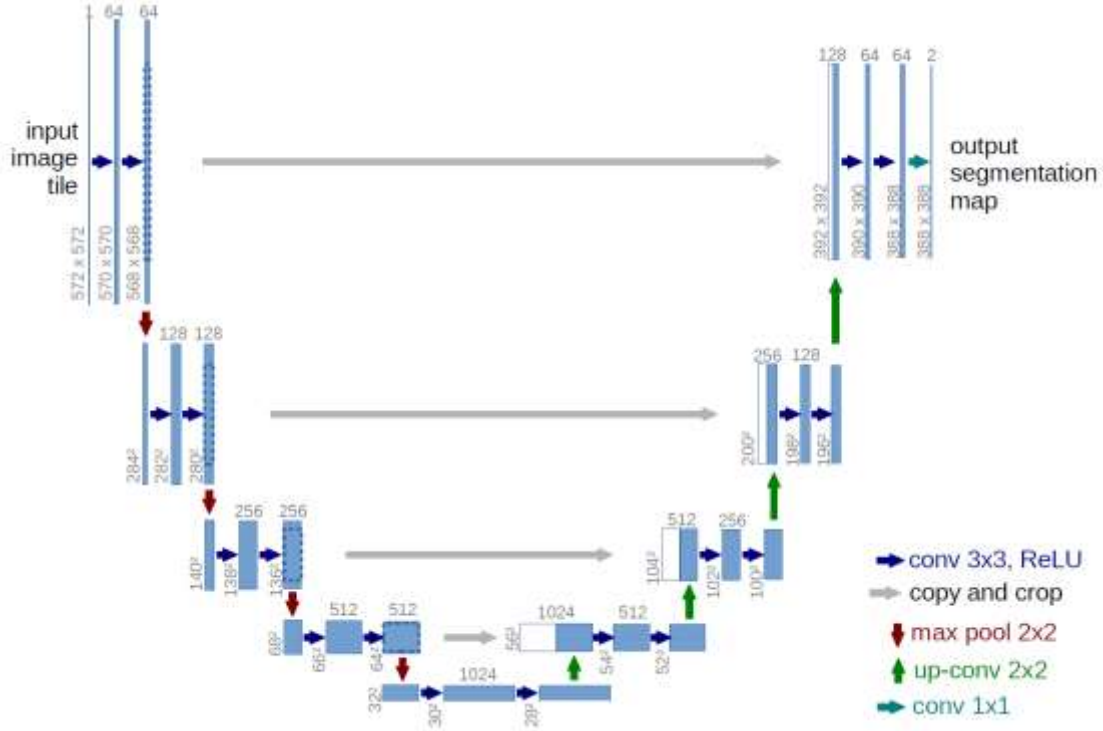

Figure 2. Structure of U-net[3].

Testing

We use the rest of the training data (10%) to test the trained model and evaluate their performance.

Hyperparameter Tuning

The hyperparameters in these two DNN models is the number of epoches.

Inference

We directly use the test image as the input of the best model we obtained and construct the outputs into image stacks.

Experiments (20 figures here)

Training and Testing

SVM and Random Forest

We use random search to find the best hyperparameters for our SVM and RF model, as shown in Table 1.

Table 1. Best hyperparameters and the corresponding testing results of the SVM and RF model.

SVM		**Random Forest**	
Hyperparameter	**Test results**	**Hyperparameter**	**Test results**
• C:- • kernel:'linear' • degree:- • gamma: 'auto'	• accuracy:0.83 • precision:0.86 • recall: 0.92 • F1: 0.89	• n_estimators:15 • criterion: 'entropy' • max_depth: 8 • min_samples_split: 3 • min_samples_leaf: 18 • max_features: 'auto' • bootstrap: 'True'	• accuracy: 0.84 • precision: 0.86 • recall: 0.94 • F1: 0.90

SVM

In the following, we show impact of each hyperparameters separately, while other hyperparameters are fixed as the optimal values in Tabel 2.

Tabel 2. Performance of the SVM model on different kernal functions(with defualt degree=3)

		Kernel Functions		
		'linear'	'rbf'	'sigmoid'
Training	**Accuracy**	0.8346	1.0000	0.7832
Testing	**Accuracy**	0.8270	0.7685	0.7682
	Precision	0.8620	0.7684	0.7682
	Recall	0.9225	1.0000	1.0000
	F1 score	0.8912	0.8690	0.8689

We can see that using linear kernel helps our SVM model achieve quite high performance on training and testing, which means our training data is linearly separable. Using RBF kernel also can

fit the model well but it is overfitting.

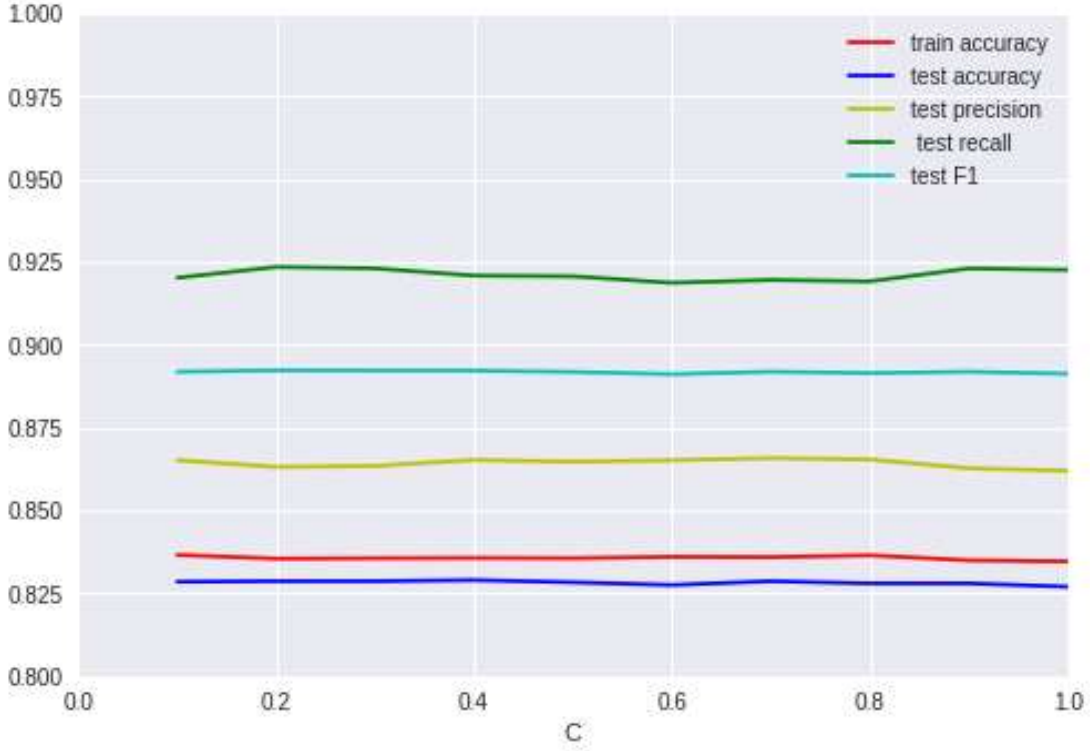

Figure 1. Performance of the SVM model on different values of the regularization parameter (C) when using Linear kernel.

From Figure 1, we can see that the regularization parameter doesn't have much influence of the traning and testing performance of the SVM model while using the Linear Kernel.

