

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 人工智能 |
| 姓 名： | 袁东琦 林子涵 |
| 学 院： | 信息与电子工程学院 |
| 系： |  |
| 专 业： | 信息工程 |
| 学 号： | 3200103602 3200104767 |
| 指导教师： | 龚小瑾 |

2023 年 6 月 15 日

**实验报告**

课程名称：人工智能 指导老师：龚小瑾

项目名称：基于TensorFlow的手写数字识别 成绩：

1. **Prerequisite knowledge and environment setup for the project**

1. Objective: Use Python and TensorFlow to design a handwritten digit recognition algorithm, and program a GUI interface to build a handwriting recognition system.

2. Basic approach:

1)Prepare training and testing data MNIST;

2)Build models - we constructed three different deep neural networks for model comparison;

3)Train and save the model for subsequent calls;

4)Evaluate the model - use the test set to observe model performance.

3. Handwritten digit recognition dataset MNIST: MNIST includes a training set of 60,000 images and labels, and a test set of 10,000 images and labels. Each image is a grayscale picture of size 28x28 pixels (784 pixels in total), with each pixel represented by a float to indicate its brightness level. The dataset contains digits from 0 to 9.

**二、The process of the project**

1. Data preprocessing

We created a homemade handwritten digit dataset by drawing the digits 0-9 using graphic software, which we will use to test our neural network and explore how the same model performs with different datasets. To prepare these homemade images for testing, we need to preprocess them to make them consistent with the MNIST dataset. First, we resize the imported images to 28x28 pixels, then convert them to grayscale and flip their black and white values. The resulting images of our homemade dataset are shown below.

2345

1. Data importation

For our self-built UNet and ResNet neural network models, we use the mnist.load\_data() function to import the MNIST dataset, normalize the data and expand it to four dimensions. During each training session, we randomly select 5000 data points as the training set for the network.

As for the VGG16 network model, we need to process the data using functions from the cv2 library to convert grayscale images to RGB images, in order to meet the dimensionality requirements of the VGG16 function for training data.

1. **Other attempts**

**After my presentation, the feedback from the teacher prompted some reflection on why the Resnet18 network's test results were not as good as those of the VGG16 network. The main reasons could be:**

**1)The GUI interface randomly selected 30 samples from a pool of 10,000 testing data in the MNIST dataset, which introduced a large amount of randomness into the test results. Each training and testing process can produce significant fluctuations, which heavily impact the final test results.**

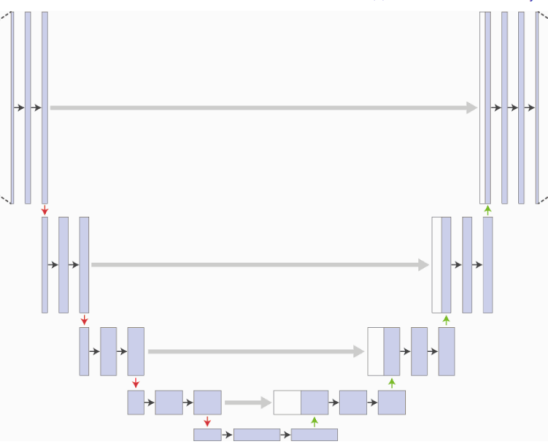
**2)The Unet and Resnet18 networks were trained using grayscale images while the VGG16 was trained using RGB images. RGB images typically contain more information than grayscale ones, which allows for the extraction of more features, resulting in better training results. This may explain why the Resnet18 network's test results were not as good as VGG16's.**

**To address these possible issues, I made adjustments to the model training function and conducted two experiments. Please refer to *section four improvements*.**

2、模型搭建

1）unet

The basic model of Unet is shown in the following figure:



The network structure of Unet resembles the letter U and is called Unet. The left half is the feature extraction part, the right half is the upsampling part, and there are skip connections in between.

