COMP9517 PROJECT

19 T2

TEAM GANK

Cong CONG z3414050
Guanqun ZHOU z5174741
Fan LIU z5188300
Xiao TAN z3413898
Zixin Fang z5210737

Abstract

Our project has successfully developed some methods for a neuron membranes segmentation task with some promising results. Two machine learning-based approaches (SVM and Random Forest) and two deep learning-based approaches (U-net and ResUnet) are developed to solve tackle this problem.

Keywords -- neuron membranes segmentation, machine learning, deep learning

Introduction

Image Segmentations plays a significant role in the field of medical image analyse. In this project, which is based on ISBI 2012 EM Segmentation Challenge, 30 images are provided along with their labels. To tackle the above task, four methods are implemented, namely two machine learning-based approaches and two deep learning-based approaches. For machine learning models, image features are extracted using different methods (Haralick and LBP etc.). Then two machine learning algorithms (SVM and Random Forest) are trained based on the extracted features. For deep learning models, on the other hand, U-net and RestUnet are successfully implemented, and some modifications are implemented for further improvement of these models. Outputs of each model are shown with some necessary comparisons. In general, it can be found that deep learning models perform better than machine learning models.

Experimental Setup

In this project, the dataset is given from an ISBI Challenge, namely Segmentation of neuronal structures in EM stacks. This dataset contains 30 microscopic images of Drosophila larva ventral nerve cord (VNC), and corresponding image masks of these images are also provided.

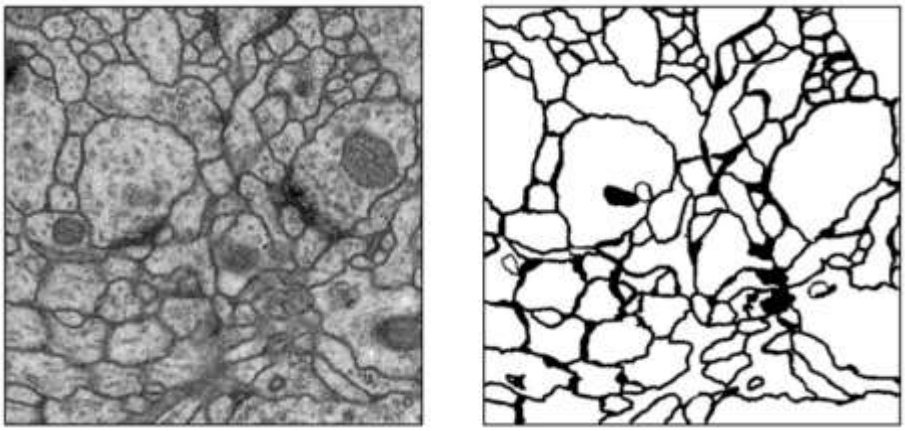


Figure 1. Example of Provided Image and Its Label

Evaluation of model performance is based on Rand score V^{Rand} and Information Theoretic score V^{Info}. In order to run the evaluation script, two images with TIFF format are required to be opened. To satisfy this requirement, both model outputs and given labels are assembled into a 3D TIFF image.

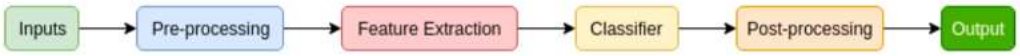
Methods

Machine Learning

In machine learning, we implemented two combinations of model and feature extraction methods, namely feature extraction with SVM, and LBP feature with Random Forest.

Feature extraction + SVM

In this scenario, the pipeline is like below:



i. In the pre-processing stage, we have done thresholding and blob detection. In this way, we can remove the noise in original images in order to guarantee the accuracy in following steps.

ii. In the feature extraction stage, the whole work is divided into two main steps:

a) For each image and corresponding label:

1. Randomly select 4000 pixels
2. For each pixel, set a patch whose size is 3x3 and set value of the pixel having the same location in corresponding ground truth as 0 or 255

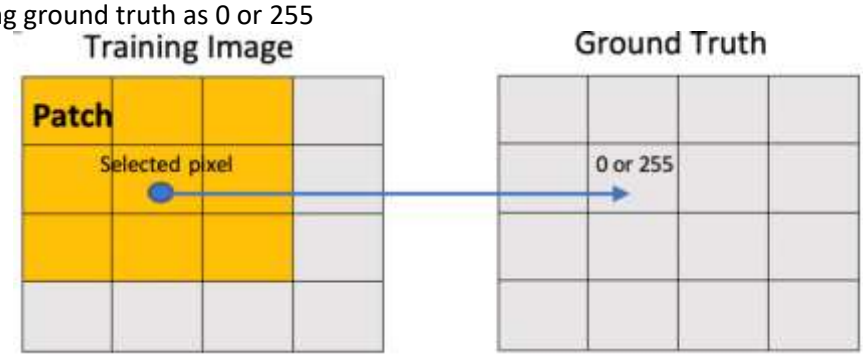


Figure 2. Illustration of patch selection

3. Transform the image features into a vector T (length is 24)

b) The contents of vector cells include:

- The intensities (3x3) of the patch S
- The median value of patch S
- The range of patch, which equals to Max(S) – Min(S)
- The energy of patch, which equals to $\sum_{i \in S} i^2$
- The 2rd, 3nd and 4th spatial moments (3) of patch S, which equals to $\frac{1}{9}\sum_{i \in S}(i-\mu)^r$ for r = 2, 3 and 4 respectively, where μ is the mean value of patch S.
- The gradients (3x3) of patch S, which can be used to increase the visibility of edges and other details.

iii. In this scenario, SVM (Support Vector Machine) is chosen as the classifier. Since SVM is a fundamental binary classification algorithm, and in the feature extraction stage, we have extracted 4000 feature vectors with the corresponding label whose value is 0 or 255. In this way, we have successfully transformed this image segmentation problem into a binary classification problem where SVM can perform perfectly.

iv. Post-processing is a crucial part of image processing, and it is worth implemented in this project. Especially for the results came up from machine learning, the blobs in the masks were often isolated, whereas post-processing is helpful to join these blobs together and reduce the noise of the image.

v.