Random Forest

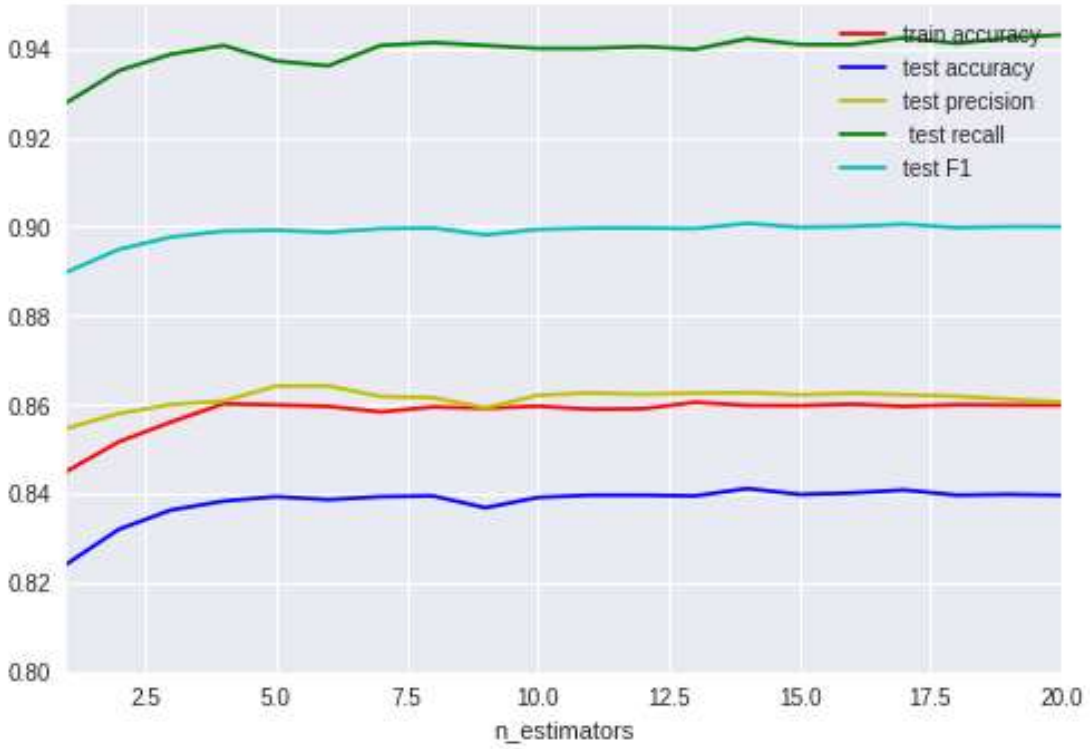

Figure 2. Performance of the RF model on different number of trees (n_estimators)

In Figure 2, we can see that the performance on training and testing increases when the number of trees increases from 1 to 15. But after that, the testing performance begin to decrease. In order to avoid overfitting, we choose n_estimators = 15.

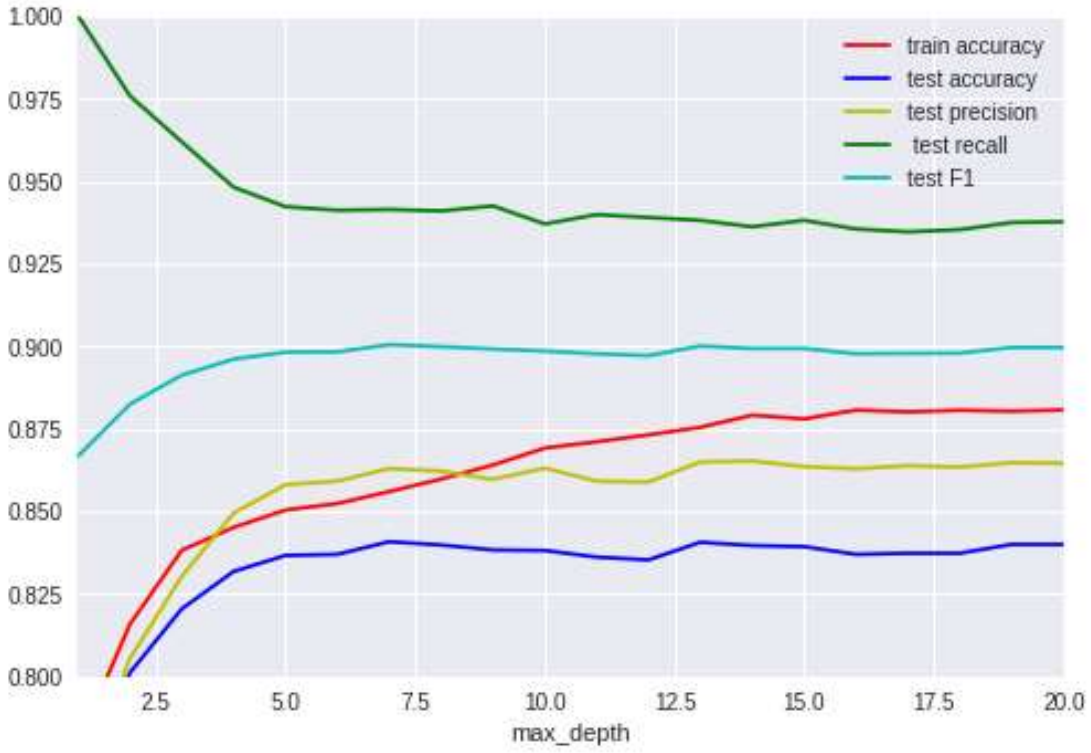

Figure 3. Performance of the RF model on different maximum number of depth (max_depth).

From Figure 3, we can see that training and testing performance increases when the maximum number of depth increases from 1 to 8, except the test recall. Test recall keeps decreasing. When max_depth is greater than 8, the train accuracy keep increasing, but test performance begin decreasing. In order to avoid overfitting, it's better to choose max_depth =8.

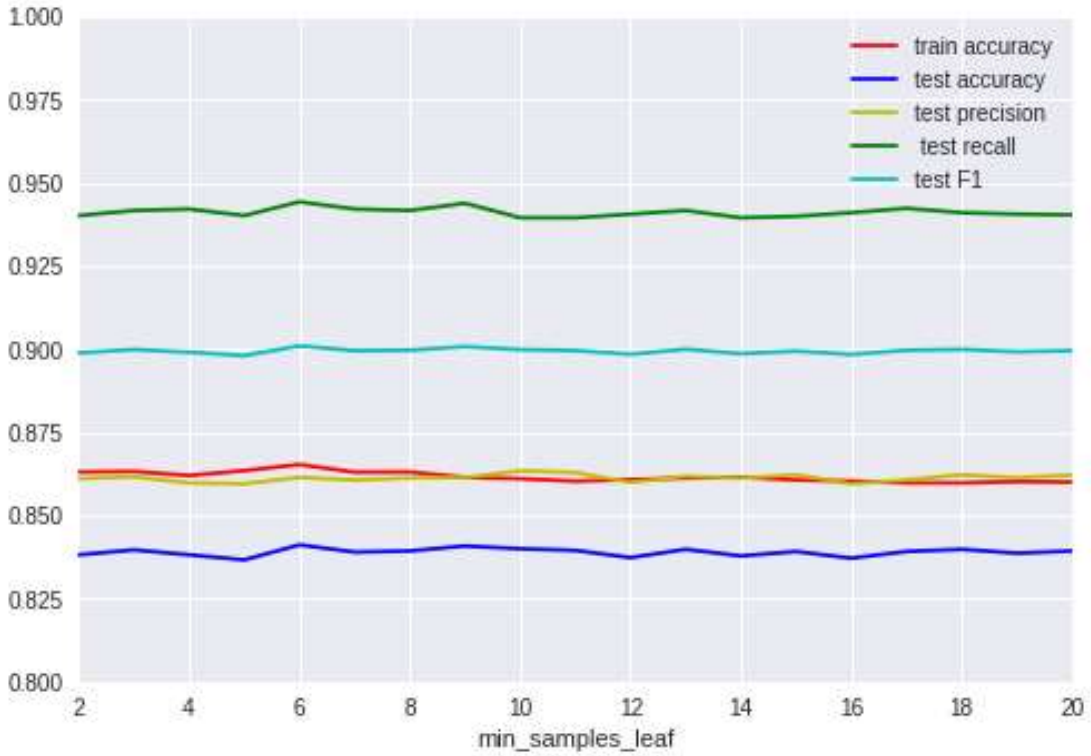

Figure 4.Performance of the RF model on different minimum number of samples to be at a leaf node (min_samples_leaf).

In Figure 4, we can see that min_samples_leaf does not have impact on the performance of the our Random Forest Model.

