Electron Microscopy Image Segmentation

Wu Shiqu 518021910665
Hou Shengyuan 518021910604
Yan Binghao 518021910753

Abstract. Image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics. In this project, we try to solve the problem in ISBI(2012) challenge where a full stack of EM slices will be used to train machine learning algorithms for the purpose of automatic segmentation of neural structures. We tried two models (U-Net and U-Net++), and also conducted corresponding experiments and comparisons. Experimental results shown that Our Unet model achieves 0.934 for V_{rand}, 0.923 for *accuracy* and 0.978 for V_{info} on five testing figures, and our Unet++ model achieves 0.454 for V_{rand}, 0.908 for *accuracy* and 0.831 for V_{info}. Programming code is available on github[1].

Keywords: Image segmentation, ISBI, U-Net, U-Net++

1 Introduction

The images used in the ISBI (2012) challenge represent actual images in the real world, containi ng some noise and small image alignment errors. None of these problems led to any difficulties in the manual labeling of each element in the image stack by an expert human neuroanatomist. The aim of the challenge is to compare and rank the different competing methods based on their pixel and object classification accuracy.

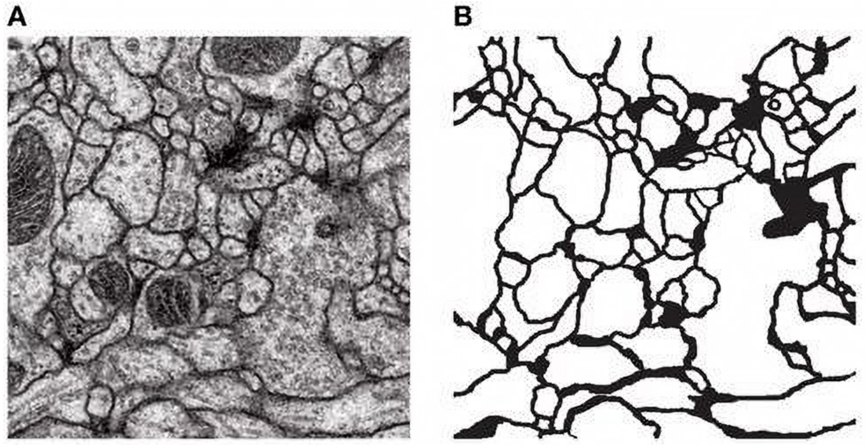


Fig. 1: (a)Grayscale EM image;(b)Sccurate boundary map

The accuracy of segmentation is very important for medical images, because the edge segmentation error will lead to unreliable results, in that case it will be rejected for clinical application.

[1] https://github.com/BrianWU-S/electron_microscopy_image_segmentation

Obtaining these sample images to train the model may be a resource consuming process because of the need for high-quality, uncompressed and accurately annotated images reviewed by professionals. Therefore, the algorithm designed for medical imaging must achieve high performance and accuracy with less data samples. In addition, border detection is challenging because many borders look blurry. In addition, only the boundaries between neurites should be detected, and the boundaries of intracellular organelles such as mitochondria and synaptic vesicles should be ignored.

Considering the above factors, we finally selected two models of U-Net, U-Net++ for experimentation, in order to convert the gray-scale EM image (Fig.2 .a) into an accurate boundary map (Fig.2 .b), and achieve higher accuracy rate.

2 Related work

2.1 U-Net

U-Net [1] is a convolutional neural network that is popular in medical image segmentation. The network is based on Fully Convolutional Network (FCN), and its architecture has been modified and expanded to use fewer training images and produce more accurate segmentation. On modern GPUs, the segmentation time of a 512 $\times$ 512 image is less than one second.

The network consists of a contraction path and an expansion path, which gives it a U-shaped architecture. The contraction path is a typical convolutional network, which consists of repeated applications of convolutions, each of which is followed by a rectified linear unit (ReLU) and a maximum pooling operation. During the contraction, the spatial information decreases and the feature information increases. The expansion path combines features and spatial information with high-resolution features from the contraction path through a series of up convolutions and concatenations.