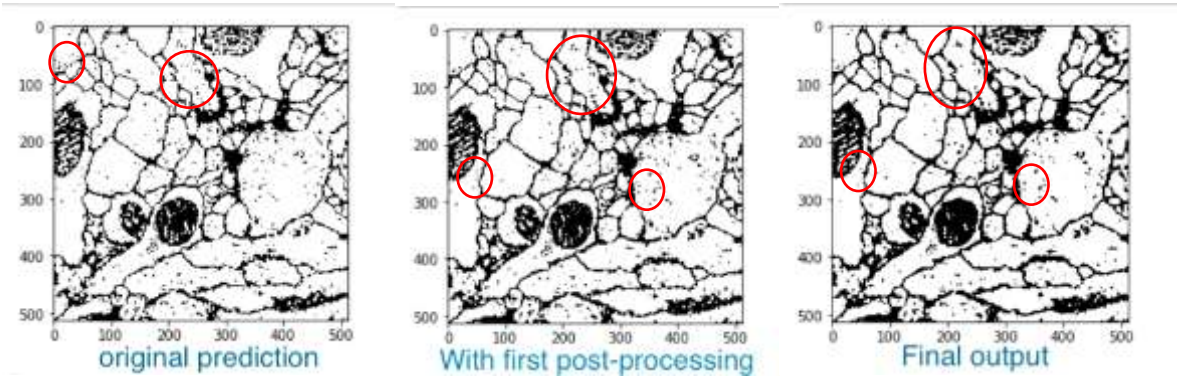


Figure 3. Results before and after post-processing.

Further improvements have implemented, other than the conventional methods for feature extraction+SVM, which is called image patch-sampling. For each image in training, we split them into 4 patches respectively. In this way, we have a batch of images whose size is 256x256, just like the diagram below showing:

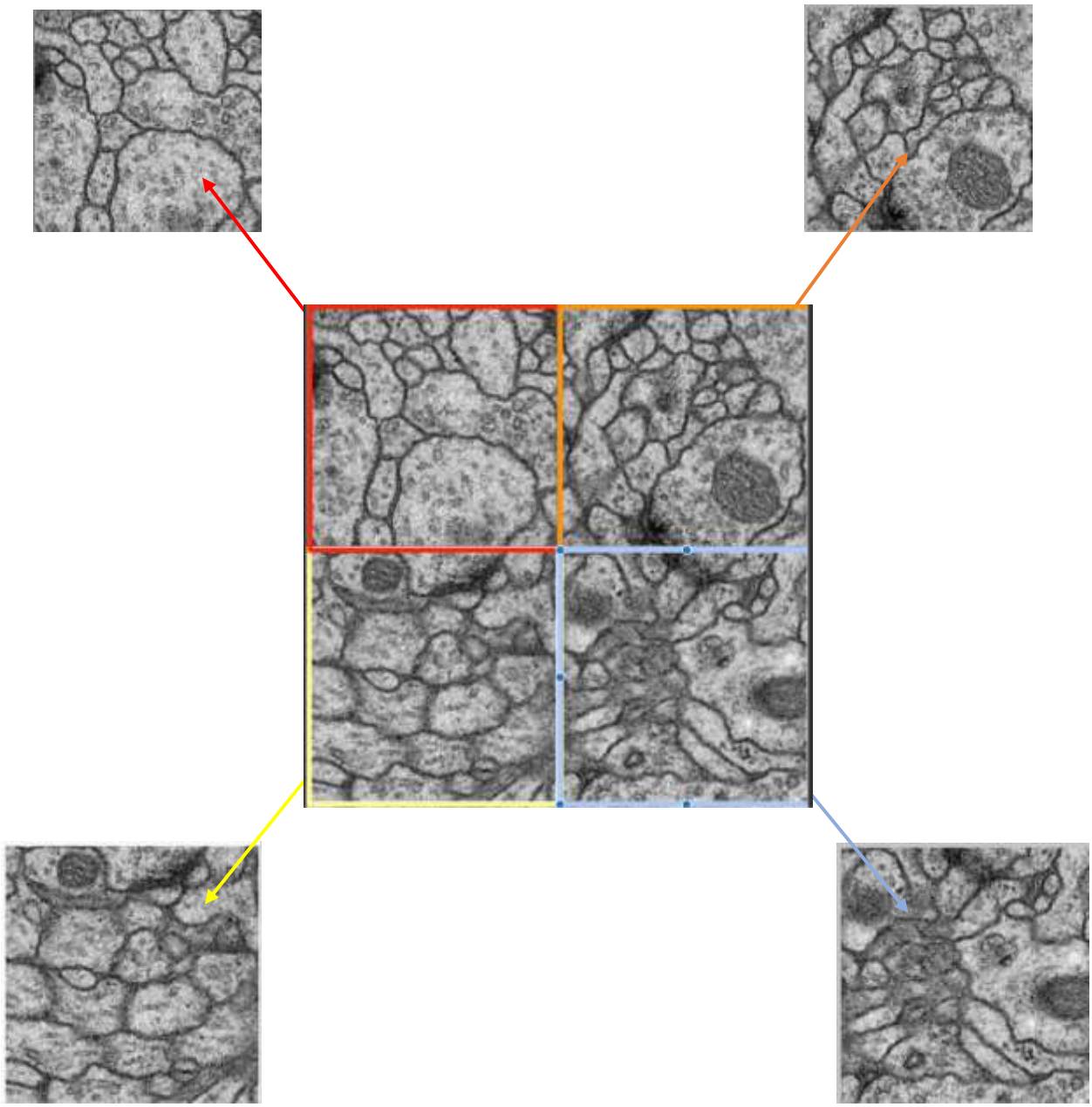


Figure 4. Diagram of image patching

After image patching, 1000 pixels are selected in each 256x256 image and produce corresponding vectors and label vectors. Then the model is trained based on these vectors. In prediction, each testing image is split into 4 patches and these small patches are used for prediction. At the end of prediction, we merge every 4 patches together so that we obtain several predicted images with original size. It can be seen that image patching can improve accuracy of our predication.