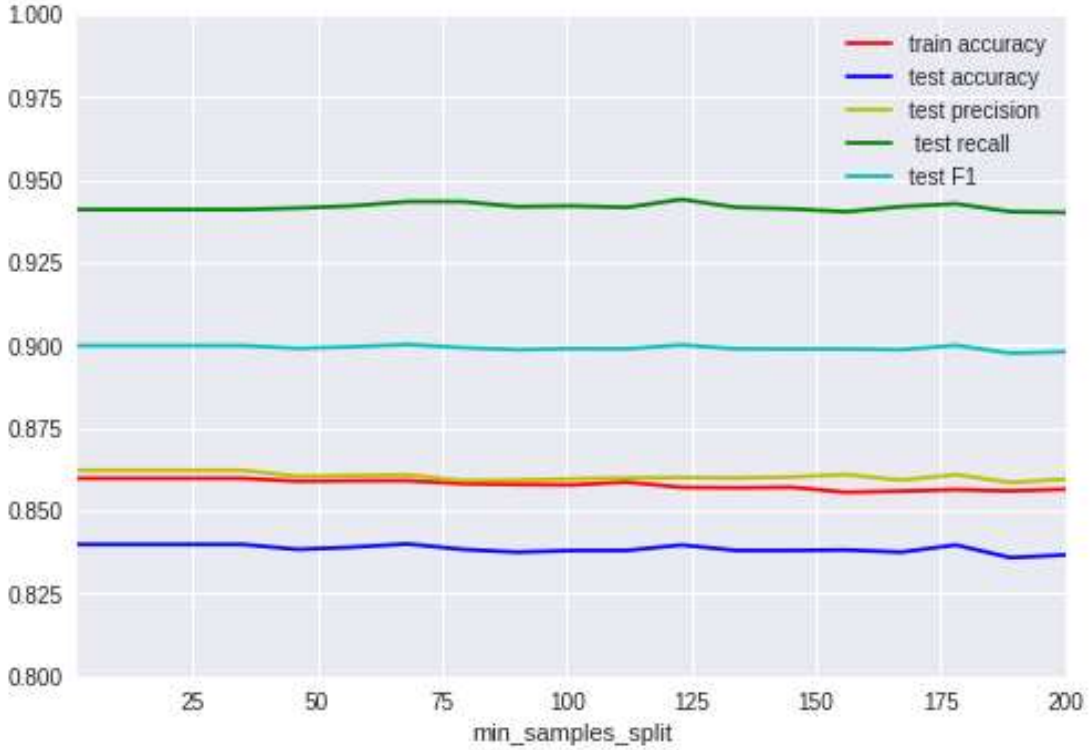

Figure 5. Performance of the RF model on different minimum number of samples to split an internal node (min_samples_split).

From Figure 5, we can see that min_sample_split does not have much impact on the performance of our Random Forest model.

In Table 3, we show the performance of using different criterions, max_features and bootstrap in our Random Forest model. The results shows that all of them do not have much impact on the performance of our Random Forest model.

Table 3. Performance of the RF model using different criterions, max_features and bootstrap.

		criterion		**max_features**		**bootstrap**	
		'gini'	'entropy'	'auto'	'sqrt'	'True'	'False'
Training	**Accuracy**	0.8611	0.8597	0.8598	0.8597	0.8597	0.8613
Testing	**Accuracy**	0.8385	0.8398	0.8398	0.8398	0.8398	0.8348
	Precision	0.8604	0.8622	0.8622	0.8622	0.8303	0.8244
	Recall	0.9416	0.9409	0.9409	0.9409	0.8398	0.8348
	F1 score	0.8991	0.8999	0.8998	0.8998	0.8293	0.8226

SegNet and U-net

SegNet

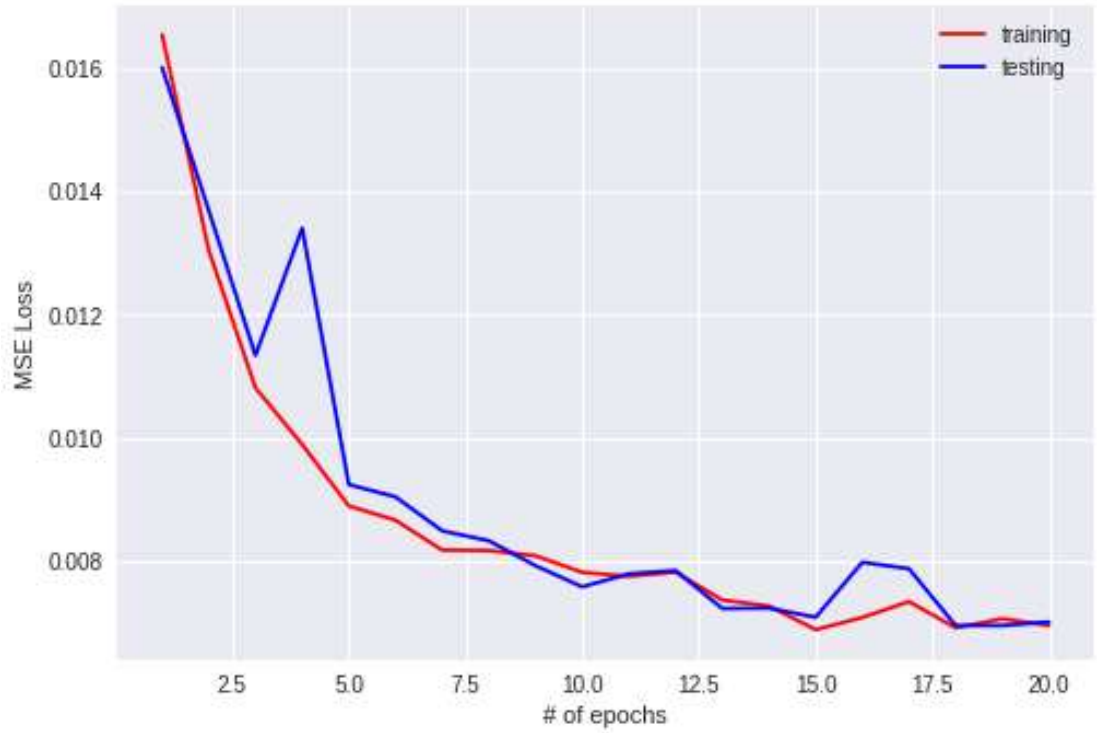

Figure 6. MSE Loss of training and validation (SegNet).

From Figure 6, we can see that the training loss and validation loss decrease as the number of epochs increases. And when the number of epochs is greater than 15, the training loss and validation loss don't change much.

In the following, we show all of the testing results on the testing set which includes 4 images. The outputs of our SegNet model are probabilities. In Figure 7, we show the first 2 predictions (in probability) and corresponding testing images. Then, we convert the probabilities into binary value (0 and 255) and show the results in Figure 8.