The model we built takes an input of 28\*28\*1. The feature extraction part consists of three convolutional layers with d=16 and kernel size=3, followed by maxpooling pooling. Then there are two convolutional layers with d=32 and kernel size=3, followed by maxpooling pooling. Finally, there are two convolutional layers with d=64 and kernel size=3.

The upsampling part is implemented using deconvolution and convolution functions, and the overall structure is symmetric to the feature extraction part. The output of the neural network is obtained by adding the input with the output of a convolutional layer with d=1 and kernel size=1 through a skip connection, followed by two fully connected layers. The output result is the one-hot matrix of the input label.

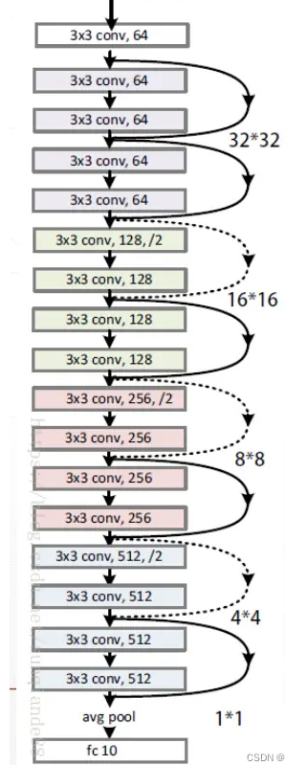
The loss function is selected as Huber, and the optimizer is selected as Adam.

1. Resnet

The first two layers of ResNet consist of 3\*3 convolutional layers with 64 output channels and a stride of 2, followed by a 3\*3 max pooling layer with a stride of 2. Unlike GoogLeNet, ResNet adds a batch normalization layer after each convolutional layer.

ResNet then uses four modules, each consisting of several residual blocks with the same number of output channels. The number of output channels in the first module is the same as that of the input channels. Since a max pooling layer with a stride of 2 has already been used, there is no need to reduce the height and width. Each subsequent module doubles the number of output channels in the first residual block and reduces the height and width by half.

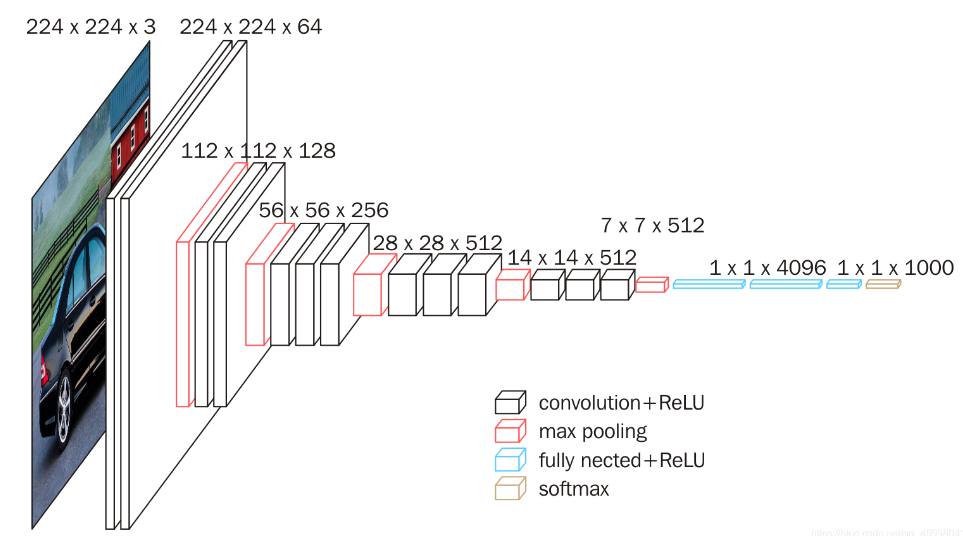
The basic structure of ResNet is shown in the following figure:



The network model we built is consistent with the diagram shown above, with an input of 28281 and an output of a one-hot matrix representing the label of the input image. The loss function used is categorical cross-entropy, and the optimizer used is Adam.