With or without image patching	accuracy
Yes	0.66
No	0.71

Table 1. Result comparison with or without image patching (SVM)

Comparison of SVM Parameter

According to result of grid search below. Optimal parameter setting of SVM is C = 1000, gamma = 'auto', kernel = 'rbf'.

```
# Tuning hyper-parameters for recall

import finish 0.00012493133544921875
fit finish 9346.804817199707
Best parameters set found on development set:

{'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
para 9346.805188179016

Grid scores on development set:

stage1 9346.805423259735
0.618 (+/-0.025) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
0.500 (+/-0.000) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
0.712 (+/-0.041) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
0.618 (+/-0.025) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
0.710 (+/-0.039) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
0.712 (+/-0.041) for {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
0.695 (+/-0.032) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
0.712 (+/-0.041) for {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
0.700 (+/-0.036) for {'C': 1, 'kernel': 'linear'}
0.695 (+/-0.036) for {'C': 10, 'kernel': 'linear'}
0.695 (+/-0.035) for {'C': 100, 'kernel': 'linear'}
0.695 (+/-0.036) for {'C': 1000, 'kernel': 'linear'}
specific 9346.80577993393
```

Figure 5. Result of grid search (SVM)

Li (2018)[1] provides other feature extraction methods which are effective to segment the image with SVM. Therefore, on the basis of the above features, several extra features were added to above features as an attempt to improve the result. The formula 'grey histogram' of the 3x3 patch is used in the calculation of extra features. And that formula is defined as:

$$h(i) = \frac{n_i}{\sum_{j=0}^{L-1} n_j}, i = 0, 1, \dots, L-1$$

The additional features added to the feature vector T:

- Variance of gray value, which is the reflection of the discrete gray level of the pixel in the patch:

$$\sigma^2 = \sum_{i=0}^{L-1} (i-u)^2 h(i)$$

- Skewness, which reflects the asymmetry of gray level of pixels in patches. With skewness increasing, the middle of the gray histogram will be more asymmetric, relevantly ,the middle of the gray histogram will be more symmetrical when skewness getting smaller :

$$u_3 = \frac{1}{\sigma^3} \sum_{i=0}^{L-1} (i-u)^3 h(i)$$

- Kurtosis, which stands for the distribution of the patch where mean of pixel gray value is. The increase of amount of pixels at the mean value area could lead to the peak value being higher:

$$u_4 = \frac{1}{\sigma^4} \sum_{i=0}^{L-1} (i-u)^4 h(i) - 3$$

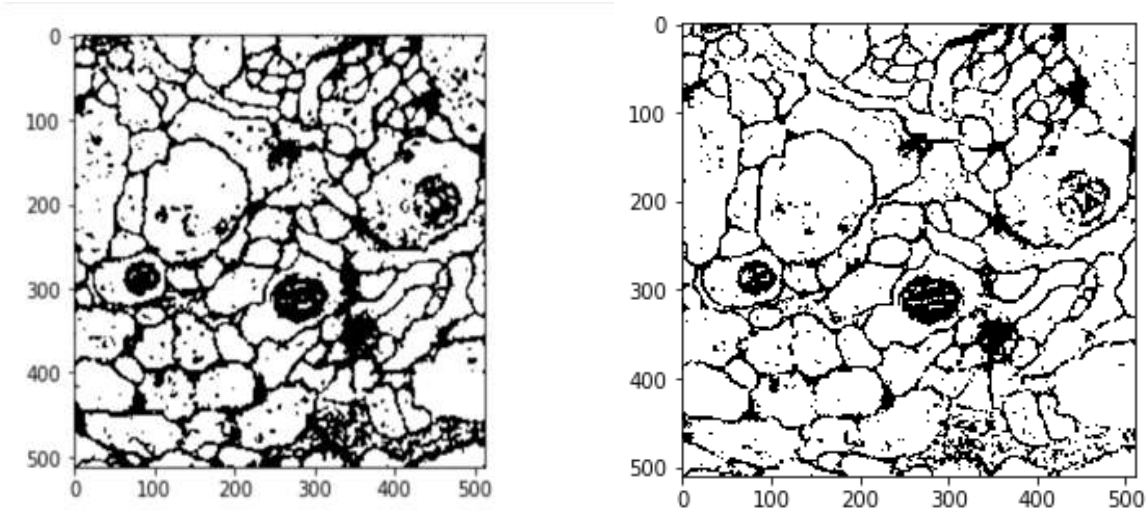


Figure 6. Left: Output of original method. Right: Output after adding new features.

An image is randomly selected, and compared with the results of utilising feature vector with length 24. After adding the new features, the accuracy has significant improvement: V^{rand} increased from 0.68 to 0.85. This maybe due to that after adding more useful features, the model's discriminative power of each feature extracted is improved.

LBP + Haralick + Random Forest

- LBP

Local Binary Pattern (LBP) describes the spatial structure of local image texture. It firstly divides the image into circular cells of a certain radius. Each pixel in a cell is then compared with its circular neighbours in clockwise/anticlockwise direction. If the neighbour pixel's value is greater than the centre value, write '0' otherwise '1'. This will produce an 8-digit binary number, which is then combined with all other binary numbers within the same cell. Next, a histogram can be generated to compute the frequency of each binary numbers occurred in this cell. Finally, the histograms from all cells are concatenated to obtain image-level descriptor.

The reason we chose LBP feature is that LBP can be insensitive to rotations and resolutions. From the image data provided, the neuron membranes may have different sizes and shapes with various angles. In addition, the electron microscopy images contain distinct patterns and textures, and these features can be easily captured and described by LBP.

- Haralick

The co-occurrence matrix is a matrix that is a distribution of co-occurring pixel values over an image at a given offset. The technique using co-occurrence matrices to measure the texture of the image is called Haralick. It is also capable of capturing textural properties.

Because the size of the patches is 11*11, in order to avoid the effect of the boundary pixels, the image size is decreased to 502*502 (i.e. the four corners: (5, 5), (5, 505), (505, 5) (505, 505)). Getting patches that centring each pixel in the cropped image and store them into a list. Randomly choosing 10000 numbers between 0 and 252004, which stands for the position in the list, extract the features of selected patches and store them into the feature list.

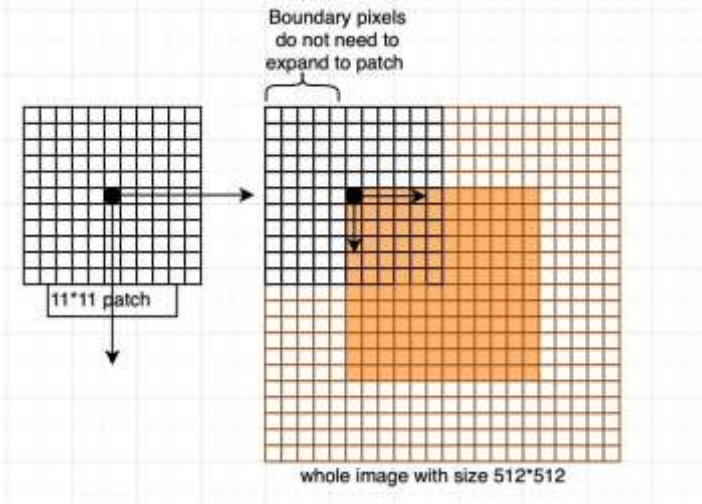


Figure 7. Explanation of patch selection for Haralick features.