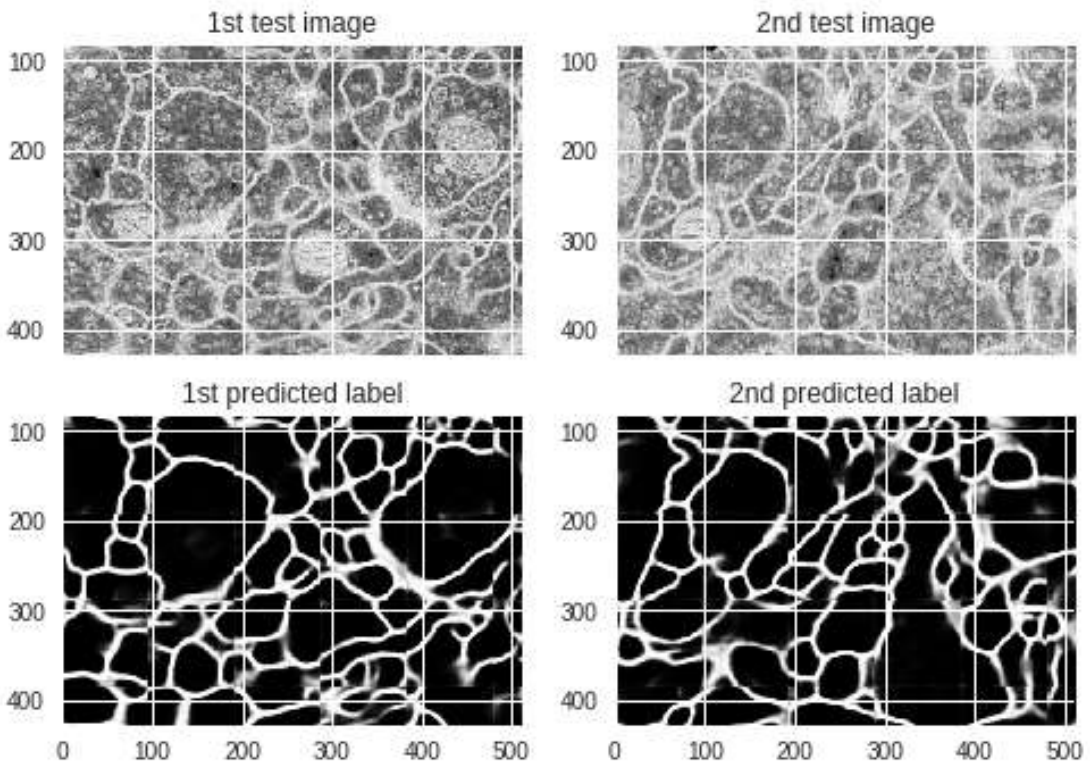

Figure 7. Predictions on the testing images 1 and 2 using SegNet(in probability).

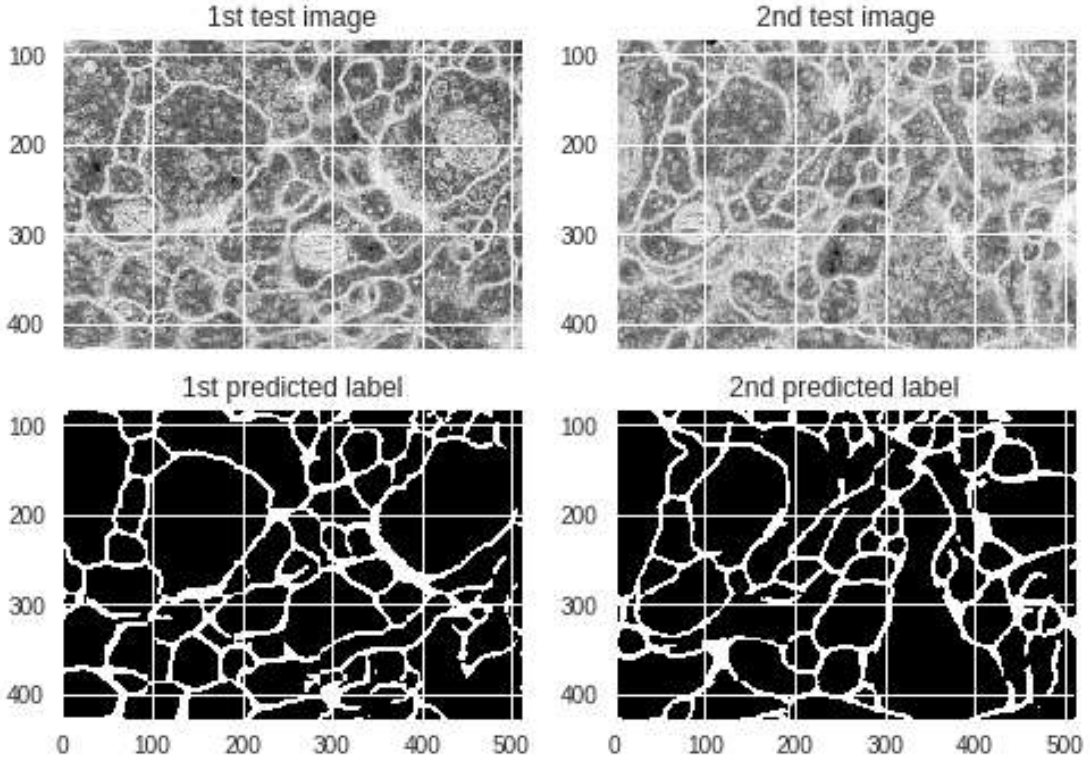

Figure 8. Predictions on the testing images 1 and 2 using SegNet (convet probability into binary value 0 and 255).

In Figure 9 and 10, we do the same process to show the prediction results of the last 2 testing images.

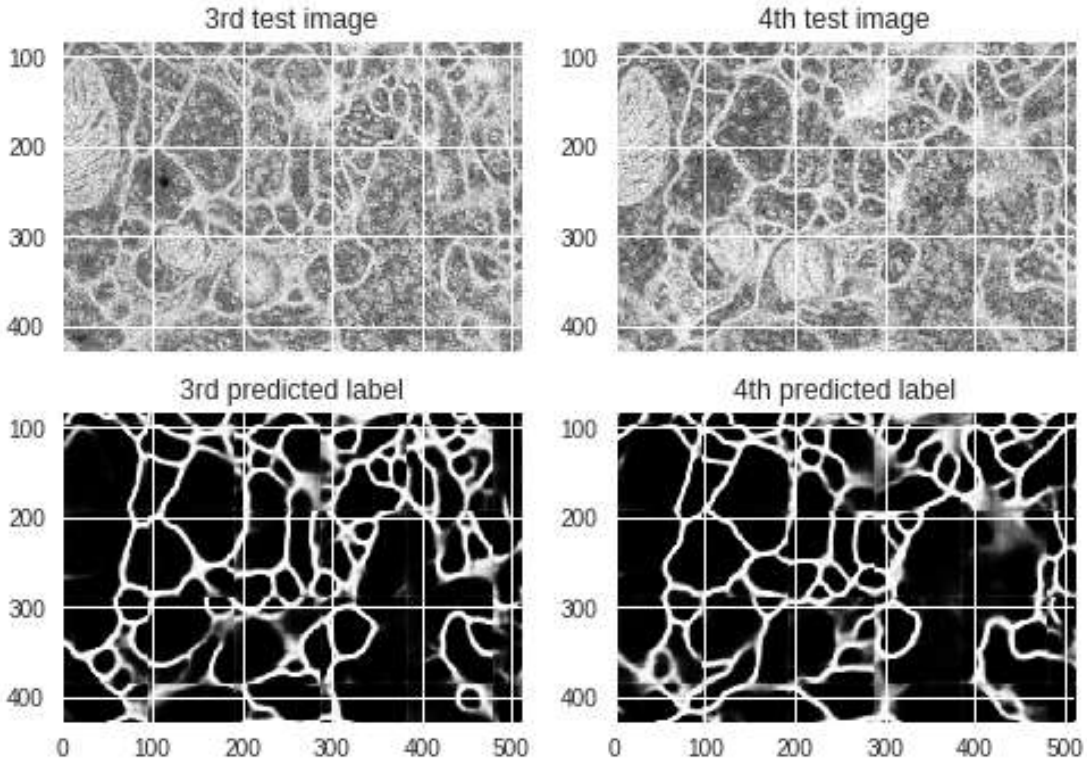

Figure 9. Predictions on the testing images 3 and 4 using SegNet (in probability).

