# Deep Learning models

In recent years, researcher have shown that deep learning models can be used as tools in the field of image segmentation. In this report, two deep learning-based model 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 as 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 transformHowever, more data does not always guarantee better results, deep learning model can also suffer from overfitting problems, thus it is also important to have certain amount of data as a validate how well the model generalize (Thomson 2019).

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

A picture containing building, window

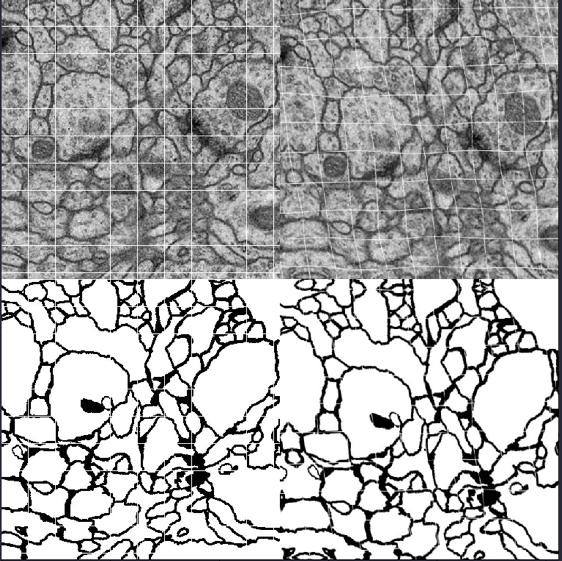
Description automatically generated 

Fig1. horizontal flipped Image and their masks Fig2. 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:

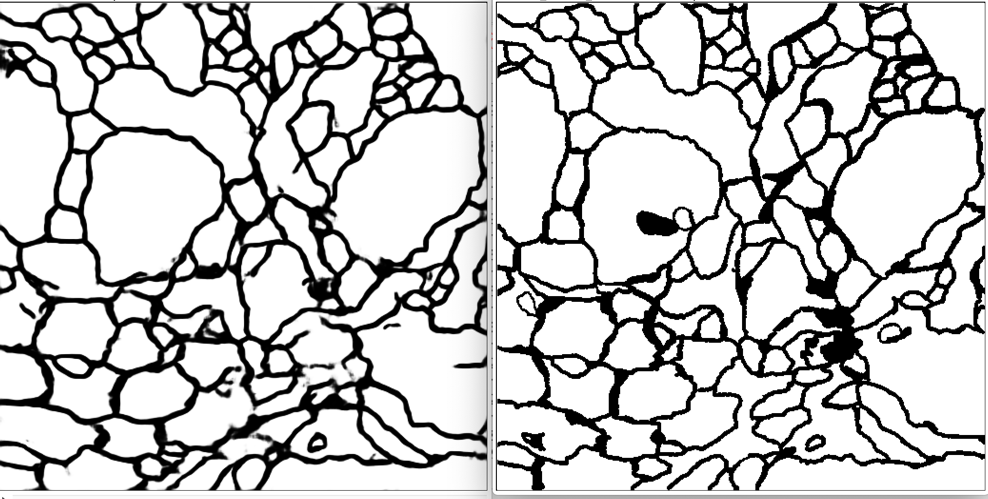


Fig3. Model prediction(left) and given label(right)