1. Random Forest Concepts

Random Forest utilizes bootstrap aggregating to create an ensemble of decision trees. It combines all the optimal results from a group of decision tree classifiers. Each decision tree classifier only processes a portion of attributes of partial images set, where the images and their attributes are both selected randomly from the whole image set and the attribute set. The structure ensures each decision tree become a specialist of its own unique data set, and the final decisions are determined by the decisions made by the majority of the decision trees in the forest.

2. Procedure

The LBP feature was extracted using the built-in methods from Sk-Learn package. We mainly focused on two parameters needed for the feature extraction: radius of circle and number of neighbour points. The radius was chosen from 6 with an increment of 2, and the number of neighbour points is set to be the radius multiplied by 8.

The training data was originally set to be the LBP feature map, which is the same size as the original image. The data is then sent to the random forest classifier to train the model. However, the model is trained this way generated incorrect images. Hence, we combine the LBP feature map with the original image by stacking one on top of another and then feed the reshaped data (of size (512*512, 4)) and label (of size (512*512, 1)) into random forest classifier.

Next, we tried to add Haralick features into our training data to train a better model. The combined feature map was re-shaped to fit the Haralick feature size in order to combine them together.

In addition, it takes a tremendously long time to train with full-size images and features. Therefore, we implemented sub-sampling by randomly choosing a certain number of features (pixels) across the images and the corresponding indices from the labels (masks). The number of sub-samples is tested from 4000 to 15000, and the training time was significantly reduced while retained acceptable accuracy.

3. Post-processing

Firstly, each of the generated/predicted masks was blurred. Similar to Feature extraction with SVM, Gaussian filter was applied with kernel size 9x9 as we need not only to reduce noise but also to thicken the edges. This was accomplished by thresholding with a suitable threshold (from trial and error).

Then we apply morphological opening with kernel size 3x3 to combine the separated parts and further remove small noise points. Larger noise points that cannot be removed through previous procedures can be identified and then removed via blob detection. However, as there are also isolated parts that belong to the membrane, which will cause false-negative if all blobs are removed. Eventually, blob detection was not implemented in the algorithm.

4. Result

Image shown below is an example output of the random forest model with only LBP features:

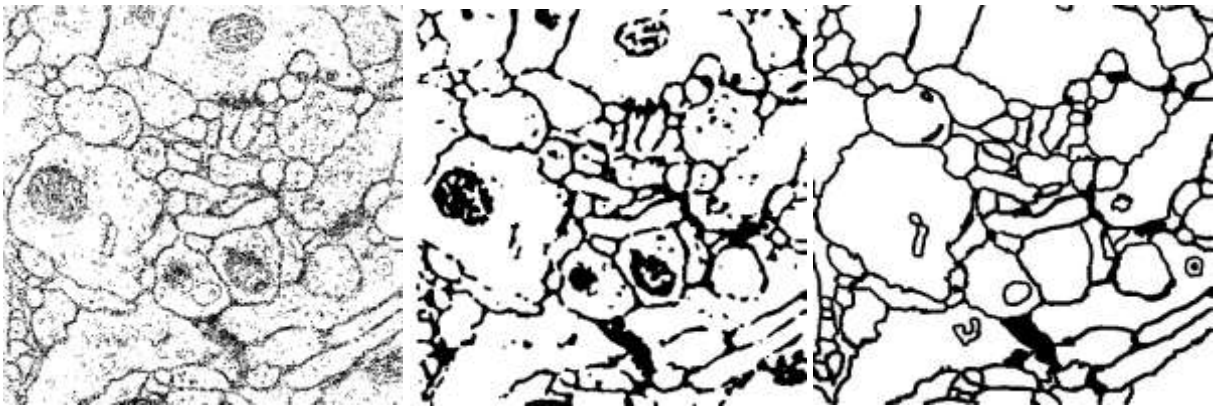


Figure 8. [1] Result from RF model. [2] Result after post-processing. [3] Ground truth.

Image shown below is an example output of the random forest model with LBP and Haralick features:

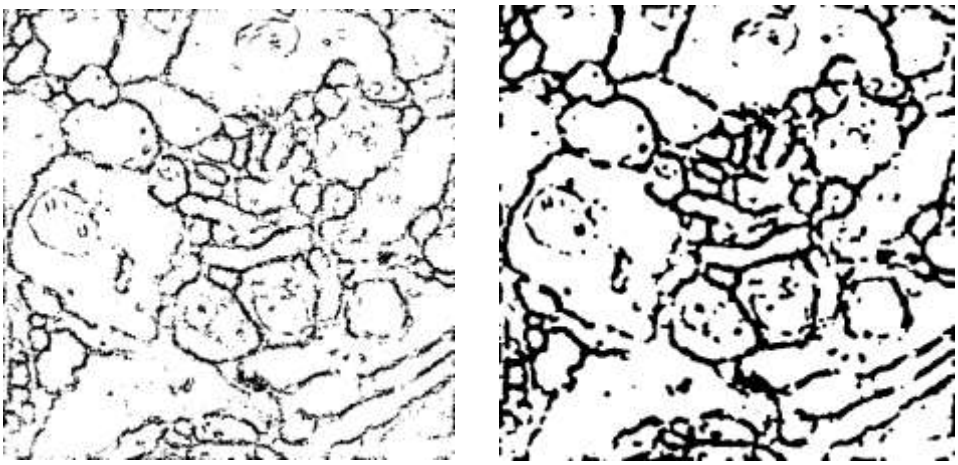


Figure 9. [1] Result from RF model. [2] Result after post-processing.

5. Evaluation

Both algorithms (with or without Haralick features) achieved acceptable results. By comparing the accuracy, it has insignificantly increased after adding Haralick features. This minor improvement was expected as more features were presented for each image, which could be beneficial for the training algorithms such as random forest.

	V^{rand}	V^{info}
LBP	0.521	0.788
LBP + Haralick	0.540	0.834

Table 2. Comparison of accuracies between algorithms.

However, it takes drastically longer to compute Haralick features such that it significantly increased the training time.

