CS6350: Final Report:
Electron Microscopy Image Segmentation

Tianlong Zhang, Chien-Wei Sun

2018/12/14

1 Background

ISBI challenge is designed for segme-ntation of neuronal structures in EM stacks. The input images are all $512 * 512$ pixels, and it is very similar to EM images of connectomes. Usually connectomes are mainly referring to the brains and the neurons in the brain [1].

We want to find a better way to segment cells in a cell image, or even a cell volume. The cell has, for example, core, that looks like cell edge to normal people. If we can find an algorithm or function to solve this problem, we can save many time for experts to annotate cells manually.

2 Important ideas we explored

Image contains 2 dimensional information, and pixels in each image are related to neighbor pixels. The most important idea we have, is to transfer the image data into the feature vector, so it will be easier for classifiers to learn and make predictions. Since the choice of features is very important, we tried out different features, to make the learning and predicting more robust and more convincing. We also tried a deep neural network called **U-net**, to see how if has better performance on our project.

Deep neural network is a good approach to make correct predictions on input data. Successful training and proper neural network setting will result in high accuracy of the predictions. In the area of image processing, deep neural network is now becoming a popular method to perform image segmentation. In recent years, deep convolutional networks have outperformed the state of art in many visual recognition tasks, e.g. [2]

3 Ideas from class

In class, we saw a lot of machine learning methods, and we absorb some of them into our project.

1. Different algorithms. Several algorithms from class, like SVM and Naive Bayes, are implemented to find the best one for our project.
2. Optimization methods. We saw in class that the number of training epochs, the bias and variance are all important parameters in training, as well as the hyper parameters of the algorithms. Different parameters are tried to find the best combination to improve the accuracy.
3. Data management. The image data are interpreted into the data settings of homework. More specifically, we collect the image data and use image processing methods to get different features to represent the feature of the images, like Fourier transform, and use this information to represent each pixel, each pixel of the test image will be predicted by different classifiers we have trained.

4 Experiment and Results

The experiment are split into to portions. The first one is using the machine learning methods we saw in the class to predict the results, this is mostly done by Chien-Wei Sun. The second part is implementing the **U-net**, which is mostly done by Tianlong Zhang.

4.1 Feature collection

If we see our data pixel by pixel, we can only get the gray level value of each pixel as our feature. So we decide to process the image first with some widely used methods in image processing, and get the features from the processed result image.

- Gray Level Value.
- The fft power spectrum. The idea is we can see the image data as waves and use 2D Fourier Transfer on it. The Fast Fourier Transfer Power Spectrum will let us know the distribution of power and it could have certain patterns to observe if part of the images are somehow similar. We use $5 * 5 = 25$ pixels around our target pixels and do 2D FFT. Each value can be a feature. Therefore, it is 25 features for this method.
- The fft radial. Same as the power spectrum. This time we use only the pixels around our target pixels and classify these pixels by distance to the target pixels. We take 10 circles, so it is $19 * 19$ around for each pixel. Each circle is a feature, so we get 10 feature for this method.
- Law's Texture Analysis [3]. Kenneth Ivan Laws find a way to analyze image pattern by doing convolution on images with some filters. Please see more detail of this method in code. We get 10 feature for this method.
- Image Gradient. This method is often to find edges in image processing. OpenCV has Laplacian, Sobel functions to called and use directly use them. We get 3 features using this method.

We get 49 features from these methods in total, and that is our new dataset. We could use Perceptron, KNN using LSH method directly from this new dataset. However, we need another process to make the features used in methods like SVM, Logistic, and Naive Bayes.

We need to convert decimal and floating point number to binary lists. To do this, we go over our image dataset and get the maximum value and minimum value. Then we divide the all value between the max and min into 8 scales. If the value is located on a scale, it will be 1, otherwise 0. So a binary list will have only one 1 and the other value should be 0.

One thing to note: We should normalize the decimal and floating point number for easier implementation.