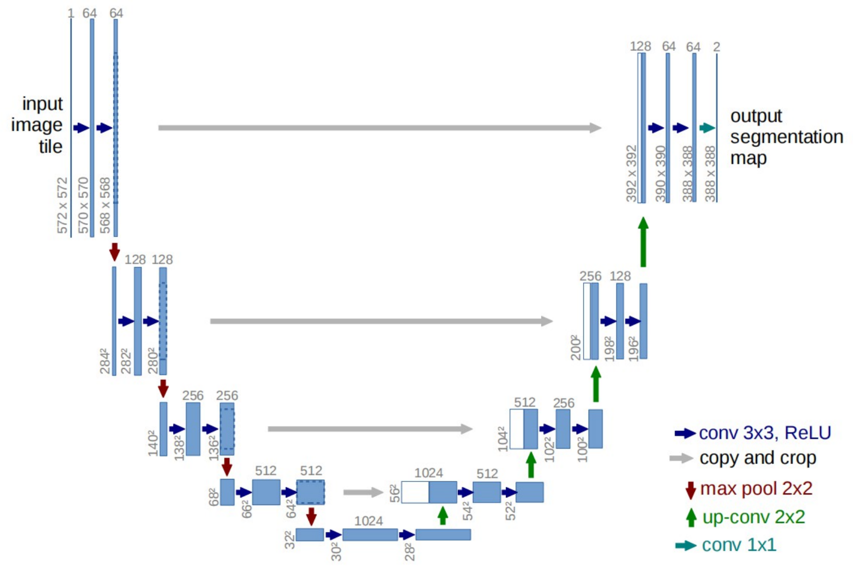


Fig. 2: U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

U-Net can be divided into three main parts: down-sampling, up-sampling and Skip-connection.

- **Down-sampling:** It can increase the robustness to some small disturbances of the input image (such as image translation, rotation, etc), reduce the risk of overfitting, reduce the amount of calculation, and increase the size of the receptive field.
- **Up-sampling:** Its biggest function is actually to restore and decode the abstract features to the size of the original image, and finally get the segmentation result.
- **Skip-connection:** As the receptive field gradually becomes smaller during the down-sampling process, the model focuses on different features. For classification problems, only deep features need to be paid attention to, so it is an FCN structure; however, for image semantic segmentation, shallow features and deep features are equally important. Therefore, by adding Skip-connection, U-net facilitates the up-sampling of shallow extracted features.

It is worth mentioning that the skip connection of U-net is a long connection (U-net++ in the following text uses a short connection). Its advantage is that it has a simple structure and can efficiently spread the features extracted from shallow layers. Its disadvantage is that the feature superposition method is relatively simple and cannot perform deep-level feature extraction and superposition. Unet++, which we mentioned later, is an effective improvement on this problem.

2.2 U-Net++

U-Net++ [2] expands on the basis of U-Net, which improves the segmentation accuracy by adding a dense block and a convolutional layer between the encoder and the decoder. The main improvements can be divided into the following three parts:

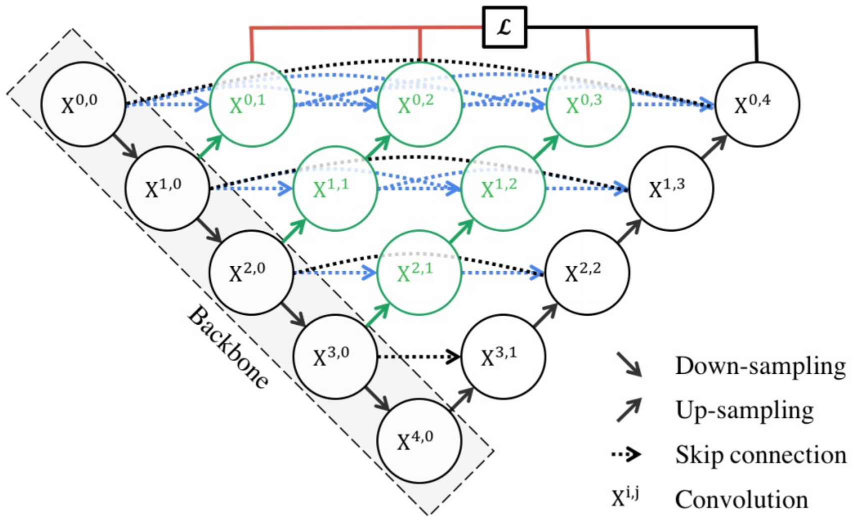


Fig. 3: UNet++ structure diagram

- **Redesigned jump path(Fig.3 Green part):** In U-Net++, the output of the previous convolutional layer of the same dense block is fused with the up-sampling output corresponding to the lower layer of the dense block. This makes the semantic level of the encoded feature closer to the semantic level of the feature map waiting in the decoder, so when a semantically similar feature map is received, optimization is easier.

- **Dense jump connection(Fig.3 Blue part):** Dense jump connection ensures that all prior feature maps are accumulated and reach the current node through the dense convolution block on each jump path. This will generate full resolution feature maps at multiple semantic levels, thereby improving segmentation accuracy and improving gradient flow.
- **In-depth supervision(Fig.3 Red part):** This is to facilitate the adjustment of model complexity by trimming the model, and to achieve a balance between speed (inference time) and performance. The specific details of trimming will be expanded in detail below.

