Nucleus Detection and Segmentation

Jiahua Li, li.jiahu@husky.neu.edu
Bingling Fu, fu.bi@husky.neu.edu

Abstract

The nucleus is an organelle that is present in all eukaryotic cells, including human cells. Abnormal nucleus shapes can be used to identify cancer cells (e.g., cervical smears and diagnosis of cervical cancer). Similarly, a growing body of literature indicates that there is a link between the shape of the nucleus and human disease states such as cancer and aging. Therefore, quantitative assessment of nuclear size and shape has important biomedical applications. Methods for assessing kernel size and shape typically involve identifying cores by conventional image segmentation methods. Here, we show a deep learning method for identifying and segmenting nucleus from cell images. We use a U-Net CNN model to recognize these images, the recognition rate reached above 97%.

Keywords: Nucleus, Detection, Segmentation, CNN, U-net CNN, Deep Learning, Contracting Path, Expanding Path

Ⅰ. Introduction

Background

We've all seen people suffer from diseases like cancer, heart disease, chronic obstructive pulmonary disease, Alzheimer's, and diabetes. Identifying the cells' nuclei is the starting point for most diseases analyses because most of the human body's 30 trillion cells contain a nucleus full of DNA, the genetic code that programs each cell.

Motivation

By far, success rates of the CNN methods, which based on mainstream deep learning structure, can be over 97% on the benefit of the development of the convolutional neural network technology, and this is different from the traditional machine learning methods. However, based on the actual situation, we try to reconstruct the CNN structure to U-Net CNN.

Approach

CNN uses a variation of multilayer perceptron designed to require minimal preprocessing. Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. However, in our research, we use a U-net CNN. The architecture of U-net CNN consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. We show that such a network can be trained end-to-end from very few images and outperforms the prior method

Ⅱ. Methods

We use CNN as our main approach to do our project. In machine learning, CNN is a class of deep, feed-forward artificial neural network that has successfully been applied to analyzing visual imagery. CNNs use a variation of multilayer perceptron designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks, based on their shared-weights architecture and translation invariance characteristics.

Convolutional Neural Network

Convolutional networks are inspired by biological processes because the pattern of connections between neurons is similar to that of the animal's visual cortex. Individual cortical neurons respond to the independence of prior knowledge, and human effort in feature design is a major advantage.

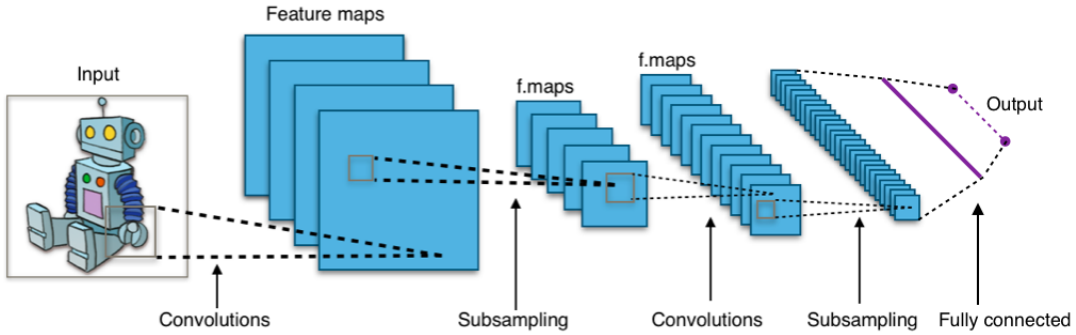


Image 2-1

Image 2-1 shows the typical architecture of CNN. We can see that the architecture consists of a bunch of different layers that convert the input into output through a differentiable function. Several different types of layers are typically used.

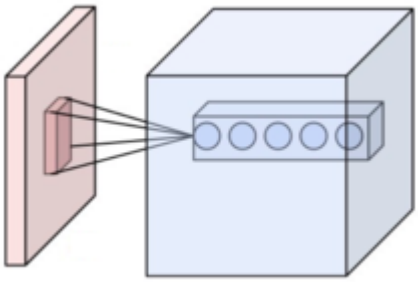


Image 2-2

The image 2-2 is a structure of a convolutional layer, the parameters of a layer consist of a set of learnable filters (or kernels) that have a small perceived area but extend to the entire depth of the input volume. During forward transfer, each filter convolves over the width and height of the input volume, calculates the dot product between the filter's entries and the input, and produces a two-dimensional activation map of the filter. As a result, the network learns filters that are activated when a particular type of feature is detected at a

spatial location in the input.