Deep Learning models

In recent years, researchers have shown that deep learning models can be used as tools in the field of image segmentation. In this report, two deep learning-based models are shown, namely U-net and ResUnet. Results of using both models are shown, and some modifications are made in order to improve model performance.

Image Augmentation

Similar to Machine Learning models, in order to train a powerful deep learning model, feeding the model with enough amount of data plays a significant role. Image augmentation is necessary for this problem because only 30 images and their corresponding image masks are given. The first object to solve this issue is to increase the dataset by augmenting the given images. Various image augmentation methods are used, which include rotation, width shift, height shift, shear, zoom, horizontal flip, fill and elastic transform. However, more data does not always guarantee better results. Deep learning model can also suffer from overfitting problems. Thus it is also essential to have a certain amount of data as a validate how well the model generalize (Thomson 2019)[2].

Below are some visualizations of the result after some of the image augmentation methods that are implemented:

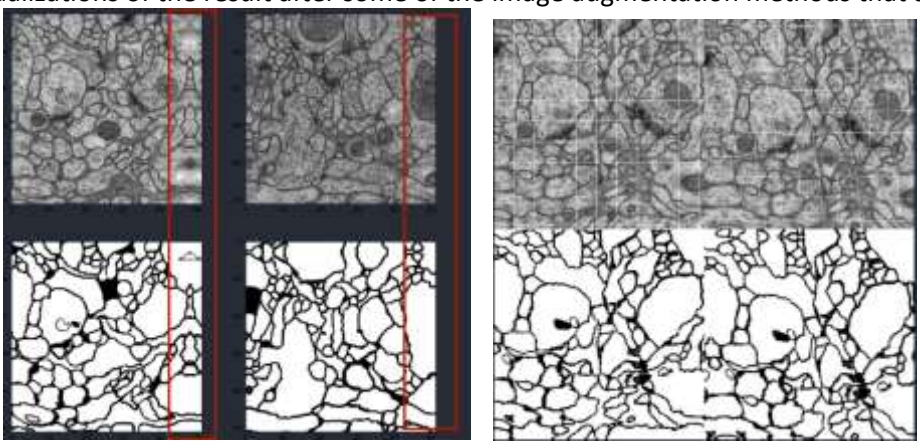


Figure 10. Left: Horizontal flipped Image and their masks. Right: Elastic Transformed Image and their masks.

However, implementing more data augmentation methods to the original image does not guarantee to produce better model performance. This report has shown two different combinations of different image augmentation methods. Firstly, rotation, width shift, height shift, shear, zoom, horizontal flip, and fill are used to augment the input image, and the results are displayed:

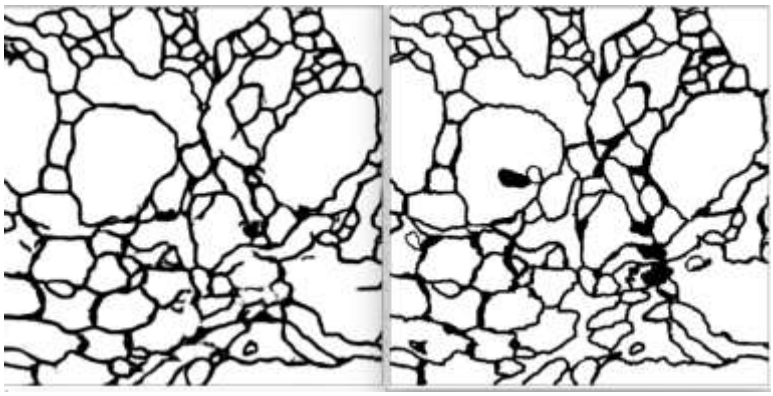


Figure 11. Left: Model prediction. Right: Image label.

However, when elastic transform is added to the above combination, the trained deep learning model performs worse, the results are shown:

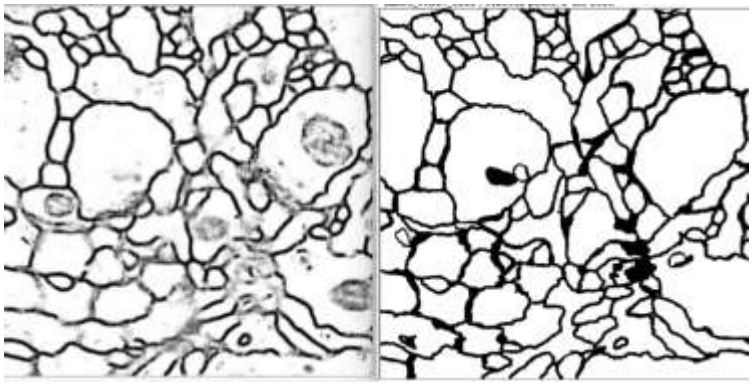


Figure 12. Left: Model prediction. Right: Image label.

Image Augmentation Methods	V^{Rand}	V^{Info}
rotation, width shift, height shift, shear, zoom, horizontal flip, fill	0.95	0.96
rotation, width shift, height shift, shear, zoom, horizontal flip, fill, elastic transform	0.82	0.91

Table 3. Comparison of evolution results by applying different image augmentation approaches.

From the table, the importance of choosing different image augmentation methods can be shown. For this project, the combination of rotation, width shift, height shift, shear, zoom, horizontal flip and fill outputs the best results.

Loss Function

Different loss functions are tried in order to examine their effects on model performance. In this report, two sets of loss functions are shown, namely binary cross entropy and binary cross entropy combined with dice loss.

Based on the given image masks, it is reasonable to apply a thresholding algorithm on them. By applying the algorithm, the image masks' pixel values will be a binary value (0 or 1). Then the problem can be treated as a binary classification issue. The loss function which is normally used for binary classification problems is the binary classification loss function:

$$E = \sum_{i=1}^{n} y_i \cdot \log\big(p(y_i)\big) + (1 - y_i) \cdot \log(1 - p(y_i))$$

By setting the model loss function as binary cross entropy, the learning curve is shown below:

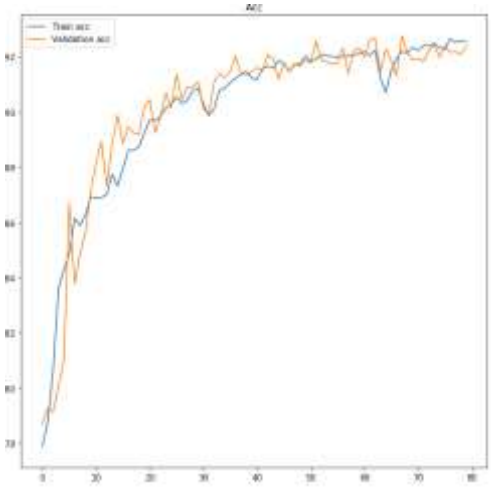


Figure 13. Learning curve with binary cross entropy as loss function.