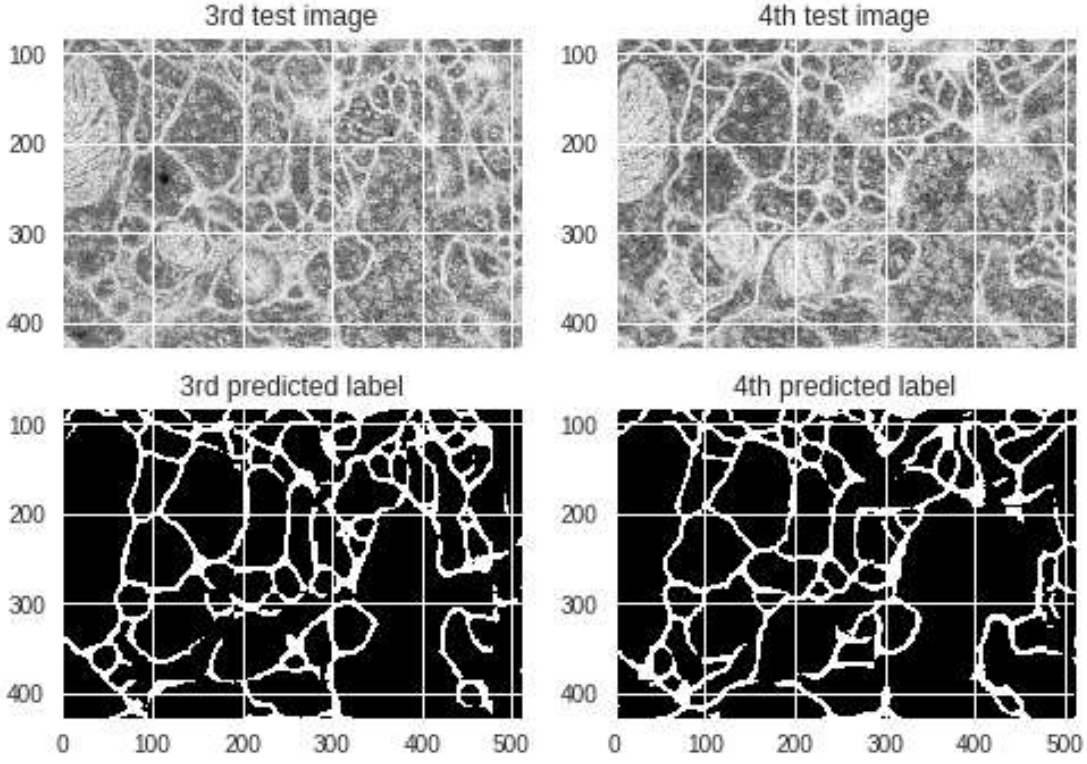

Figure 10. Predictions on the testing images 3 and 4 using SegNet (convet probability into binary value 0 and 255).

In Figure 7 and 9, we can see that SegNet model can only capture the obvious membranes but fails

to capture the structures that are not obvious, like the dark nucleus between the membranes. After we convert the probability value of each pixel into a binary value, as shown in Figure 8 and 10, the prediction contains less information about the original testing image. Comparing with the true label in Figure 11, we can see that in general SegNet can predict the profile of the neuronal structures but fails to capture the details.

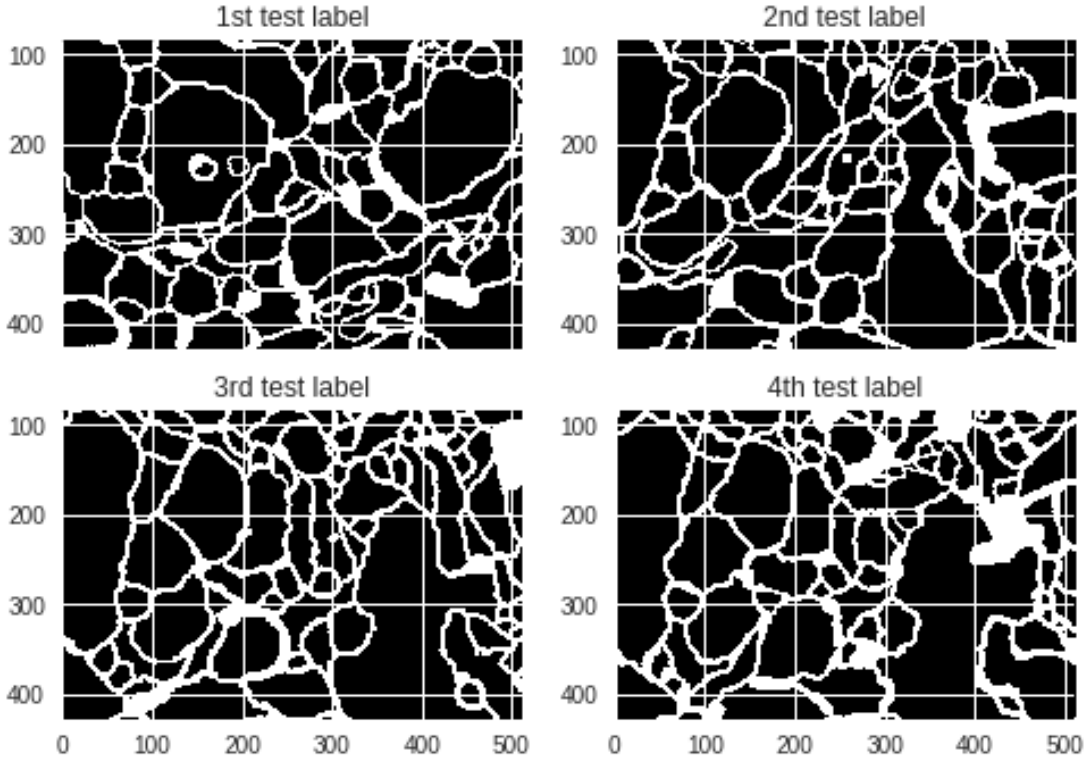

Figure 11. Testing Labels.

U-net

In the section, we use the same training and testing set of SegNet to fit and test the U-net model.

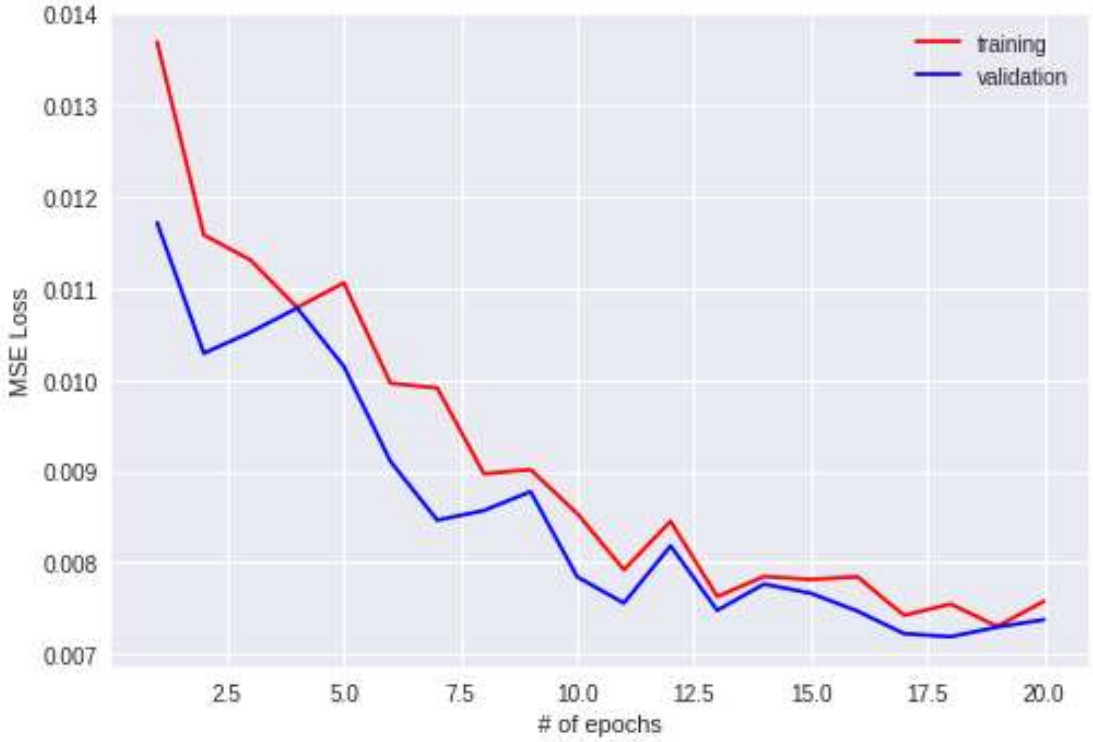

Figure 12. MSE Loss of training and validation (U-Net).

In Figure 12, we can see that the training loss and validation loss stop increasing when the number of epochs is greater than 17. So we use the model trained for 20 epochs as the final U-net model we obtained.