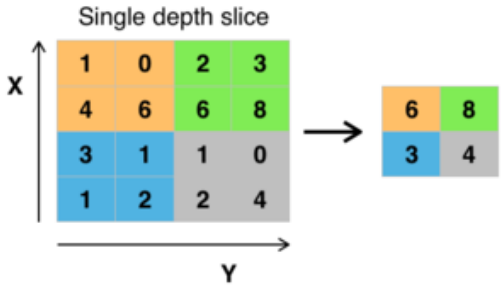


Image 2-3

Image 2-3 is the sample of the pooling layer. The polling layer operates independently on each depth slice entered and spatially resizes it. The most common form is a pooling layer with a 2×2 size filter that applies 2 down sampled steps along the width and height at each depth slice in the input, discarding 75% of the activation. In this case, each maximum operation exceeds 4 digits. The depth dimension remains the same.

The loss layer specifies how the training penalizes the deviation between the predicted label and the real label, and is usually the last layer. There are various loss functions that are suitable for different tasks, which is our main point of view.

Enter the convolution operator. Given a two-dimensional image, I, and a small matrix, K of size $h \times w$, (known as a convolution kernel), which we assume encodes a way of extracting an interesting image feature, we compute the convolved image, $I*K$, by overlaying the kernel on top of the image in all possible ways, and recording the sum of elementwise products between the image and the kernel:

$$(I * K)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} K_{ij} * I_{x+i-1,y+j-1}$$

U-Net Convolutional Neural Network

U-Net is a convolutional neural network developed for biomedical image segmentation at the Department of Computer Science at the University of Freiburg, Germany. The network is based on a fully convolutional network whose architecture has been modified and extended to use fewer training images and produce more accurate segmentation.

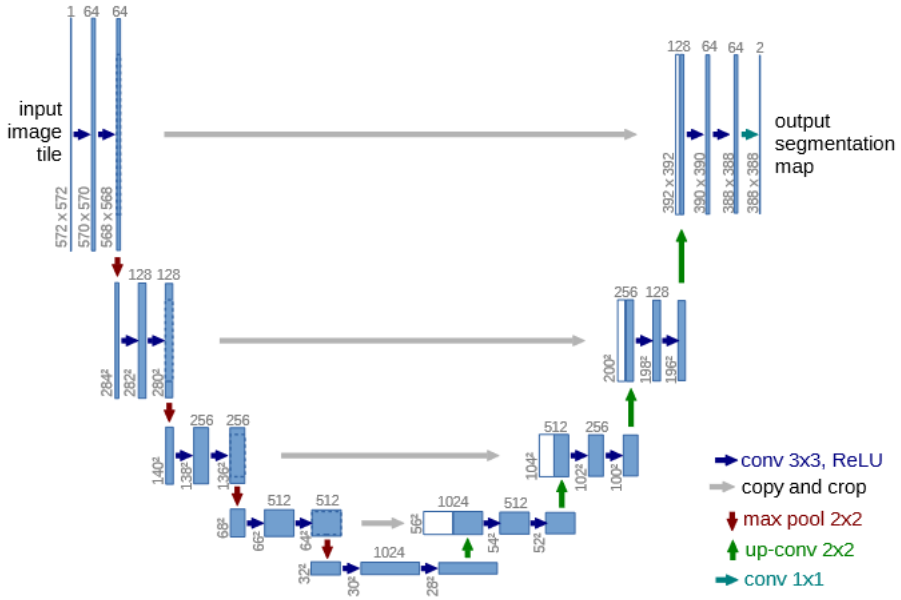


Image 2-4

Image 2-4 is the architecture of U-Net CNN (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.