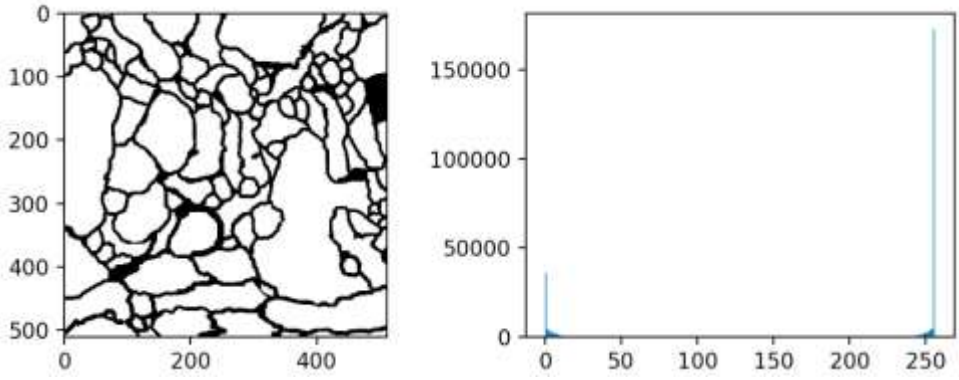


Figure 14. Image label and its histogram.

Then, by plotting the image histogram of given image labels, it can be noticed that the pixel classes are imbalanced, in order to design the model to tackle the problem, dice loss function is introduced:

$$E = 1 - \frac{\sum_{i=1}^{n} p_n * r_n}{\sum_{i=1}^{n} p_n + r_n}$$

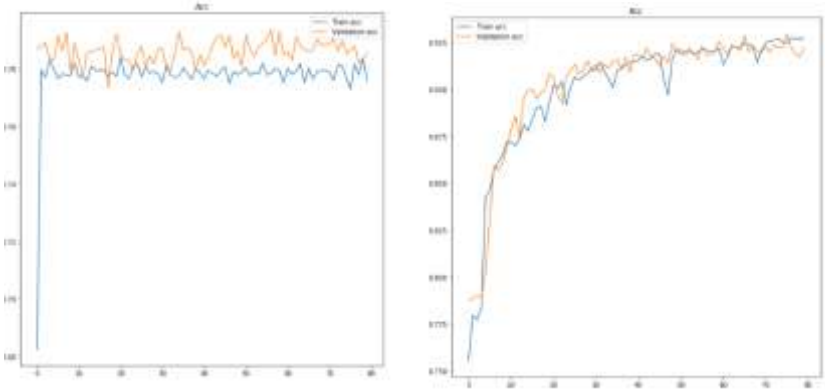


Figure 15. Left: Learning curve with dice loss as loss function. Right: Learning curve with combining binary cross entropy and dice loss.

From the plotted learning curves, combining binary cross entropy and dice loss as loss function shows better accuracy and a faster convergence rate.

Loss Functions	Accuracy[1] after 80-epochs
Binary Cross Entropy	0.9118
Dice Loss	0.7757
Binary Cross Entropy + Dice Loss	0.9253

Table 4. Comparisons between outputs with different loss functions.

Image Patching

The result of both models after training the deep learning models with images of full size (512 × 512) do not seem to be very promising. One improvement to our input might be decreasing the input image size. Instead of reshaping the input from (512 × 512) to a smaller size, we tried to divide the input image to smaller patches (256 × 256), then these smaller patches are trained. Moreover, test images are divided into smaller patches (256 × 256) with overlapping edges and the trained model will make predictions based on these divided patches. Image patching sampling methods in deep learning is similar to the one used on machine learning but with different number of patches.

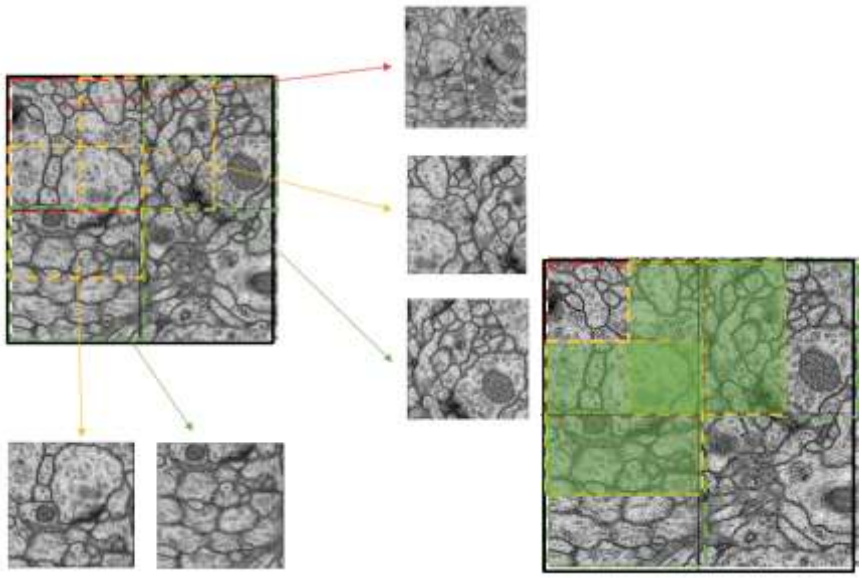


Figure 16. Demonstration of image patching and merging.

[1] Accuracy in the table is calculated based on a pixel-wise comparison.

Patch sampling is done as follows.

- Read the original image from images and labels.
- Split the image to 9 pieces with size 256x256, and most important of all is that patches should have overlapped edges.
- Patches are generated using a sliding window, the window size equals to patch size, and the window moves horizontally/ vertically with a step equals to half of the patch size.

So, for 30 images, we will get 270 pieces of patches. After training and prediction, predicted small patches are required to be merged together to form a whole image of the original size. However, when merging the 9 pieces into one image, directly concatenating them would cause some issue because of overlapped edges, so we take the average of the places which are overlapped and then do the merge. Merging operations are done horizontally then vertically. Both horizontal and vertical merging need to consider the overlapped areas.