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

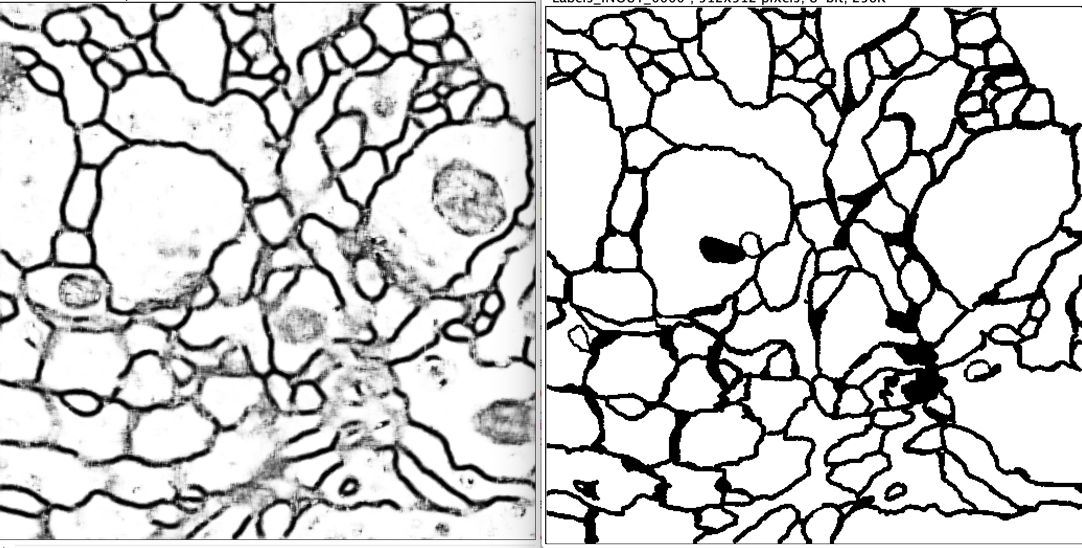


Fig4. Model prediction(left) and given label(right)

|  |  |  |
| --- | --- | --- |
| Image Augmentation Methods |  |  |
| 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 |

Table1. Comparison of evolution results by applying different image augmentation approaches

From the table, the importance of choosing different image augmentation methods can be shown and in this case, 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. In this report two set 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:

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

A close up of a map

Description automatically generated

Fig5.Learning curve with binary cross entropy as loss function

A close up of a logo

Description automatically generated

Fig5.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:

The figure on the left shows the learning curve of having only dice loss as loss function, it can be seen that the overall accuracy is low. Figure on the right shows the learning curve of combining binary cross entropy and dice loss, the results shows a better accuracy and a faster convergence rate.

A screenshot of a cell phone

Description automatically generatedA close up of a map

Description automatically generated

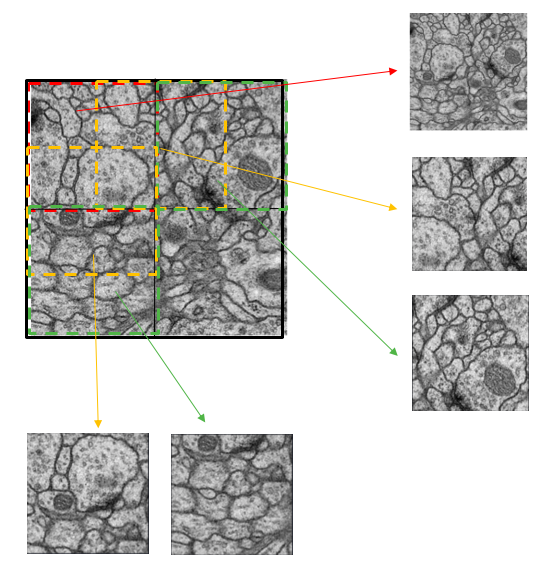
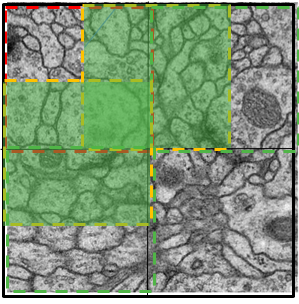
Fig5.Learning curve with dice loss as loss function; Learning curve with combining binary cross entropy and dice loss

Table2. Comparisons between outputs with different loss functions

|  |  |
| --- | --- |
| Loss Functions | Accuracy after 80-epochs |
| Binary Cross Entropy | 0.9118 |
| Dice Loss | 0.7757 |
| Binary Cross Entropy + Dice Loss | 0.9253 |