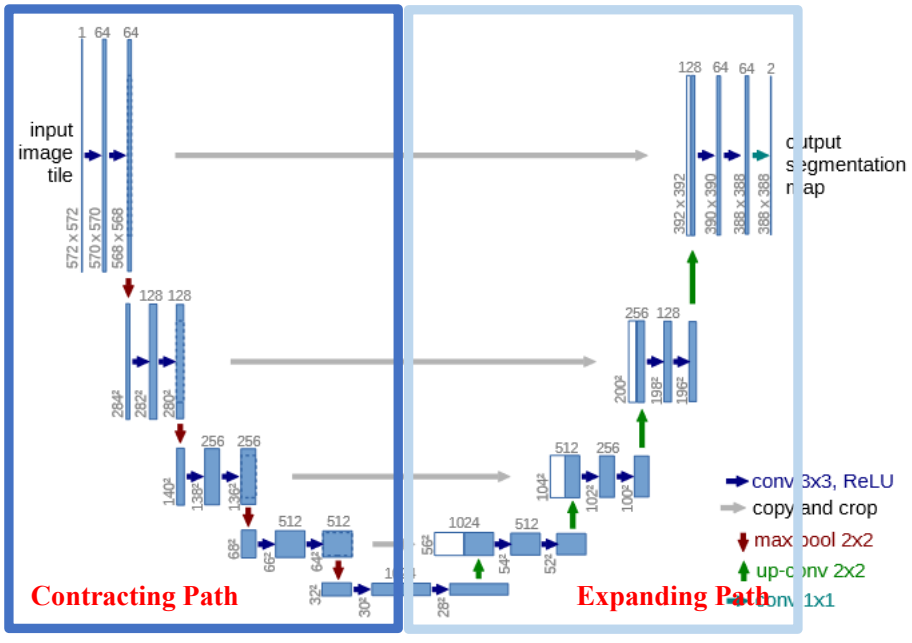


Image 2-5

The architecture of U-net CNN consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. We show that such a network can be trained end-to-end from very few images and outperforms the prior method.

Ⅲ. Practical Methods & Approaches

Simple Convolutional Neural Networks

Layer (type)	Output Shape	Param #
NormalizeInput (BatchNormali	(None, None, None, 3)	12
conv2d_1 (Conv2D)	(None, None, None, 8)	224
conv2d_2 (Conv2D)	(None, None, None, 8)	584
conv2d_3 (Conv2D)	(None, None, None, 16)	1168
conv2d_4 (Conv2D)	(None, None, None, 16)	2320
conv2d_5 (Conv2D)	(None, None, None, 32)	4640
conv2d_6 (Conv2D)	(None, None, None, 16)	528
conv2d_7 (Conv2D)	(None, None, None, 1)	17

Image 3-1

Image 3-1 shows the basic structure of Simple CNN, which just includes one input layer, seven convolutional layers and one fully connected layer.

Prediction Result of Simple CNN

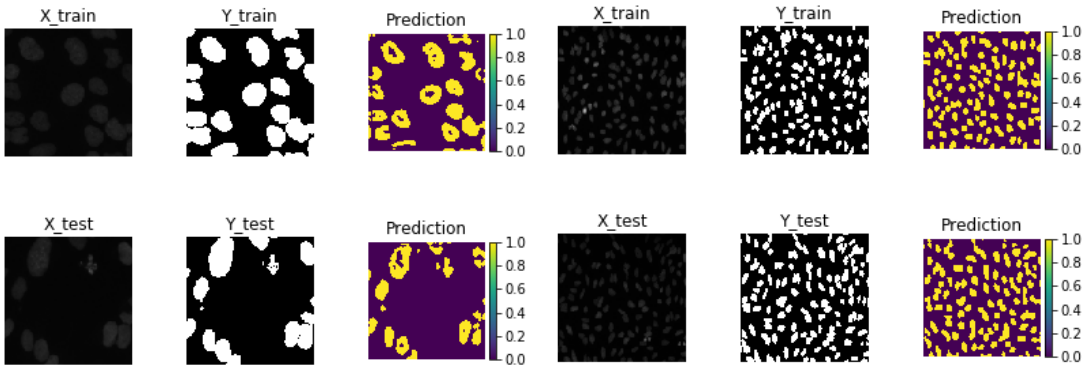


Image 3-2

Image 3-1 shows the partial prediction result of simple CNN model. The x_train and x_test are inputted images. Y_train and Y_test are clipped masks, which is the true masks of inputted images. And the Prediction is the prediction result of simple CNN model.