In the following, we show all of the testing results on the testing set which includes 4 images. The outputs of our U-net model are probabilities. In Figure 13, we show the first 2 predictions (in probability) and corresponding testing images. Then, we convert the probabilities into binary value (0 and 255) and show the results in Figure 14.

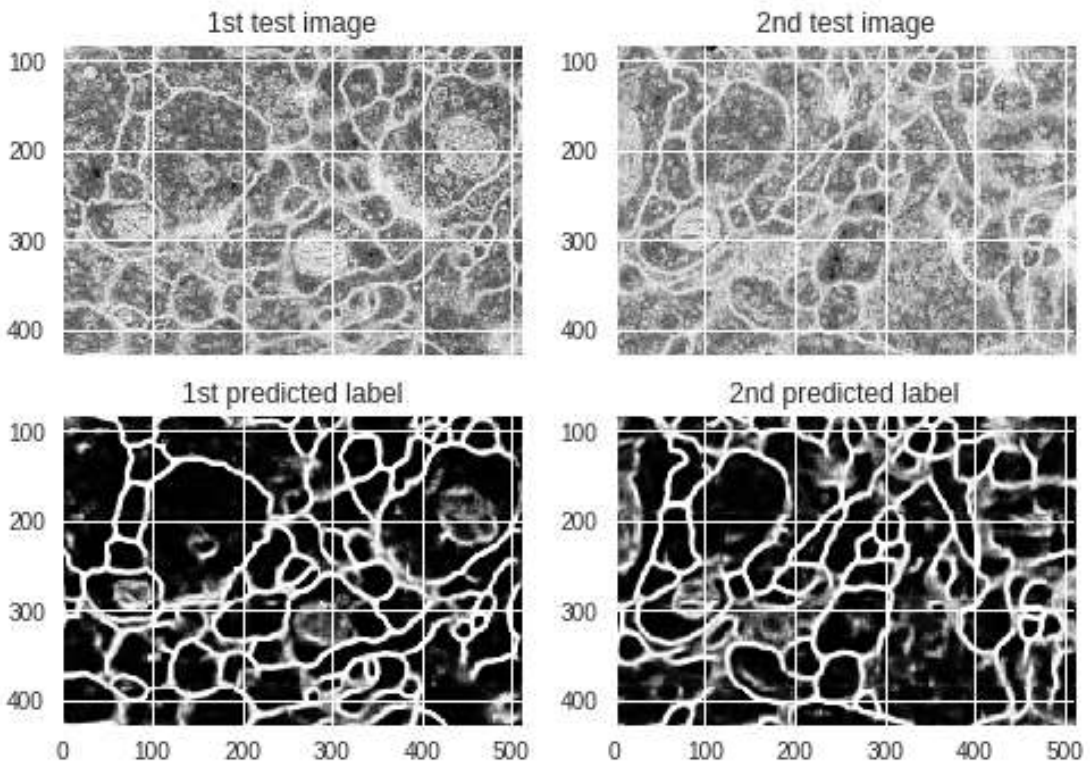

Figure 13. Predictions on the testing images 1 and 2 using U-net(in probability).

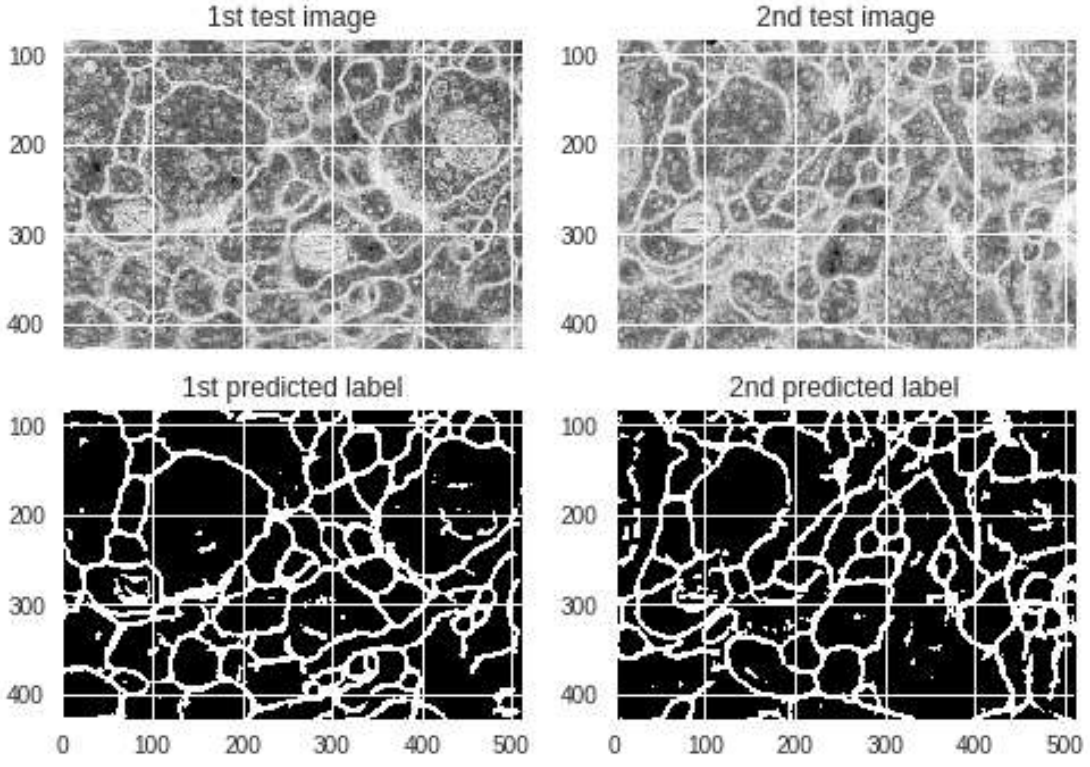

Figure 14. Predictions on the testing images 1 and 2 using U-net (convet probability into binary value 0 and 255).

In Figure 15 and 16, we do the same process to show the prediction results of the last 2 testing images.

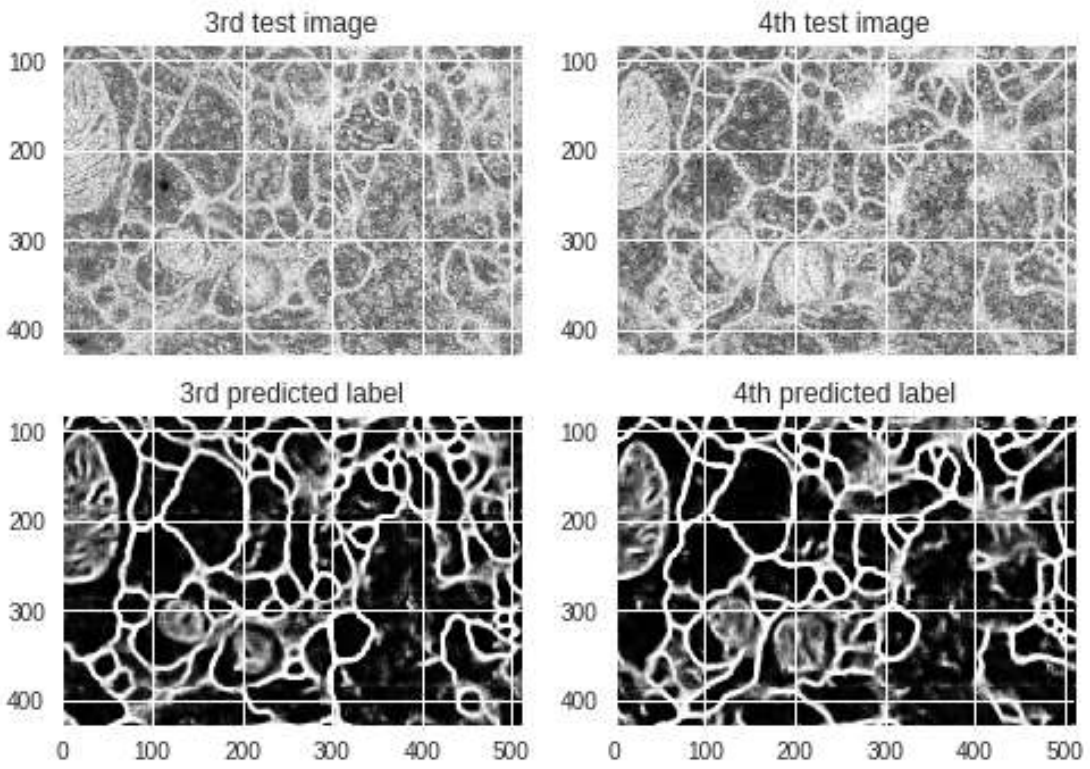

Figure 15. Predictions on the testing images 3 and 4 using U-net (in probability).