In addition to the above improvements, the structure of U-Net++ also allows the model to be pruned, thereby reducing the amount of model parameters. For example, in Fig.4, $UNet++L^3$ prunes the rightmost backbone part compared to $UNet++L^4$. In the test phase, since the input image will only be forwarded, the pruned part has no effect on the previous output; in the training phase, because the model is forwarded and then back-propagated, it is cut off The part of will help other parts to update the weights, so we only perform pruning during the testing phase and maintain the original network structure during the training phase.

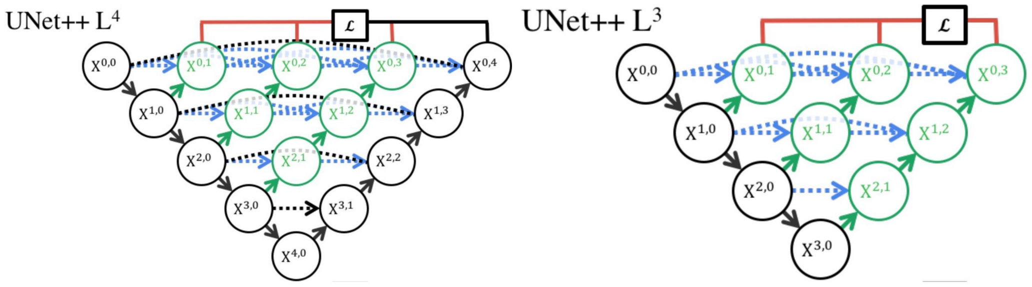


Fig. 4: UNet++ can be pruned at inference time, if trained with deep supervision.

Considering that in the process of deep supervision, the output of each sub-network is actually the result of image segmentation, so if the output of the small sub-network is good enough, we can cut off those extra parts to reduce the model's The amount of parameters. This is of great benefit to mobile applications. Small model parameters can not only save storage space, but also speed up model prediction. The experimental results of the original paper show that the performance of $UNet++L^2$ is almost the same as that of $UNet++L^4$, but its test speed is three times that of the latter, and the amount of parameters of the model is only $\frac{1}{16}$ compared with the latter.

3 Experiment

3.1 Data set and Experimental settings

The data description is same with ISBI Challenge [3] except that we split the raw train data set (consist of 30 samples) into two parts: the new train set and new test set. The downloaded data set consists of 30 samples, 25 for train and 5 for test. We simply train our model on the newly split data sets and did not use pre-training models.

In addition, since the training data size is too small (only 25 training images), which will brings about the overfitting of model, We use the **ImageDataGenerator** function in the $keras.preprocessing.image$ library for image augmentation. Basic operations include rotation, shear, zoom ,horizontal flip and vertical flip(as shown in table 1), the augmentation effect is shown in Fig 5. After that, we get an augmented dataset of 10000 images.

operation	value range
rotation	± 20
shear	± 0.1
zoom	± 0.1
vertical_flip	True
horizontal_flip	True

Table 1: Augmented settings

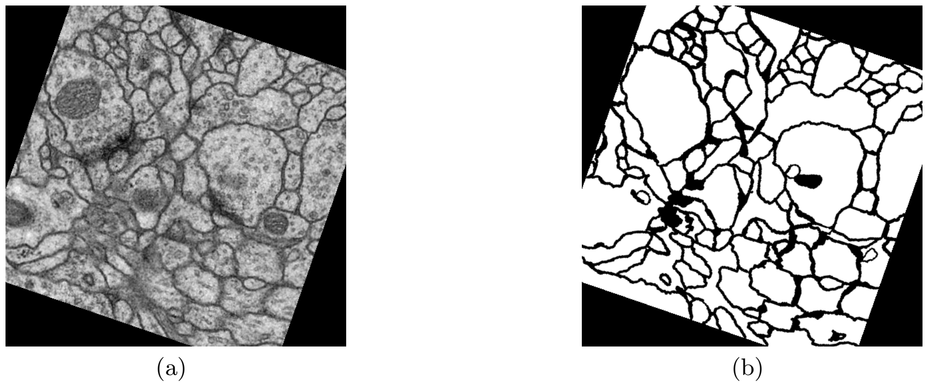


Fig. 5: Augmented images with its label

For training process, we take advantage of two loss functions: one is cross-entropy and the other one is dice loss.

$$BCE = -\sum_i (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i))$$

$$DiceLoss = 1 - \frac{2 * |X \cap Y|}{|x| + |Y|}$$

Binary crossentropy loss is sensitive to the class imbalance problem, this could be solved by dice loss. But dice loss is also easy to be trapped into overfitting. To make the best use of two loss functions we integrate them into the following form.