4.2 Machine Learning Methods

Same as in the HW, we run cross-validation and get the best hyper-parameters for each method. Because one image can take some very long time to process, for training and testing set we only use about 20000 number of pixels data. Here is what we have:

1. Perceptron Algorithm
   This one is same as the one we implement in HW2. We used original dataset that has decimal and floating point numbers. The result seems good because accuracy is around 0.9. However, the original rate of positive/negative value is already 0.13/0.87. So we just have very little improvement here.
2. Support Vector Machine using Stochastic Sub-Gradient Descent
   This one is same as the one we implement in HW5. We used new dataset that has binary list for each feature. The result is not so good after a few trials.

3. Naive Bayes Algorithm This one is same as the one we implement in HW5. Because we have different results to other methods in HW5 so we want to see how it works here. We used new dataset that has binary list for each feature. Unfortunately, the result is also not so good after a few trials.

4. KNN: Locality-Sensitive Hashing(LSH) We try to use other method that didn't include in HWs. Locality-Sensitive Hashing is a method to find K-nearest neighbors in high-dimension vector data [4]. Our idea is to find the closest pixel and using its label as our predict result. We can also find more neighbors and get the mean of the labels. Here, we just find the closest one. As we expected, if we set w to 10000 we can get 100% on training set. The w is how much distance could be tolerant to be a neighbor. If we set too small w, we can't find much neighbor so the result is not correct. On the other hand, if we set too large w, then we may find not close enough pixels as neighbors.

   LSH is interesting method and have other parameter to try. We just implemented it and try for w this time. There are number of hash table, and the element of hash value picked to generate hash vector, can be picked as parameters.

	Best hyper-parameters	Train P/R/F1 or Accuracy	Test P/R/F1 or Accuracy
Perceptron	$\gamma_0 = 0.1$	0.92	0.90
SVM	$\gamma_0 = 1, C = 10$	0.31/0.71/0.19	0.17/0.54/0.18
Naive Bayes	$\lambda = 1.5$	0.55/0.44/0.77	0.44/0.29/0.90
LSH	$w = 10000/100000$	1.00/0.46	0.0004/0.5632

Table 1: Results table for four methods we have experimented with.

From the result table, we know that maybe the impact of turning decimal and floating point number to binary vector. Same as HW5, we now know our result in HW5 could be improve if we have better way to handle this scarification. Also, as mentioned in the class, we didn't normalize on the data, so the KNN result looks not so good when we change the value w.

4.3 Discussion: Cons for Machine Learning Methods

Although we didn't get better performance using the method in this class, we learn something from this experiment. The pain points of using machine learning methods to our problem are:

1. The performance is not better then deep learning methods.
2. For image data, it is hard to decide which features to pick.
3. Even we decide which features to use, it cost us some time to preprocess on training dataset and also the dataset we want to query. The time spend here is not faster then doing deep learning directly.
4. The intermediate data before final result may cost large amount of space. In our case here, we need to use at least $200MB$ intermediate data to store just one $512 * 512$ for an image. It is much more larger then original image data.
5. The performance depend on the method we extract the feature. For example, in our HW5, we got the preprocessed binary data. To generate this sort of data, we must sacrifice some performance to reduce the burden of computing.
6. Need a better way to track effects for each feature. Our previous solution–using deep learning is easier to set up and have better result. Although we kind of knowing what we are doing because we generate the features(instead of black box in deep learning), the different source image is hard to track every changes in those features.

4.4 Neural Network

In this part, we will discuss how nerual network, specially deep neural network, will perform on image segmentation.

So far, I have collected the dataset from the image segmentation challenge called *ISBI*. There are 30 training samples and 30 labels for training. The examples are very limited, but by applying deep neural network, the result can be very promising.

This project is using *Keras* with *tensorflow-gpu* as backend, and *Python3* with some libraries such as *numpy*, *PIL*, *TIF*, etc.

In this part of the project, I pre-process the training examples and labels by doing a module called **ImageDataGenerator** in **keras.preprocessing.image** to do data augmentation. This will give training data more robust representation in the network. Then I apply a neural network called **u-Net** which is shown in Figure 1.