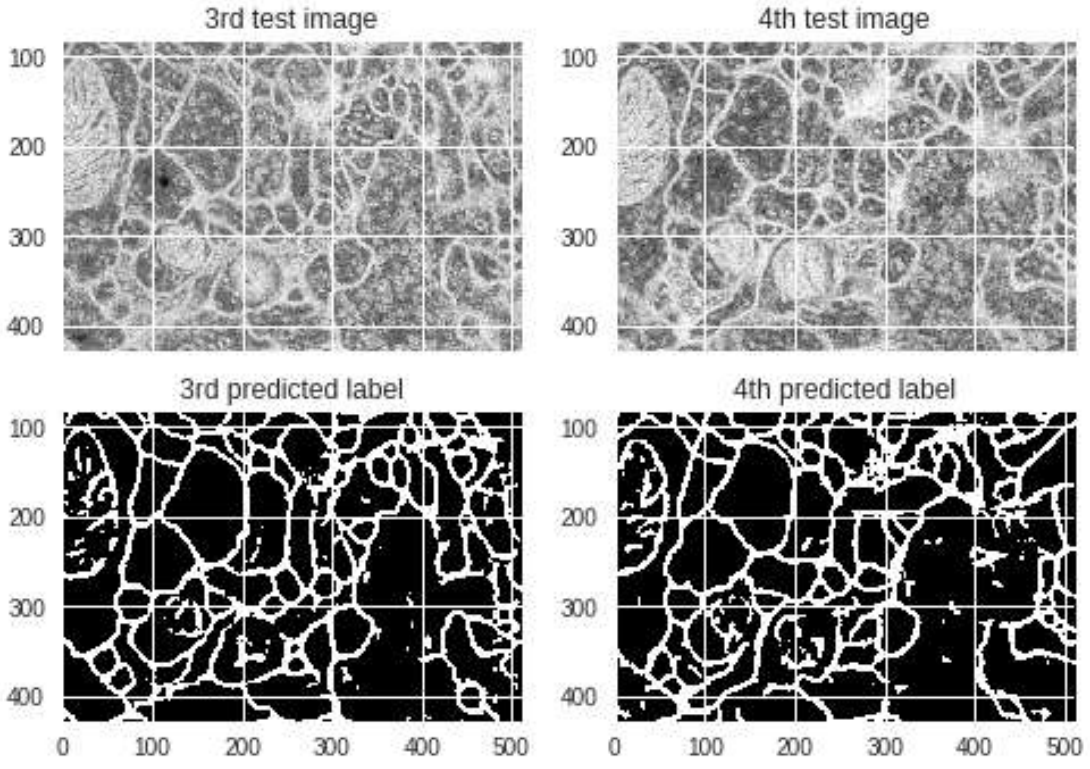

Figure 16. Predictions on the last two testing images using U-net (convet probability into binary value 0 and 255).

Compared to the corresponding outputs of SegNet, U-net is better to capture some objects that are not obvious (no matter it is the neuronal structure or not). However, after we convert the

outputs into binary value, the results fail to exclude the objects that are not neuronal structures. So U-net tends to capture objects that are not neuronal structures. From the prediction results of SegNet and U-net, we can see that we should be careful when we are converting the outputs that are in probabilities into the categories.

Inference

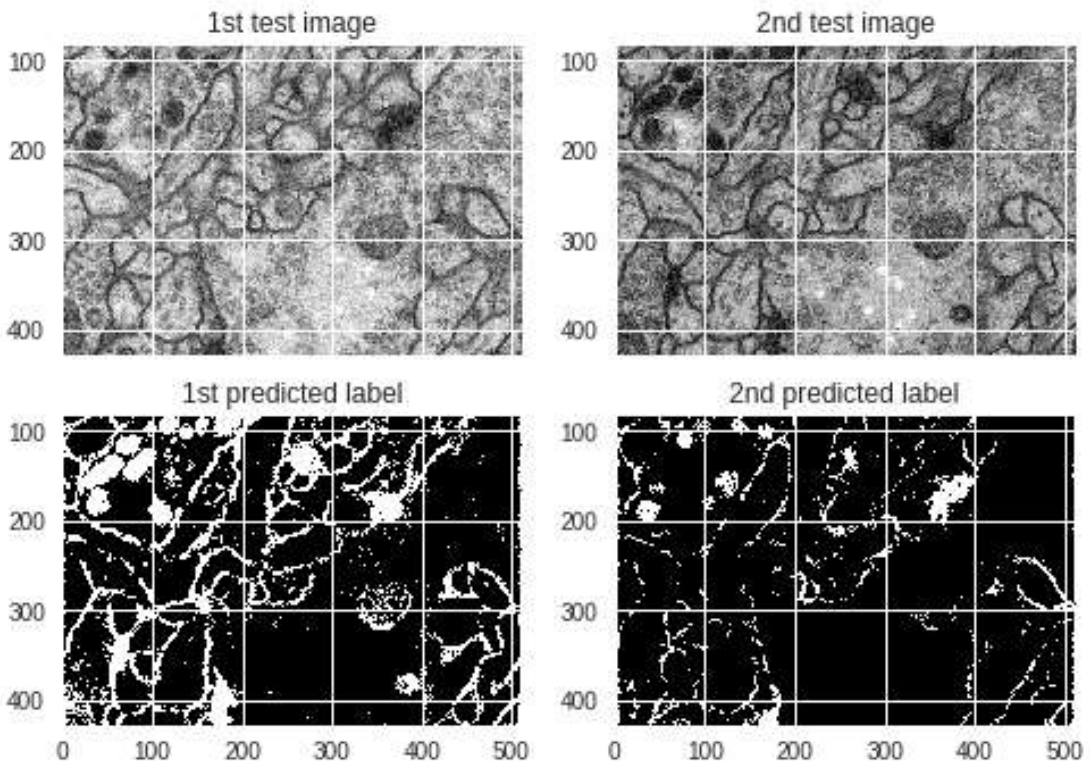

Figure 17. The test images and the corresponding predicted groundtruth using SVM.

Obviously, the inference results of SVM model is bad. It fails to caputure any complete neuronal structures.