## 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 prediction based on these divided patches.

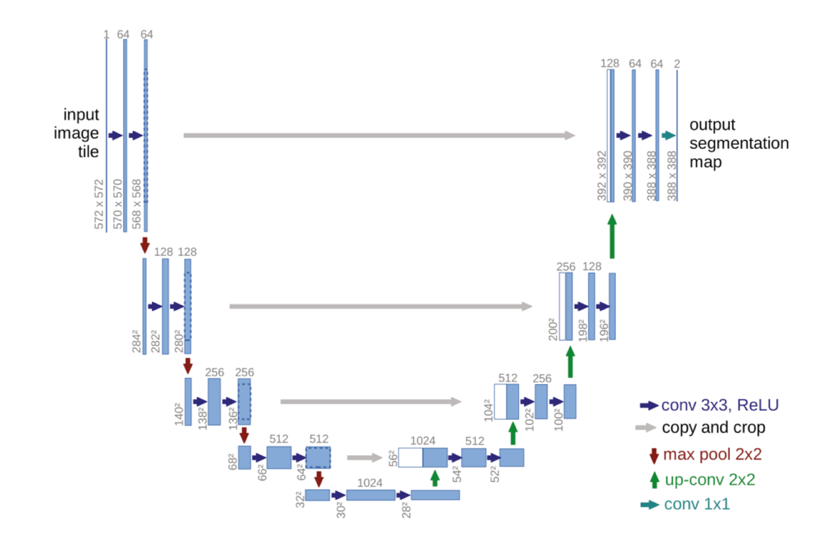
Patch sampling is done as following.

* Read the original image from images and labels.
* Split the image to 9 pieces with size 256 \* 256, and most important of all is that patches should have overlapped edges.
* Patches are generated using sliding window, the window size is 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 image. 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, concatenating them directly 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 horizonal 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.

 A screenshot of a cell phone

Description automatically generated

Figure8.U-net Structure Figure5. ResUnet Structure

***U-net***

According to Ronnerberger, Ficher and Brox (2015), 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 is consisted 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 is consisted of one upsampling layer and two convolution layers. Input images after being up sampled will have their channels halved and size doubled. Ronnerberger, Ficher and Brox (2015) 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) ‘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) ‘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 deeper layer. On the one hand, it is stated by Olafenwa (2018) that implementing batch normalization can effectively normalized 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

Table1. Model accuracy comparison with different parameter settings.

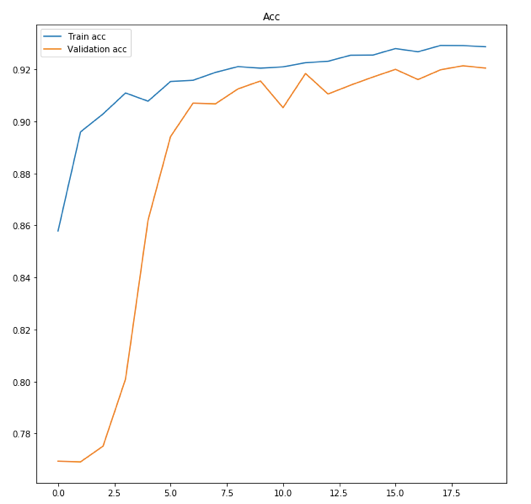
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Training Data (%) | Testing Data(%) | Number of Epoch | U-net Model Accuracy | | ResUnet Mode Accuracy | |
| Train | Validation | Train | Validation |
| 80 | 20 | 80 | 0.9232 | 0.9373 | 0.9316 | 0.9319 |
| 80 | 20 | 160 | 0.9393 | 0.9342 | 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 |

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 training set, 20% as validation set and 80 epochs. ResUnet model’ optimal performance can be reached by using the same setting (0.9319). By testing model with different settings, it can be found that model tends to be overfitting when it is trained with higher number of epochs.