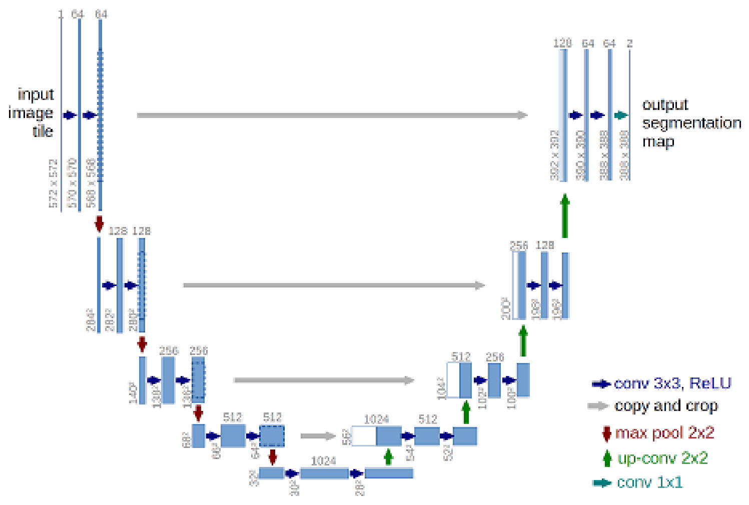

Figure 1: U-Net architecture

The network architecture is illustrated in Figure 2. It consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two $3 * 3$ convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a $2 * 2$ max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels.

Every step in the expansive path consists of an upsampling of the feature map followed by a $2 * 2$ convolution (upconvolution) that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two $3 * 3$ convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a $1 * 1$ convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers [5].

Output from the network is a $512 * 512$ array which represents a mask that should be learned. Sigmoid activation function makes sure that mask pixels are in [0, 1] range.

The model is trained for 12 epochs. After 12 epochs, calculated accuracy is about 0.9593 which is shown in Figure 2. Loss function for the training is basically just a binary cross entropy.

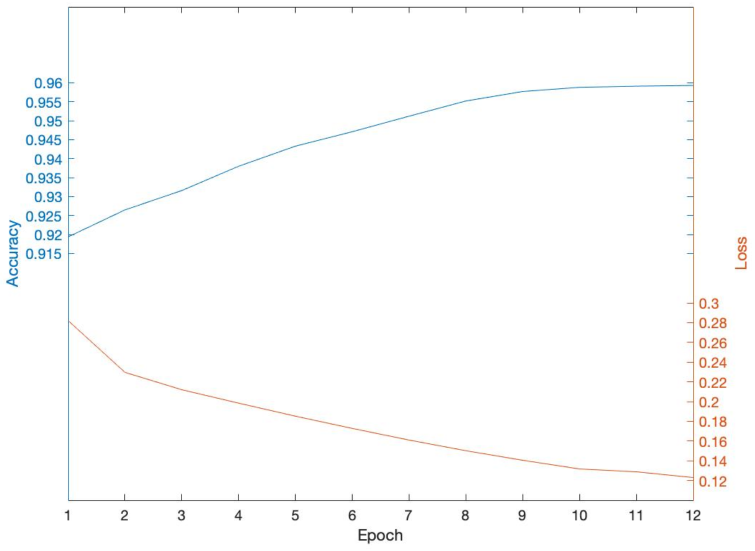

Figure 2: U-Net accuracy and loss

According to the loss and accuracy figure, we can clearly see that as the epoch increases, the accuracy also increases and the loss decreases. This means the network is working properly. Also, we can see that the curve begin to be flat after 10 epochs, this shows that the loss and accuracy are converging. Each epoch will run on the *Titan* GPU for 15 minute, and I collected the predictions after 10 epochs.

In order to have a visual idea on how the test images and predictions are, the result of the segmentation on test images are shown in Figure 3. Due to the length limit of the report, only 4 test images and 4 test results are shown, the test images and result images are next to each other so it is easier to compare.