Models

In this section, basic network structures of two deep learning models (U-net and ResUet) are descripted. Moreover, outputs from both networks and comparisons between them are shown. Furthermore, further improvements and some discussions are pointed out at the end of this section.

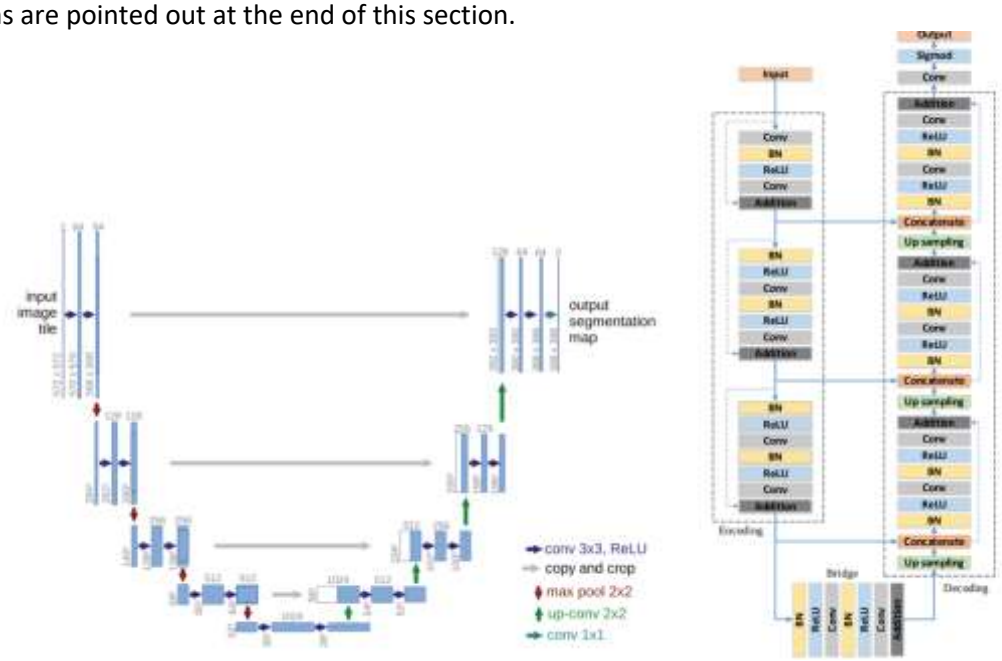


Figure 17. Left: U-net Structure. Right: ResUnet Structure.

U-net

According to Ronnerberger, Ficher and Brox (2015)[3], U-net model is constructed based on FCN model by adding upsampling layers successively to the downsampling layers, and these two parts form the two main parts of this U-net structure. Each downsampling layer consists of two convolution layers and one maximum pooling layer, input images after being down-sampled will have their channels doubled, and size halved. Whereas, the input to each upsampling layer is combined with output from the downsampling layers in order to localize. Furthermore, upsampling layer consists of one upsampling layer and two convolution layers. Input images after being upsampled will have their channels halved, and size doubled. Ronnerberger, Ficher and Brox (2015)[3] also assert that the point of implementing downsampling layers is to extract image features. On the other hand, by developing upsampling layers, image details can be retained.

The implemented U-net model for this project is constructed based on Ronnerberger, Ficher and Brox (2015)[3] 's original network structure. However, we have made two changes to the model, and they perform differently. 1. Changing the model to accept inputs of size (512 × 512) and output the image of the same size (512 × 512). 2. Divide the image into (256 × 256) patches and train model using image patches.

ResUnet

The ResUnet model used in this project is constructed based on Diakogiannis, Waldner, Caccetta and Wu (2019)[4] 's work. Using the constructed U-net model, all max-pooling layers in the downsampling layers are replaced by an addition layer which combines the layer output with layer input. Moreover, one batch normalization layer is added in between of two convolution layers in the first downsampling layer, and an additional batch normalization layer is added at the beginning of other downsampling layers. Similar operations can be done for the upsampling layers.

The basic idea of adding batch normalization layer and addition layer is would be to ensure the performance of the model with a deeper layer. On the one hand, it is stated by Olafenwa (2018)[5] that implementing batch normalization can effectively normalize the input and ensure that convolutions always take normalized inputs. On the other hand, he also points out that by adding the identity connection, deeper models are able to perform at least the same as shallower models.

Model Outputs and Comparisons

Parameter tuning

Training Data (%)	Testing Data(%)	Number of Epoch	U-net Model Accuracy[2]		ResUnet Mode Accuracy	
			Train	Validation	Train	Validation
80	20	80	0.9232	0.9373	0.9316	0.9319
80	20	160	0.9393	0.9342	0.9443	0.9369
80	20	240	0.9474	0.9304	0.9508	0.9366
70	30	80	0.9344	0.9334	0.9258	0.9304
70	30	160	0.9412	0.9315	0.9419	0.9386
70	30	240	0.9521	0.9364	0.9482	0.9357
60	40	80	0.9272	0.9218	0.9287	0.9170
60	40	160	0.9401	0.9206	0.9488	0.9341
60	40	240	0.9514	0.9330	0.9511	0.9306

Table 5. Model accuracy comparison with different parameter settings.

From the table, it can be noted that the highest accuracy on validation sets without overfitting (0.9241) of U-net model can be reached when setting 80% of image data as the training set, 20% as the validation set and 80 epochs. ResUnet model' optimal performance can be reached by using the same setting (0.9319). By testing the model with different settings, it can be found that the model tends to be overfitting when it is trained with a higher number of epochs.