Learning curves and model outputs are shown below:

Learning curves:

A close up of a map

Description automatically generated

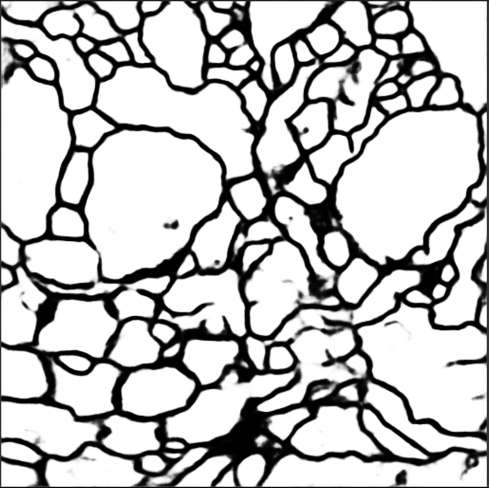
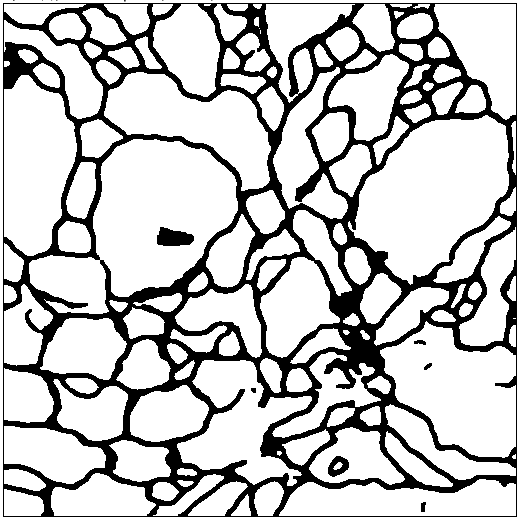
U-net Learning Curve Unet with Patch Sampling

A close up of a map

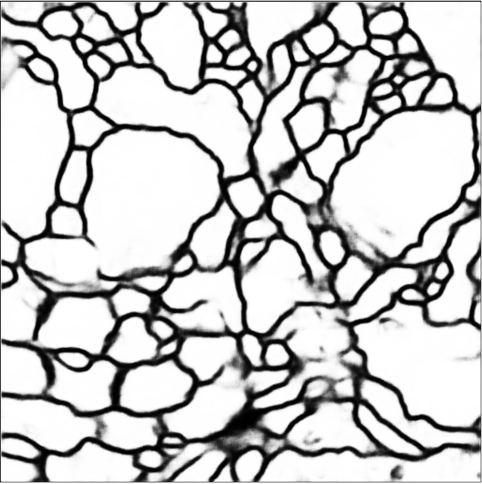
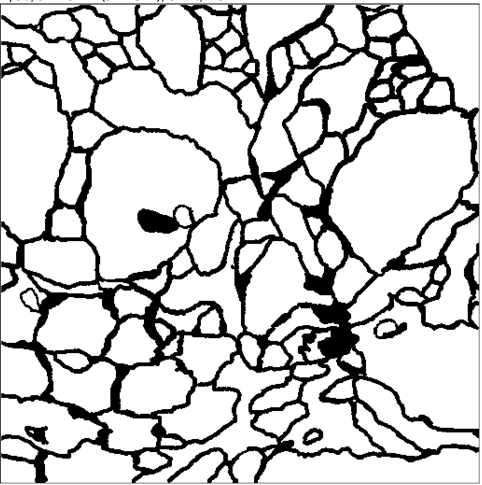
Description automatically generated

ResNet Learning Curve

Model Prediction (Image01):

U-net U-net with Patch Sampling

ResUnet Ground Truth

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

|  |  |  |
| --- | --- | --- |
| Models |  |  |
| U-Net | 0.973 | 0.980 |
| U-Net with Image Patch | 0.977 | 0.987 |
| ResUnet | 0.936 | 0.975 |

### 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 retain 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 implemening deep learning models, 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:

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/>>.

Ronneberger, O., Fischer, P., Brox, T. 2015, ‘U-net: convolutional networks for biomedical image segmentation’.

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>>

Olafenwa, J. 2018, ‘Review of Identity Mappings in Deep Residual Networks’

<https://medium.com/deepreview/review-of-identity-mappings-in-deep-residual-networks-ad6533452f33>