$$BCE\_dice\_loss = 0.5 * BCELoss(g, p) - diceloss(g, p)$$

3.2 Evaluation Metrics

In this experiment, we take three evaluation metrics as our standard: pixel-level accuracy, V_rand and V_info. For the accuracy, it is defined as the following. However, this metrics fails from trivial disturbance on the location of boundary, which brings about the lackness of robustness.

$$acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Therefore, we incorporate other two metris V_rand [4] and V_info [5] that were taken in ISBI challenge dataset. Suppose that S is the predicted segmentation and T is the ground truth segmentation. Then define p_{ij} as the probability that a randomly chosen pixel belongs to segment i in S and segment j in T. This is a joint distribution and thus could be normalized to 1 $\sum_{ij} p_{ij} = 1$. Also we could calculate the marginal distribution $p_s = \sum_j p_{sj}$ and $p_t = \sum_i p_{it}$. To reduce the splitting error, we define the V^{split}_{rand} and to reduce the merging error, we define the V^{merge}_{rand}.

$$V^{split}_{rand} = \frac{\sum_{ij} p^2_{ij}}{\sum_t p^2_t}$$

$$V^{merge}_{rand} = \frac{\sum_{ij} p^2_{ij}}{\sum_s p^2_s}$$

Then for the final score that take both metrics into consideration, we take weighted harmonic mean as final rand index. The parameter α is set as 0.5 here for experiments. Traditionally, for segmentation results we often regard the connected components of "1" as segments and every single "0" pixel as a single segment containing just one pixel. Additionally, border pixels in ground-truth image should be excluded from computation. However, for simplicity here, we ignore these two details and directly regard every "1" region as a segment.

$$V^{rand}_{\alpha} = \frac{\sum_{ij} p^2_{ij}}{\alpha \sum_k s^2_k + (1-\alpha) \sum_k t^2_k}$$

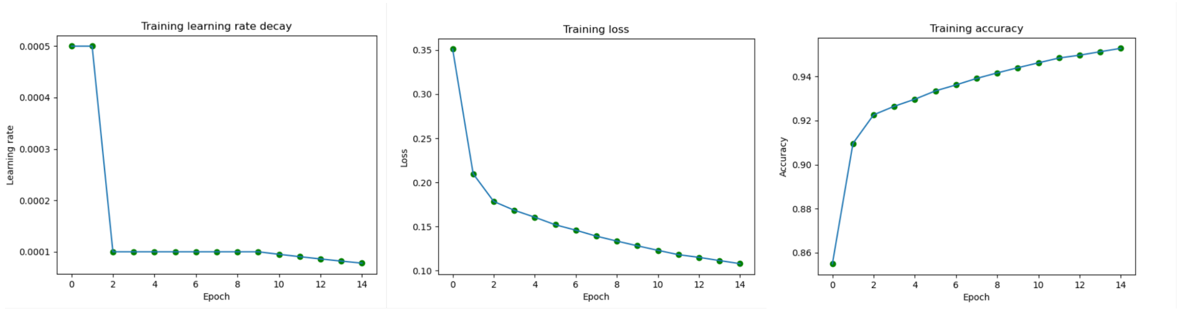
3.3 Unet

Here we list our parameters of network architecture and training process in Table 2 and experimental results are shown in Fig 7.

Parameters	Value
epoch	10
initial LR	5e-4
steps	300
batch size	1
dropout rate	0.5
optimizer	Adam
conv channel(encoder)	[64,128,256,512,1024]
conv channel(decoder)	[512,256,128,64,2,1]
pooling size	(2,2)
padding	same
activation function	relu
loss function	binary cross entropy loss

Table 2: Parameters of Unet and training process

During the training process, we take Adam optimizer to perform gradient descent on training dataset. Adam optimizer could be regarded as a combination of momentum and RMSprop and benifits from low memory requirements and adaptive different learning rate for different parameters. To avoid overfitting, we add dropout layer with dropout rate 0.5 into the border between encoder and decoder. Additionally, we adapt learning rate decay technique to better fit the training process implemented in $Unet\_scheduler$ function in $utils.py$. Within 2 epoch, the leanring rate is set as default value. When it comes between 2 and 10 epoches, we fix the leanring rate as $1e-4$ and When it comes more than 10 epoches, we fix it as $lr * \exp(-0.05)$. Changes of leanring rate is shown in fig 6(a). For simplicity, we directly take binary cross entropy as our loss function(training loss and accuracy is shown in fig 6(b) and 6(c)) and more broader trials will be discussed detailedly in Unet++ part.