Learning curves and model outputs are shown below:

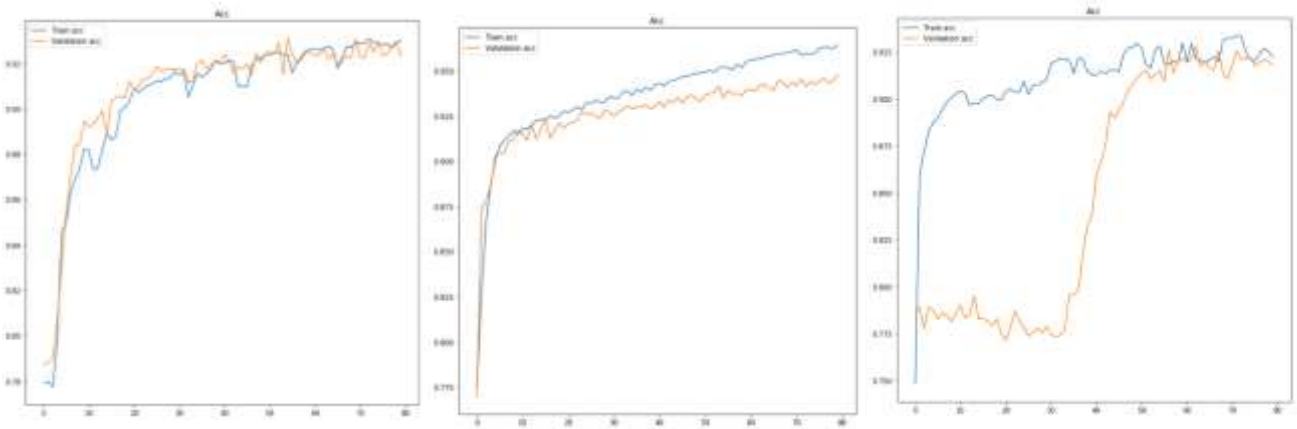


Figure 18. Left: U-net Learning Curve. Middle: Unet with Patch Sampling. Right: ResNet learning curve.

[2] Accuracy in the table is calculated based on a pixel-wise comparison.

Model Prediction (Image01):

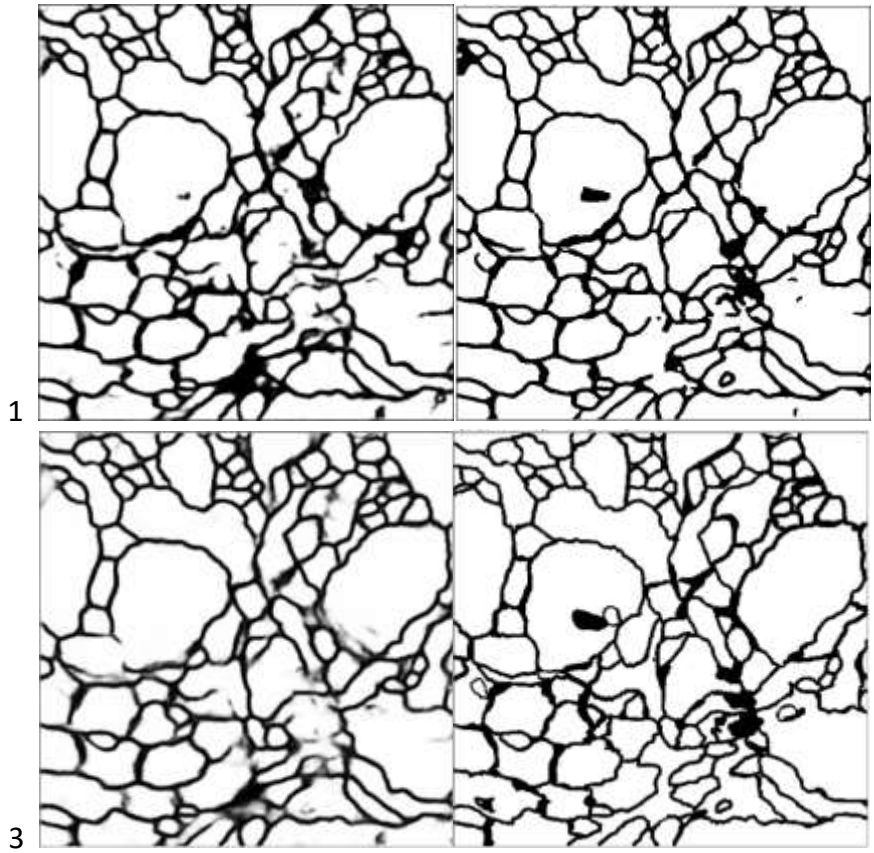


Figure 19. 1:U-net result. 2:U-net with Patch Sampling. 3: ResUnet result. 4: Ground truth.

It can be seen that ResUnet has a slower convergence rate, but its final accuracy is very similar to U-net, and this may be caused by the increased model complexity of ResUnet. Moreover, adding patch sampling can increase the prediction performance of some detail that can't be trained when we use the whole picture, and train time can be decreased.

Models	V^{Rank}	V^{Info}
U-Net	0.973	0.980
U-Net with Image Patch	0.977	0.987
ResUnet	0.936	0.975

Table 6. Result comparison between deep learning models.

Why U-net?

Under this particular circumstance, the given image sets are medical images. Medical images have some properties, such as simple semantics and stable structures. These properties make U-shape models with skip connections perform relatively well. Because these structures are able to store restore the image to the original shape while retaining its details.

Conclusion

For this project, an image segmentation task is solved using developed machine learning-based approaches and deep learning-based approaches. By comparing different results, it can be found that machine learning-based approaches have the advantages that they are simpler in theory and usually requires less time to train. However, their performance is worse than deep learning models. On the other hand, deep learning-based approaches outperform machine learning-based approaches in the area of computer vision. After implementing deep learning models, a reasonably good result can be obtained. But training a deep learning model usually requires higher computational power and more time-consuming. Besides, tuning parameters of deep learning models is also a complicated job.

Reference:

[1]. Liu, P 2018, 'Research on Machine Learning Based Sequential Ultrasound Bovine Follicular Image Set Segmentation Algorithm'

[2]. Thompson, W 2019, 'Deep Learning: The Confluence of Big Data, Big Models, Big Compute' , accessed 8 August 2019, < https://www.datanami.com/2019/01/10/deep-learning-the-confluence-of-big-data-big-models-big-compute/>.

[3]. Ronneberger, O., Fischer, P., Brox, T. 2015, 'U-net: convolutional networks for biomedical image segmentation'.

[4]. Diakogiannis,I., Waldner, F., Caccetta,P., Wu C. (2019) 'ResUNet-a: a deep learning framework for semantic segmentation of remotely sensed data', accessed 9 August 2019, <https://arxiv.org/pdf/1904.00592.pdf>

[5]. Olafenwa, J. 2018, 'Review of Identity Mappings in Deep Residual Networks' <https://medium.com/deepreview/review-of-identity-mappings-in-deep-residual-networks-ad6533452f33>

Appendix

Name	Student ID	Contribution
Cong Cong	z3414050	U-net, ResUnet
Fan, Liu	z5188300	Patch Sampling
Xiao Tan	z3413898	LBP feature with Random Forest
Guanqun Zhou	z5174741	Feature extraction with SVM
Zixin Fang	z5210737	Haralick feature with Random Forest