3)Vgg

VGG16 has a total of 16 layers, including 13 convolutional layers and 3 fully connected layers. After the first convolutional layer with 64 filters is applied twice, a pooling operation is performed. Then, after two rounds of convolution with 128 filters, another pooling operation is applied. This is followed by three rounds of convolution with 256 filters, and another pooling operation. After repeating this process twice with three convolutional layers each having 512 filters, another pooling operation is applied. Finally, the output is passed through three fully connected layers.



We built a network using the VGG16 function included in tensorflow. The input is 48\*48\*3, and the output is a one-hot matrix representing the label of the input image. The network consists of two fully connected layers with 1024 nodes each, followed by a fully connected layer with 10 nodes. The loss function used is huber, and the optimizer is Adam.

1. GUI

The GUI interface was constructed using QT Designer. The main functionality includes triggering training and testing of the model by clicking buttons, displaying the loss and accuracy obtained during training, displaying the test dataset and results, among others. The main interface of the GUI is implemented by exporting the UI file to a Python file, while the main functionality is implemented through coding.

1. the training and the test of the model









1. Loss and accuracy

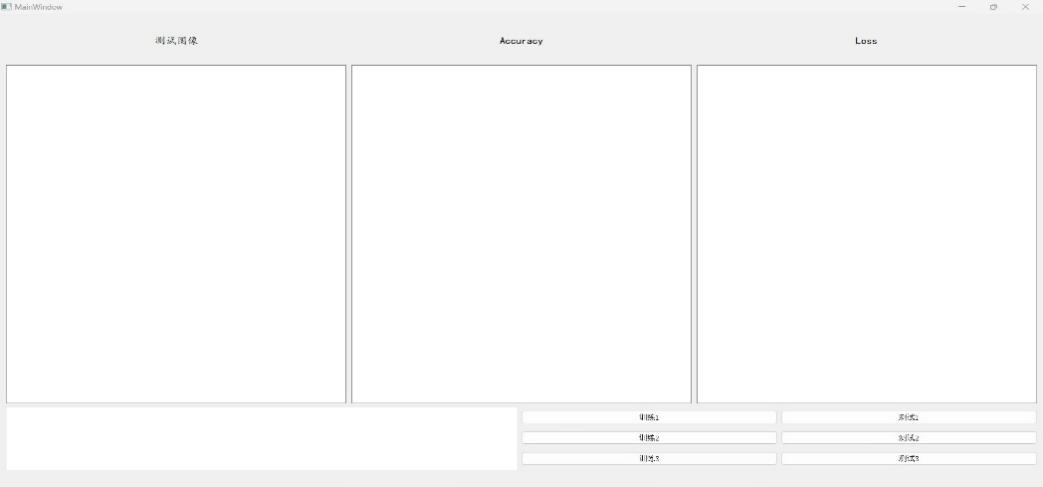


1. Displaying the test dataset and results.

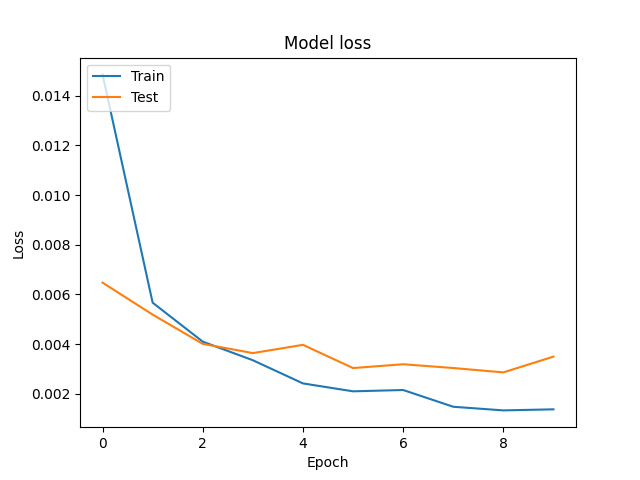
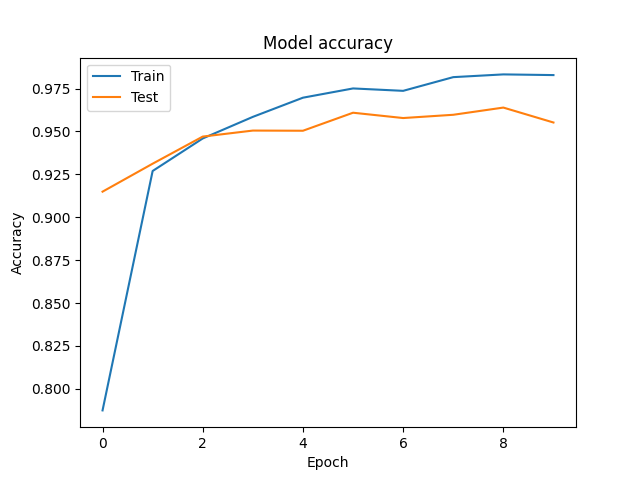


**三、Display and analysis of program running results**

1. Run the program call\_gui.py, and display the GUI main interface as shown in the following figure.

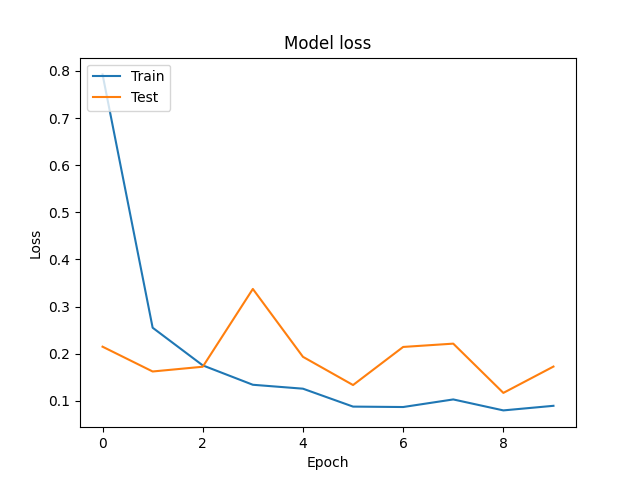
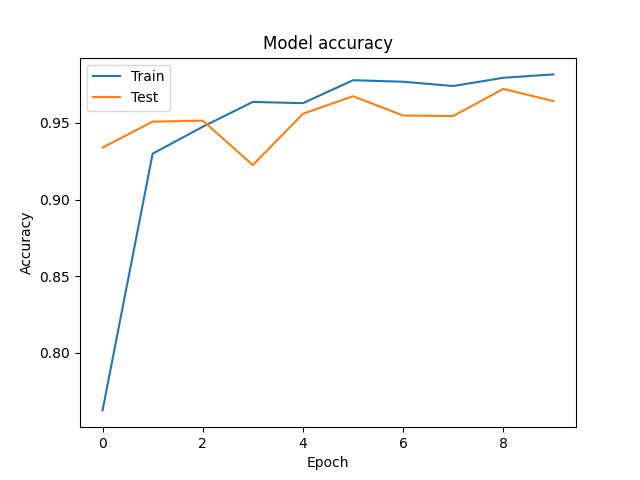


2.Click the button "Train 1" to train the U-Net model, and obtain the images of accuracy and loss as shown in the following figure.