(a) leanring rate during training (b) loss during training (c) accuracy during training

Fig. 6: Training process

After inspection, we could find that although for a few testing figures the performance w.r.t V_rand is not such satisfactory, the overall performance on table 5 appears to be relatively higher, which comes near to the topest 20 teams in IBSI2012 challenge. However, we think this result is still not so consuming for the reason that the lackness of testint data increases the variance of model performance. In pixel level, the accuracy often fluctuates up and down among 0.92 and is relatively stable. This is because pixel level measurement does not regard a segment as an ensemble and this metrics compares purely the similarity between two figures.

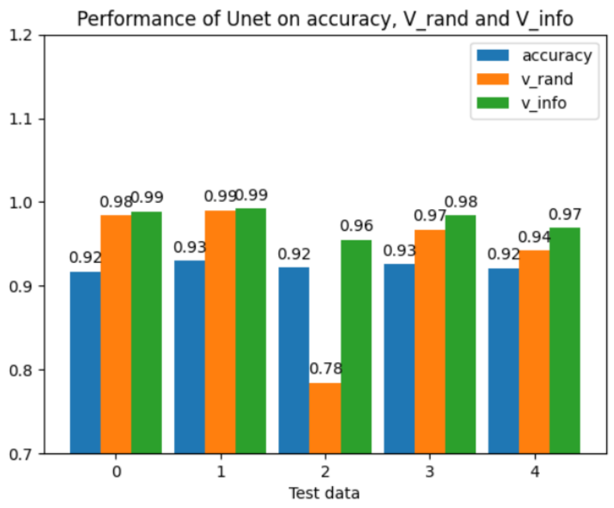


Fig. 7: Performance of Unet on three metrics

Accuracy	V_rand	V_info
0.923	0.934	0.978

Table 3: Average Performance on Three Metrics

Also, from the bar chart above, we could find that V_rand metrics which is not so stable as other two attain a lower value on test set 1 and test set 2. Here we paste label figures on test label 2 and predictive output 2 and compare them in detail8. After comparison, we could find that for most of the boundary map, the Unet model does a really satisfactory job, nevertheless, we could find that our predictive boundary image loses a huge amount of independent little areas falling within another one larger area. Additionally, from location below the picture, blocks of boundary area is cut into many pieces which increase the splitting error and since V_rand score calculates the harmonic mean of V_{rand}^{split} and V_{rand}^{merge}, any insufficient performance between two metrics will bring about the descend of final evaluation result. And this could also explain why the evaluation result is not so stable on testing dataset.

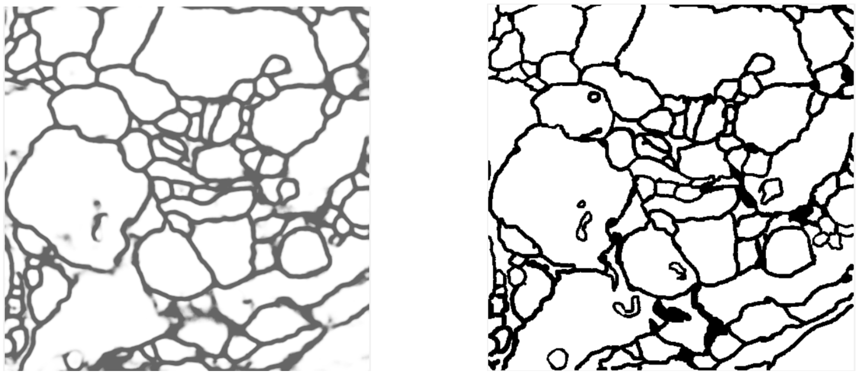

(a) *Predict output* (b) *Label*

Fig. 8: Comparison of test label and output on figure 2

3.4 U-Net++

Here we list our parameters of network architecture and training process in Table 4. Most of the hyperparameter setttings in U-net++ are identical to Unet model. Therefore, here we do not repeat the detailed description here for optimization process anymore. Otherwise, we will try differemt loss functions and data preprocessing technique and compare the performance in the following two parts(data enhancement and loss function).

Under this parameter settings, we explore the performance of U-net++ model and experimental results are shown in fig 9.