Under our observation, we found that the prediction result is not accurate enough. Obviously, the prediction result is blurred, there are some shadow inside nucleus. And the boundary of nucleus is not clear.

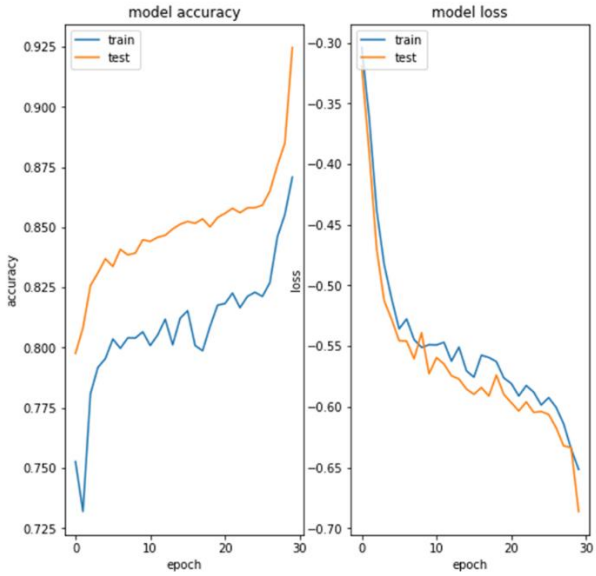


Image 3-3

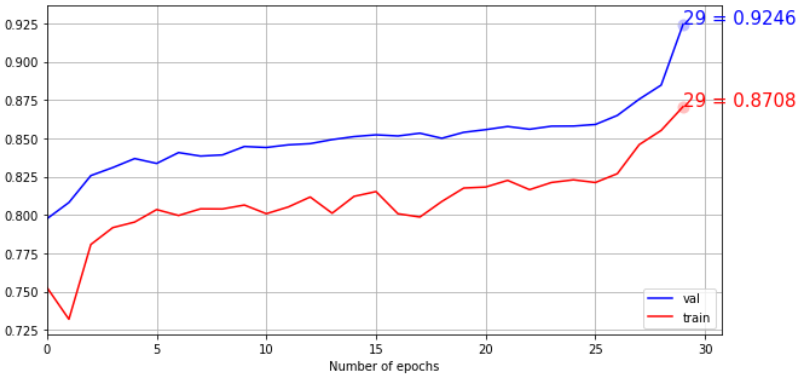


Image 3-4

Image 3-2 and 3-3 are model accuracy and model loss line chart. Although, when we increased the number of epochs, the model accuracy would be greater, and the model loss would be less. When we set the number of epochs as 30, the accuracy would be 0.87. However, this model still has some problems, such as bad prediction result, blurred and unclear boundary, which is why in next step, we would like to use U-Net CNN to further improve our model.