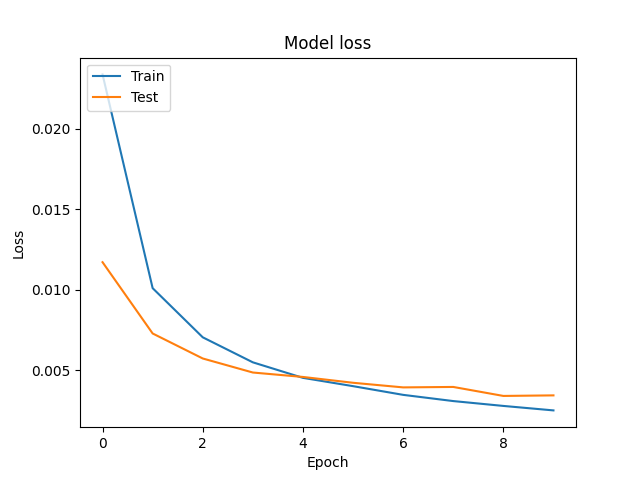
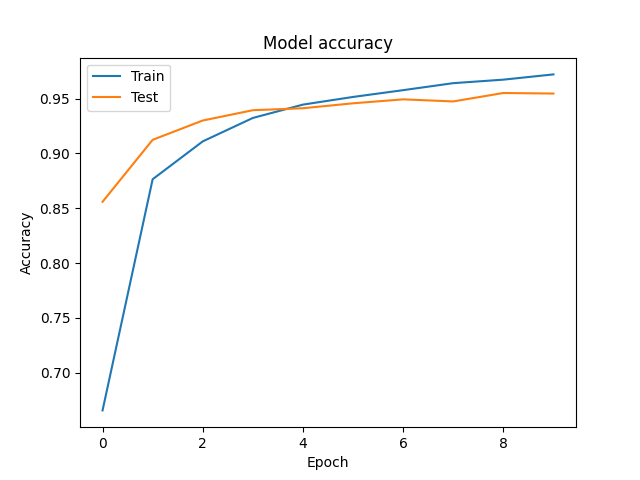
It can be seen that after 10 epochs of training, the accuracy of the network is roughly maintained at around 95%, and the loss is lower than 0.004.

3.Click the button "Train 2" to train the ResNet model, and obtain the images of accuracy and loss as shown in the following figure.



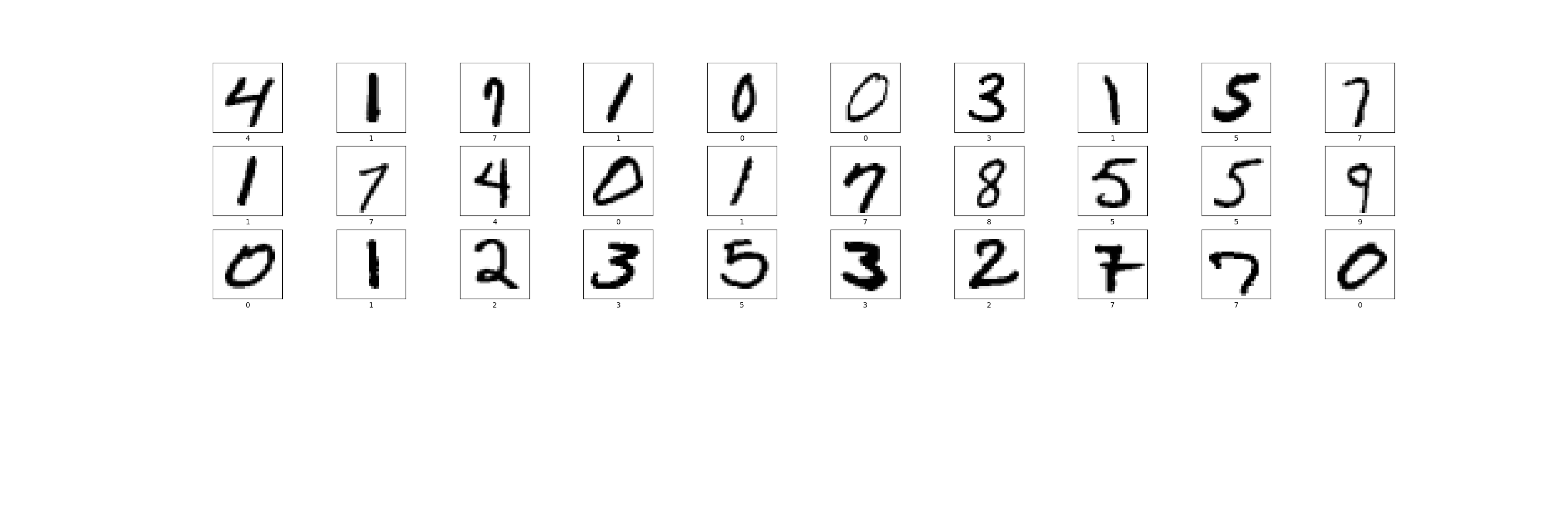
It can be seen that after 10 epochs of training, the accuracy of the network is roughly maintained at around 96%, and the loss is around 0.2 for the ResNet model.