Parameters	Value
epoch	15
LR	5e-4
steps	300
target size	(512,512)
batch size	1
dropout rate	0.5
optimizer	Adam
conv channel(encoder)	[32,64,128,256,512]
conv channel(decoder)	[256,128,64,32,1]
pooling size	(2,2)
padding	same
activation function	relu

Table 4: Parameters of Unet and training process

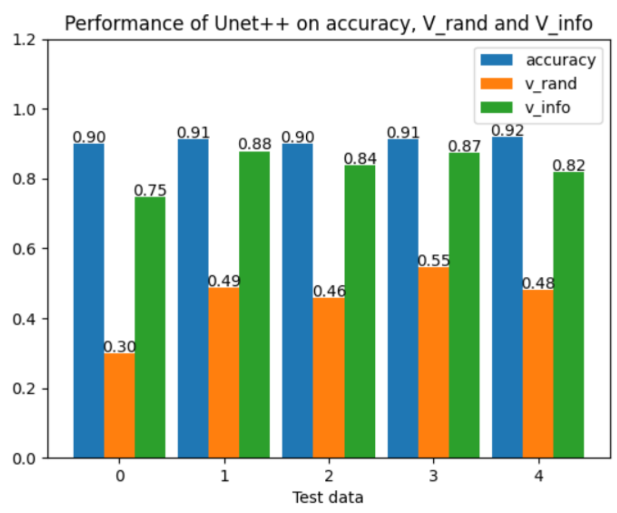


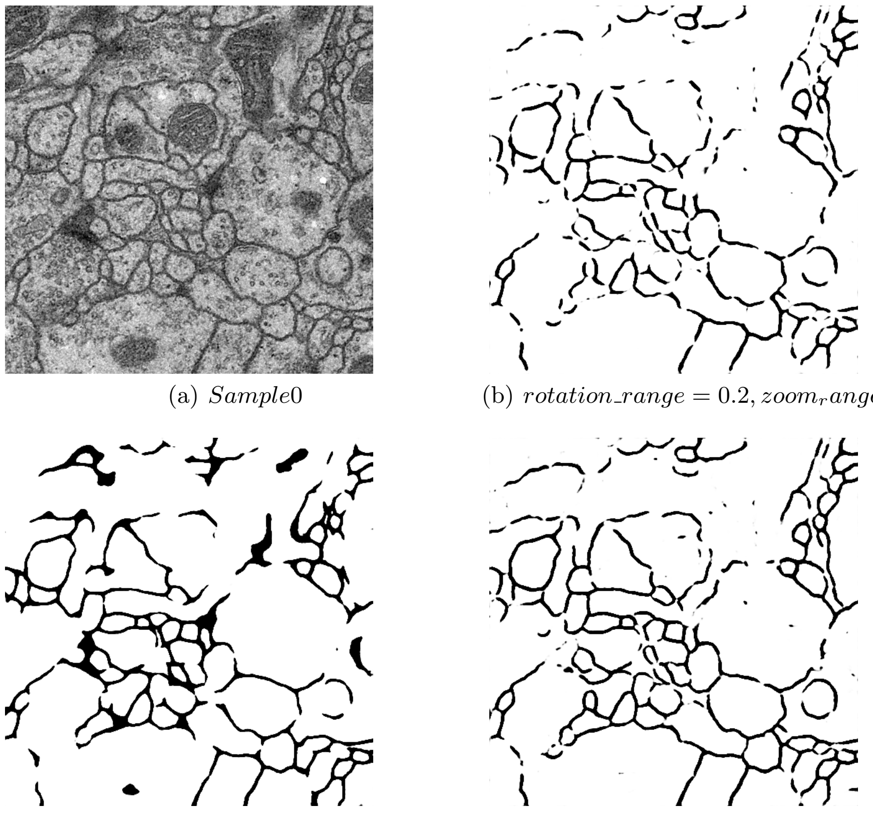
Fig. 9: Performance of Unet++ on three metrics

Accuracy	V_rand	V_info
0.908	0.454	0.831

Table 5: Average Performance on Three Metrics

After inspection, we could find that the overall performance of V_rand is not such satisfactory, as the overall performance on table 5 appears to be only 0.45, which is overwhelmed by Unet model. However, other two metrics are still great compared to the predecent one, with average accuracy 0.908 and average V_info 0.831. In pixel level, the accuracy often fluctuates up and down among 0.92 and is relatively stable. This is because pixel level measurement does not regard a segment as an ensemble and this metrics compares purely the similarity between two figures.

Data enhancement
In the process of model adjustment, we found that a class of parameters that have a greater impact on the model is the parameters for image processing. The more representative ones are *rotation_range* and *zoom_range*. The former determines the random rotation angle of the image when the data is promoted, and the latter determines the random zoom range of the image.