U-Net Convolutional Neural Networks

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 128, 128, 3)	0	
lambda_1 (Lambda)	(None, 128, 128, 3)	0	input_1[0][0]
conv2d_8 (Conv2D)	(None, 128, 128, 16)	448	lambda_1[0][0]
dropout_1 (Dropout)	(None, 128, 128, 16)	0	conv2d_8[0][0]
conv2d_9 (Conv2D)	(None, 128, 128, 16)	2320	dropout_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 16)	0	conv2d_9[0][0]
conv2d_10 (Conv2D)	(None, 64, 64, 32)	4640	max_pooling2d_1[0][0]
dropout_2 (Dropout)	(None, 64, 64, 32)	0	conv2d_10[0][0]
conv2d_11 (Conv2D)	(None, 64, 64, 32)	9248	dropout_2[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 32)	0	conv2d_11[0][0]
conv2d_12 (Conv2D)	(None, 32, 32, 64)	18496	max_pooling2d_2[0][0]
dropout_3 (Dropout)	(None, 32, 32, 64)	0	conv2d_12[0][0]
conv2d_13 (Conv2D)	(None, 32, 32, 64)	36928	dropout_3[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 64)	0	conv2d_13[0][0]
conv2d_14 (Conv2D)	(None, 16, 16, 128)	73856	max_pooling2d_3[0][0]
dropout_4 (Dropout)	(None, 16, 16, 128)	0	conv2d_14[0][0]
conv2d_15 (Conv2D)	(None, 16, 16, 128)	147584	dropout_4[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 8, 8, 128)	0	conv2d_15[0][0]
conv2d_16 (Conv2D)	(None, 8, 8, 256)	295168	max_pooling2d_4[0][0]
dropout_5 (Dropout)	(None, 8, 8, 256)	0	conv2d_16[0][0]
conv2d_17 (Conv2D)	(None, 8, 8, 256)	590080	dropout_5[0][0]
conv2d_transpose_1 (Conv2DTrans	(None, 16, 16, 128)	131200	conv2d_17[0][0]
concatenate_1 (Concatenate)	(None, 16, 16, 256)	0	conv2d_transpose_1[0][0] conv2d_15[0][0]
conv2d_18 (Conv2D)	(None, 16, 16, 128)	295040	concatenate_1[0][0]
dropout_6 (Dropout)	(None, 16, 16, 128)	0	conv2d_18[0][0]
conv2d_19 (Conv2D)	(None, 16, 16, 128)	147584	dropout_6[0][0]
conv2d_transpose_2 (Conv2DTrans	(None, 32, 32, 64)	32832	conv2d_19[0][0]
concatenate_2 (Concatenate)	(None, 32, 32, 128)	0	conv2d_transpose_2[0][0] conv2d_13[0][0]
conv2d_20 (Conv2D)	(None, 32, 32, 64)	73792	concatenate_2[0][0]
dropout_7 (Dropout)	(None, 32, 32, 64)	0	conv2d_20[0][0]
conv2d_21 (Conv2D)	(None, 32, 32, 64)	36928	dropout_7[0][0]
conv2d_transpose_3 (Conv2DTrans	(None, 64, 64, 32)	8224	conv2d_21[0][0]
concatenate_3 (Concatenate)	(None, 64, 64, 64)	0	conv2d_transpose_3[0][0] conv2d_11[0][0]
conv2d_22 (Conv2D)	(None, 64, 64, 32)	18464	concatenate_3[0][0]
dropout_8 (Dropout)	(None, 64, 64, 32)	0	conv2d_22[0][0]
conv2d_23 (Conv2D)	(None, 64, 64, 32)	9248	dropout_8[0][0]
conv2d_transpose_4 (Conv2DTrans	(None, 128, 128, 16)	2064	conv2d_23[0][0]
concatenate_4 (Concatenate)	(None, 128, 128, 32)	0	conv2d_transpose_4[0][0] conv2d_9[0][0]
conv2d_24 (Conv2D)	(None, 128, 128, 16)	4624	concatenate_4[0][0]
dropout_9 (Dropout)	(None, 128, 128, 16)	0	conv2d_24[0][0]
conv2d_25 (Conv2D)	(None, 128, 128, 16)	2320	dropout_9[0][0]
conv2d_26 (Conv2D)	(None, 128, 128, 1)	17	conv2d_25[0][0]

Image 3-5

Image 3-5 shows the structure of U-Net CNN, which consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. In the contracting path, we use convolutional layer to capture features, pooling layer to further process the feature map and the dropout layer to reduce the overfitting. On the contrary, in the expanding path, we use counter-convolutional layer to combine all feature maps into one prediction mask based on the nucleus position.