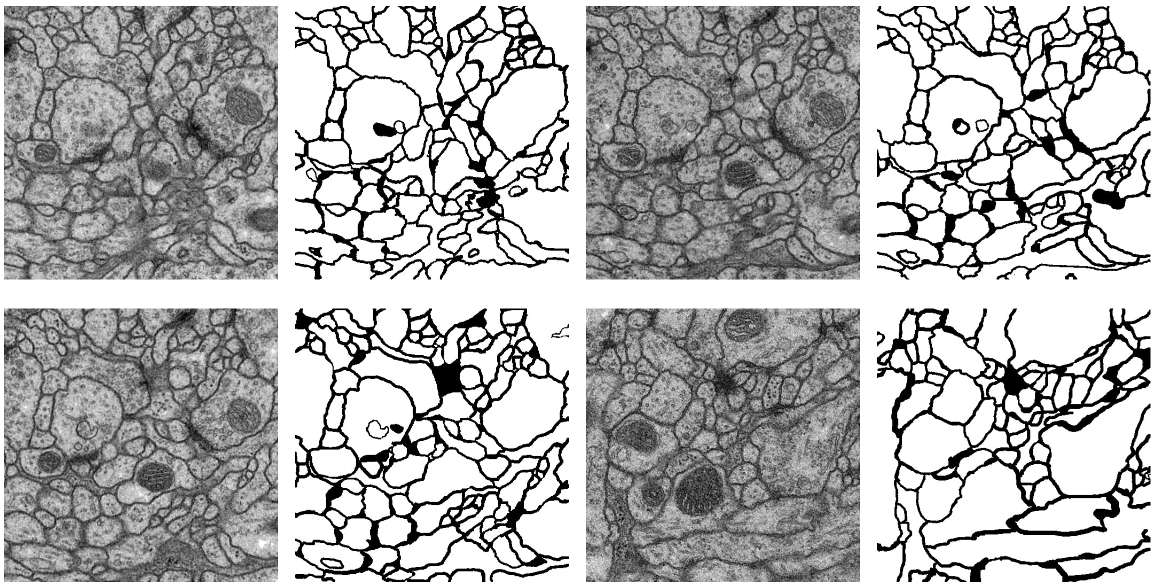

Figure 3: The result of the segmentation

We can see that the segmentation result is very good. The black lines indicate the membranes of the cells, and the core of the cell will not be mislabeled as membranes.

So far, we have seen different machine learning algorithms and deep neural network on the image segmentation. Deep neural network has a better result but the computational time is very long, and it is hard for us to get the physical meaning inside the network. Even if we print out the output of each layer, we will still find the result very confusing. However, traditional machine learning methods with features from image process will give us a better understanding of what is going on along the way, and it will be easier for us to improve the result be looking at the features and try to find a better feature to represent the image, or to find a better algorithm that will classify the image feature of cell images more correctly.

5 Plans

This work focuses on the basic machine learning methods taught in class, implementation of the feature extraction on the images, also the experiment on deep neural network. Considering the time and the amount of knowledge we have for this project, the results are good, but if we have more time, we will explore on the following topics:

1. More advanced machine learning algorithms, like K-means. Also other methods that have better performance on the clustering problem. Because image segmentation on cells is more appropriately classified as a clustering problem. This project assumes the problem as a classification problem.
2. More accurate feature of the images. Right now, we only looked at simple features like the Fourier transform. But we can gather more robust features, for example, the edge shape of the cells, the texture of the cells, or the relationship of the neighbor cells etc.
3. More appropriate evaluation method. In this report, we use pixel based evaluation, which may not be the best interpreter of the correctness on cell segmentation. If time permits, we will use the VOI (Variation of Information) metrics on segmentation results, which means we should see whether two random points in a ground truth cell will be in two different prediction cells.

References

[1] ISBI Challenge: Segmentation of neuronal structures in EM stacks. `http://brainiac2.mit.edu/isbi_challenge/`.

[2] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[3] Kenneth Ivan Laws. Textured image segmentation. March University of Southern California, 1980.

[4] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, pages 253–262, New York, NY, USA, 2004. ACM.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.