(a) $Sample0$ (b) $rotation\_range = 0.2, zoom_range = 0.05$

(c) $rotation\_range = 20, zoom_range = 0.2$ (d) $rotation\_range = 10, zoom_range = 0.05$

Fig. 10: Sample 0 and the segmentation results under different data enhancement parameters

From the above comparison, we can see the impact of data enhancement. When we fill with $fill\_mode =' constant'$ (fill with 0), both *rotation_range* and *zoom_range* will produce blank areas. When these two values are larger, the area filled is more. At this time, the model is very cautious, and only predicts 0 (that is, black) for the area that it has a large grasp of, so you can see the black in the three settings The difference in the boundary part. The reason is that the loss back propagation of the filled area guides the model to a more accurate direction of $predict = 0$. However, the advantage of *rotation_range* and *zoom_range* is that the generated images are highly diversified, so the accuracy and meanIOU that can be achieved during training are increased.

In addition, we also used two experimental indicators to score the results of each group, and the final results also conformed to our analysis(This result is only the score of a single sample 0, and is only used as an inquiry parameter to influence the experimental results.).

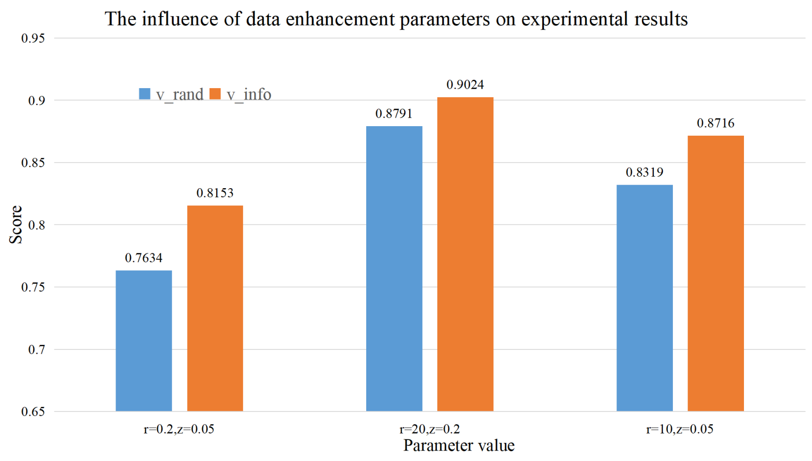
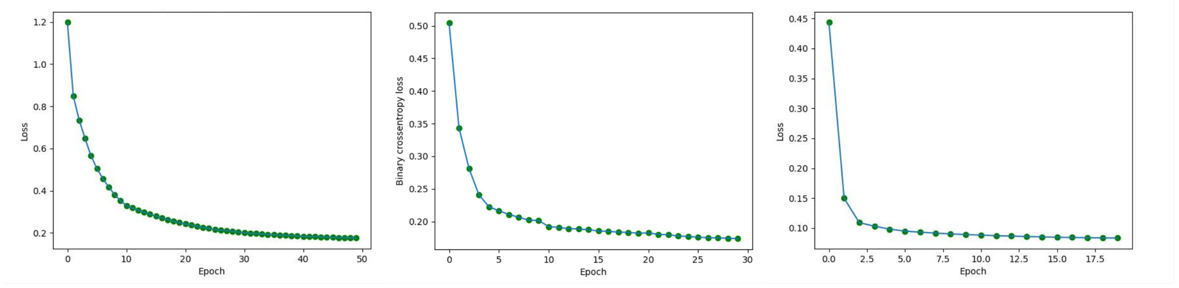


Fig. 11: The influence of data enhancement parameters on experimental results

Loss function

In the previous literature review, we found that the performance of medical image segmentation is greatly affected by the loss function. Therefore, we experimented to control the remaining parameters to be the same, and tested the results under the three loss functions of Binary Cross Entropy and Weighted Binary Cross Entropy.

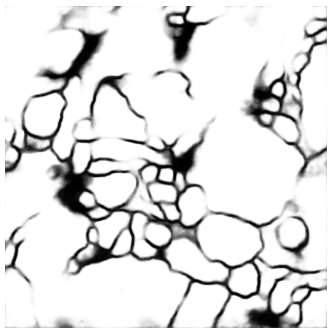
Here, we take sample 0 as an example to show the image segmentation results obtained in the three loss functions. From the training curve in the Fig.**??** below, we observe that *Dice* converges much faster than the other two (BCE roughly requires 30 rounds of Epoch to converge, while Dice only needs 5 Epochs), and the overall performance is roughly It is $Dice > Weighted\_BCE > BCE$. In order to reflect the experimental effect more intuitively, we also give the prediction results of the three in the following article.