Prediction Result of U-Net CNN

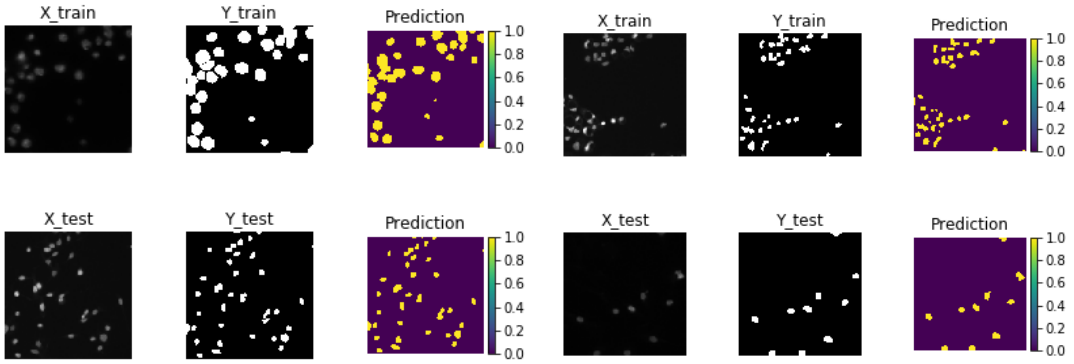
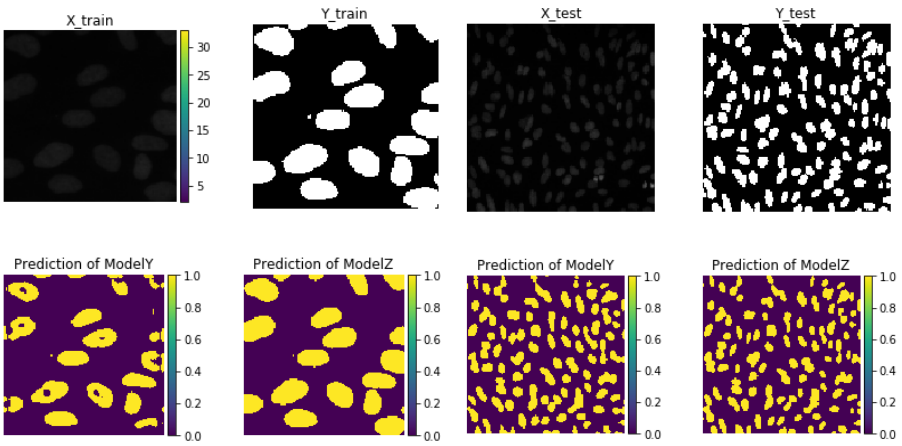


Image 3-6

Image 3-1 shows the partial prediction result of simple CNN model. The x_train and x_test are inputted images. Y_train and Y_test are clipped masks, which is the true masks of inputted images. And the Prediction is the prediction result of simple CNN model.

Under our observation, we found that the prediction result is more accurate. Obviously, the prediction result is clearer, there are almost no shadow inside nucleus. And the boundary of nucleus is clear.



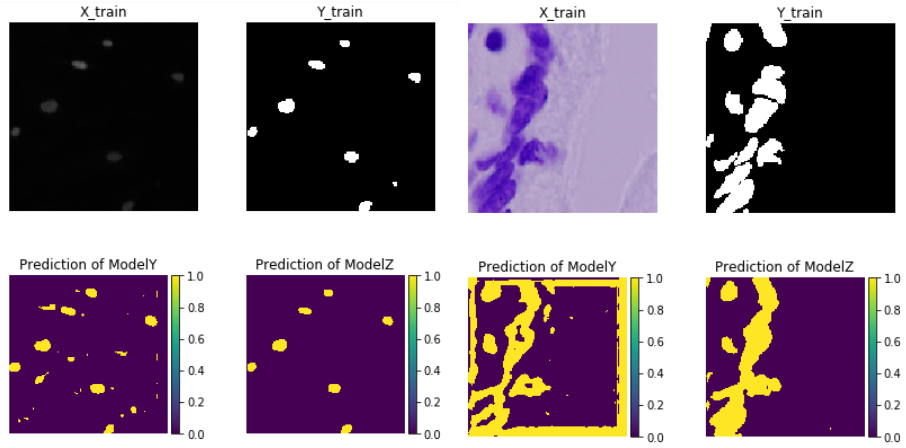


Image 3-7

Image 3-7 shows the prediction result of both model, simple CNN and U-Net CNN. The prediction of ModelY is from simple CNN and the prediction of ModelZ is from U-Net CNN. Apparently, the prediction result of U-Net CNN is clearer and less blurred shadow than the prediction result of Simple CNN.