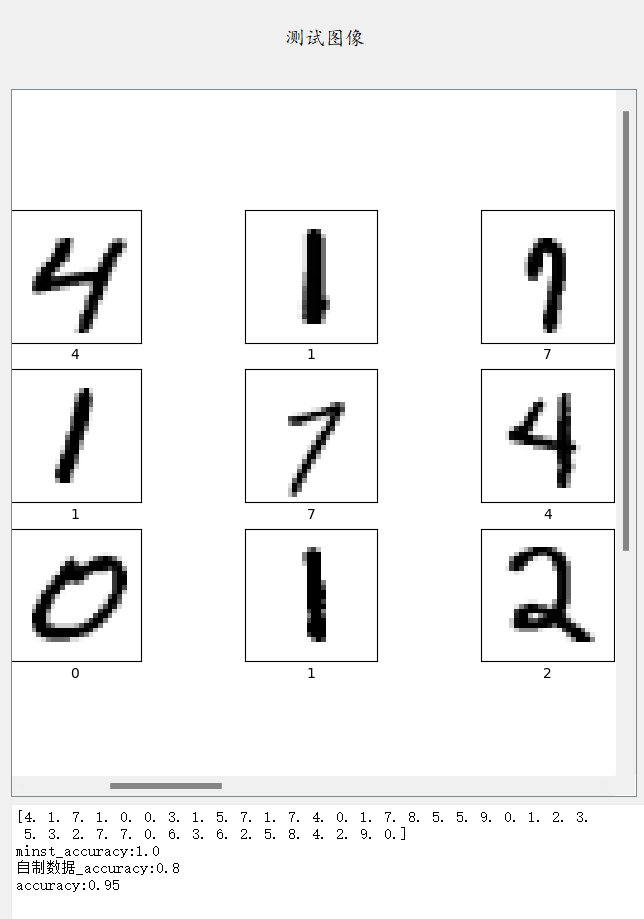
4.Click the button "Train 3" to train the VGG model, and obtain the images of accuracy and loss as shown in the following figure.



It can be seen that after 10 epochs of training, the accuracy of the network is roughly maintained at around 95%, and the loss is lower than 0.005 for the VGG model.