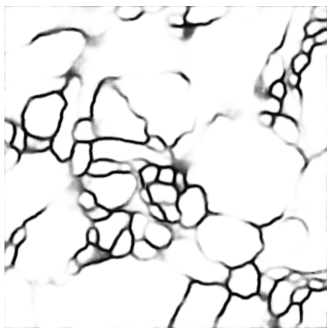


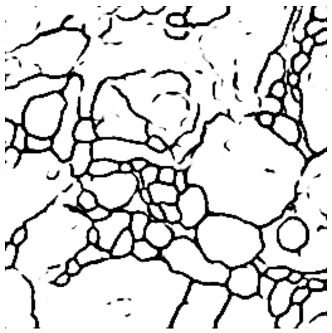
(a) BCE (b) $Weighted\_BCE$ (c) $Dice$

Fig. 12: Training loss curve of different loss functions

First, it is easy to find that the result of *weighted_bce* (Fig.13(b)) is significantly better than *bce* (Fig.13(a)). This is because in our segmentation task, the "black" pixels judged to be boundaries are significantly less than the "white" pixels judged to be internal, and the introduction of weights can better balance the gap between positive and negative samples.


(a) BCE


(b) $Weighted\_BCE$


(c) $Dice$

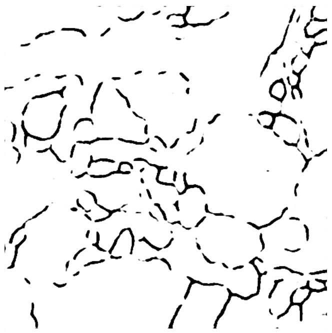

(d) IOU

Fig. 13: Sample 0 and the segmentation results under the three loss functions

The Dice coefficient is a set similarity measurement function, which is usually used to calculate the similarity of two samples. The mathematical formula is as follows:

$$Dice\_coefficient = \frac{2|X \cap Y|}{|X| + |Y|}$$

It is not difficult to find that the performance of dice (Fig.13(c)) is far better than that of bce, and the performance on the image is that there are almost no fuzzy boundaries in the segmentation results of dice. I think this is because bce only considers the difference between pixels, and does not directly consider whether the boundaries of the target image and the predict image are similar. In fact, the latter is intuitively more in line with the requirements of our task: divide the boundaries as much as possible Accurate. In other words, the real goal of segmentation is to maximize the dice-coefficient and IoU metrics. However, cross entropy is only a form of agency, which is easy to maximize optimization in BP.0

However, following this idea, when selecting $1 - MeanIOU$ as the training loss, it was found that the effect (Fig.13(d)) was lower than expected. In fact, joint intersection (IoU) is the most popular evaluation metric in object boundary detection benchmarks. However, there is a gap between optimizing the commonly used distance loss for regression bounding box parameters and maximizing this metric. The best goal of the indicator is the indicator itself. In the case of an axis-aligned 2D bounding box, it can be proved that IoU can be directly used as the regression loss. However, IoU has a plateau and cannot be optimized in the case of non-overlapping bounding boxes.

4 Conclusion

In this project, we tried to use two network structures, Unet and U-net++, to detect cell boundaries. In theory, U-net++ benefits from improving the jump path between the encoder and the decoder on the basis of U-net, improving the segmentation accuracy, and should be able to achieve better performance. But in fact, U-net++ scores in both indicators are far inferior to Unet. Limited by time, we could not find a reasonable explanation, but this is undoubtedly the direction of the subsequent expansion of the project. In addition, in the process of exploring the performance of U-net++ below expectations, we have also been able to gain a deeper understanding of the influence of factors such as loss function and data enhancement on the results of boundary prediction. In the end, our best performing model was Unet, which achieved a score of $V\_rand = 0.80, V\_info = 0.94$.

5 Contribution

Wu Shiqu
1.Program code for Unet and Unet++ model.
2.Completed most of the experimental work.

Yan Binghao
1.Paper writing (abstract,introduction, related work part and Unet++ experimental part).
2.Visualization of the corresponding paper part.

Hou Shengyuan
1.Paper writing (Conclusion and Unet experimental part).
2.Visualization of the corresponding paper part.
3.Program code for data augmentation and metrics computation.

References

1. Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation[C]//International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015: 234-241.
2. Zhou Z, Siddiquee M M R, Tajbakhsh N, et al. Unet++: A nested u-net architecture for medical image segmentation[M]//Deep learning in medical image analysis and multimodal learning for clinical decision support. Springer, Cham, 2018: 3-11.
3. Arganda-Carreras I, Turaga S C, Berger D R, et al. Crowdsourcing the creation of image segmentation algorithms for connectomics[J]. Frontiers in neuroanatomy, 2015, 9: 142.
4. Turaga S C, Briggman K L, Helmstaedter M, et al. Maximin affinity learning of image segmentation[J]. arXiv preprint arXiv:0911.5372, 2009.
5. Nunez-Iglesias J, Kennedy R, Parag T, et al. Machine learning of hierarchical clustering to segment 2D and 3D images[J]. PloS one, 2013, 8(8): e71715.