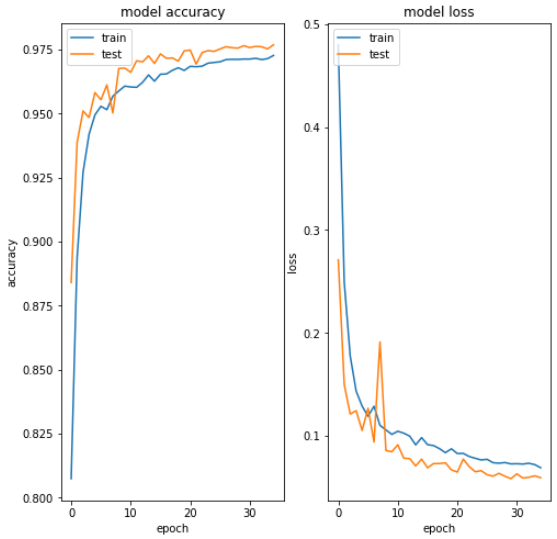


Image 3-8

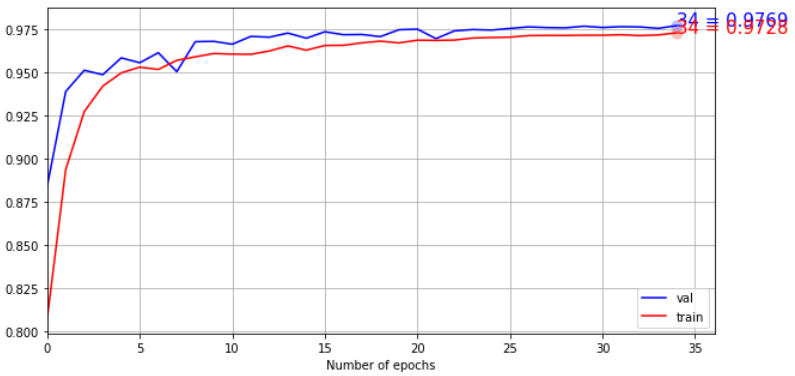


Image 3-9

We can see the line chart in the image 3-8 and 3-9. After Using U-Net, the train accuracy curve is about 0.97. Apparently, the U-Net CNN model accuracy is higher than the accuracy of simple CNN.

Ⅳ. Results

From our research, we found that the U-Net CNN has a better result than the simple CNN. The accuracy of simple CNN is about 0.87, and there is not good prediction result in simple CNN model. There are some blurred and useless shadow inside nucleus. And the outline of nucleus is not clear, and sometimes this model even is unable to detect and segment a correct nucleus. However, after we reconstruct the structure of CNN by adding U-net part, the accuracy of model is 0.97. And the outline of nucleus is clearer and less burred. Meanwhile, there is almost no useless and blurred shadow inside the nucleus. Based on all we did, we found the U-net CNN is an incredibly good model, which could be used in the medical, biological and biochemical area.

Ⅴ. Discussion

Through the whole process of building our project, we found these different aspects that we want to discuss about. Based on our results, the traditional identifying code has less accuracy. When we use the U-Net, the accuracy has improved a lot. It is show that the effect of the U-Net in image detection. By increasing the dataset, we can further improve the accuracy of model. We can add more functions, like counting the number of cells in one image.

Ⅵ. Code with Documentation

https://github.com/lijiahua92/academicprojects/blob/master/Nucleus_Detection_and_Segmentation/Project/ProjectPortfolio.ipynb

Ⅶ. References

[1]. https://en.wikipedia.org/wiki/U-Net
[2]. https://en.wikipedia.org/wiki/Convolutional_neural_network
[3]. https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html
[4]. https://www.kaggle.com/paultimothymooney/identification-and-segmentation-of-nuclei-in-cells
[5]. https://www.kaggle.com/byrachonok/find-the-nuclei-in-divergent-medical-images/notebook
[6]. Ronneberger O., Fischer P., Brox T. (2015) U-Net: Convolutional Networks for Biomedical Image

Segmentation. In: Navab N., Hornegger J., Wells W., Frangi A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer, Cham