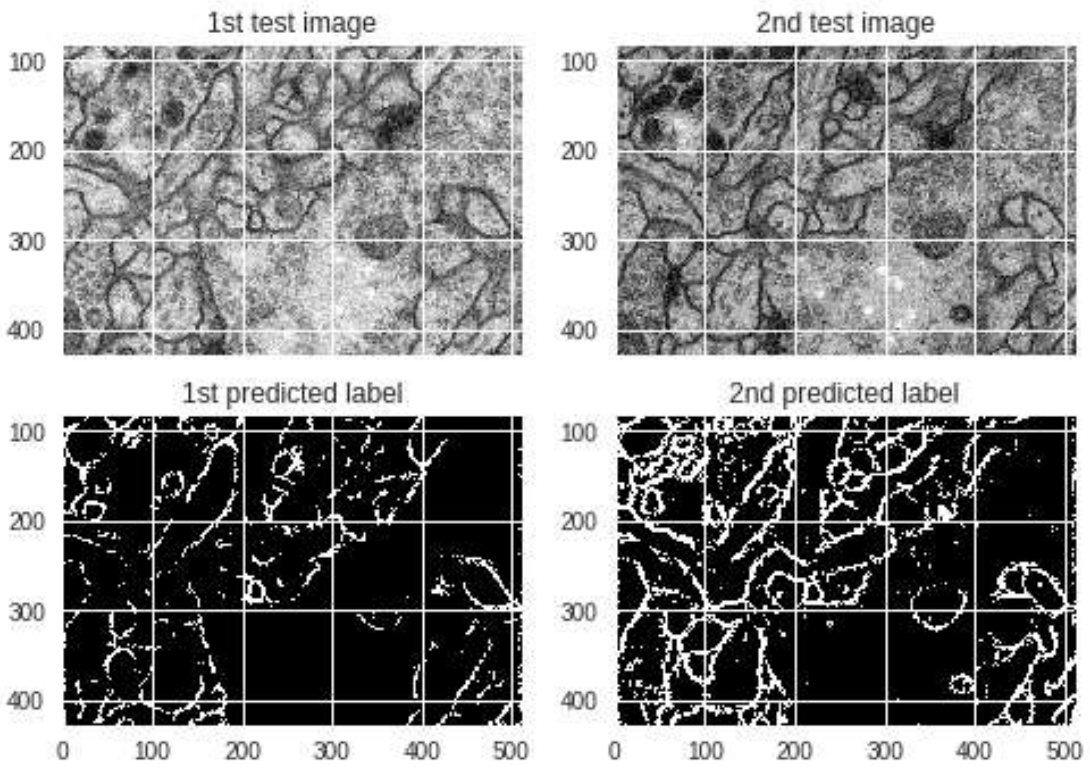

Figure 18. The test images and the corresponding predicted groundtruth using Random Forest.

The inference results of Random Forest model captures more informations of the inputs but still fails to caputure a complete structure.

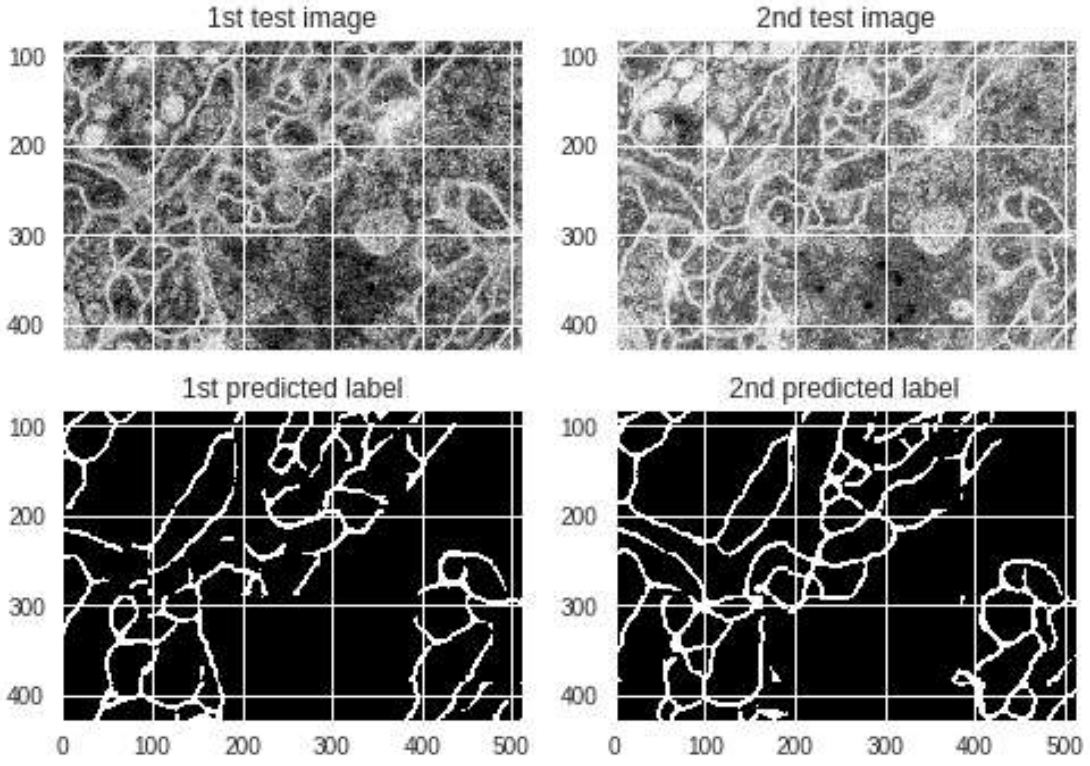

Figure 19. The test images and the corresponding predicted groundtruth using SegNet.

The results of SegNet is better than those of SVM and RF, but fails to capture the structures that are not obvious.

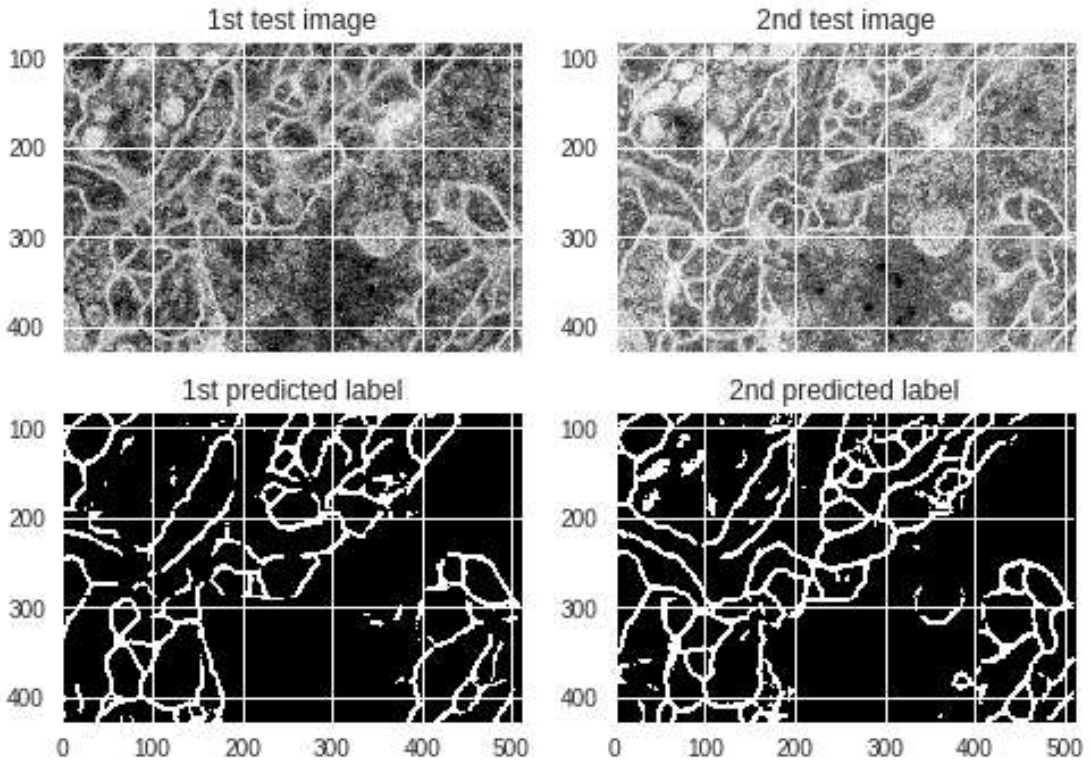

Figure 20. The test images and the corresponding predicted groundtruth using Unet.

The results of U-net is much better than SegNet because it caputures more structures of the origimal test images.

There have two metrics for the evaluation of inference results. One is Foreground-restricted Rand Scoring after border thinning (V^{Rand}), and another one is Foreground-restricted Information Theoretic Scoring after border thinning (V^{Info}). Since it is easy to verify that the inference results of U-net model is the best, we submit the results of U-net model to the website of ISBI Challenge to do evaluation. Finally, we get the feedback that is V^{Rand}= 0.827452189 and V^{Info} = 0.916702332.

Reference

[1] http://brainiac2.mit.edu/isbi_challenge/
[2] http://mi.eng.cam.ac.uk/projects/segnet/
[3] https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

Appendix

All the data and code are uploaded to my Github: https://github.com/ray-hu/Image-Segmentation