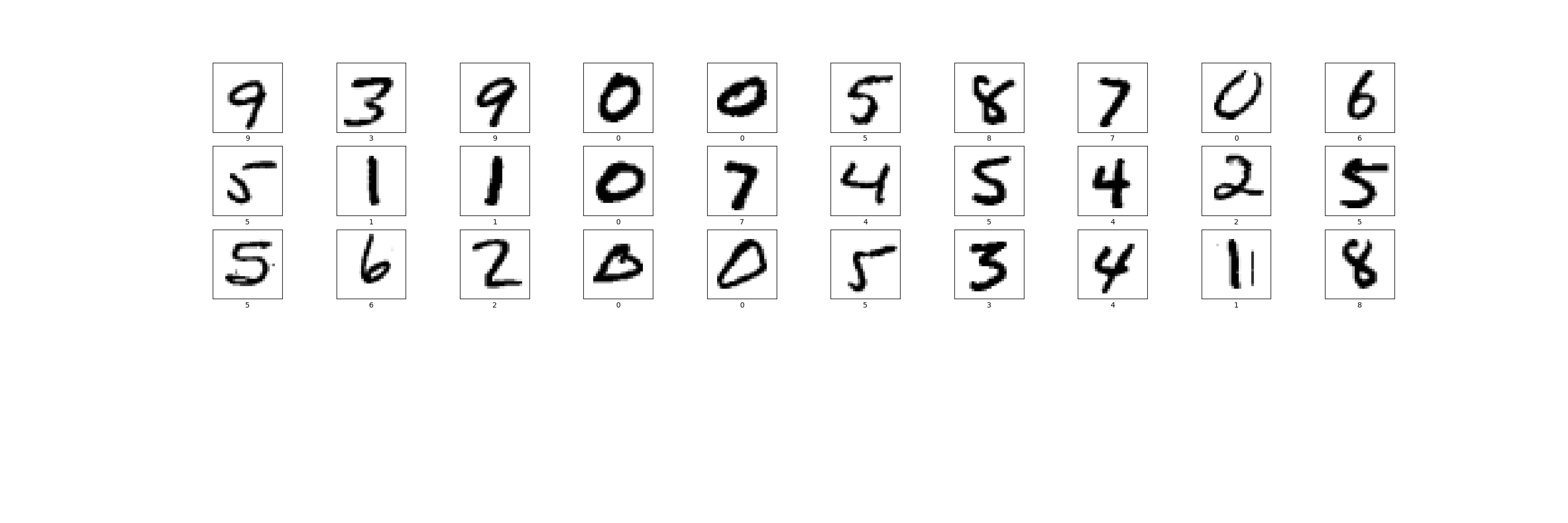
1. Click the button "Test 1" to test the U-Net model, and the detection image and results are shown in the following figure.

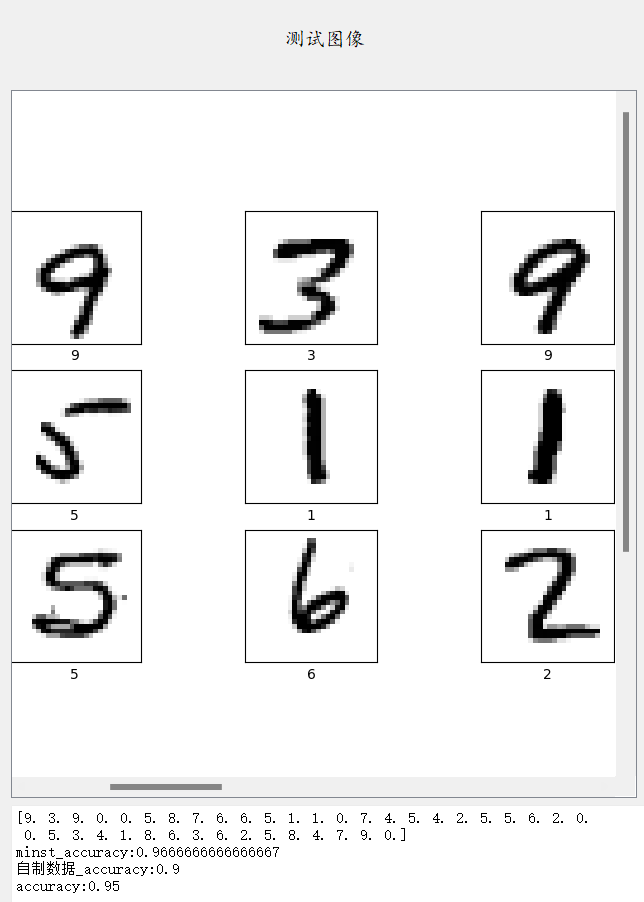




It can be seen that the U-Net network structure achieved good training results on the MINST dataset, with a recognition accuracy of 100% for 30 digits. However, the testing effect on the self-made dataset is not as good, with an accuracy of only 80%, resulting in an overall accuracy of 95%.

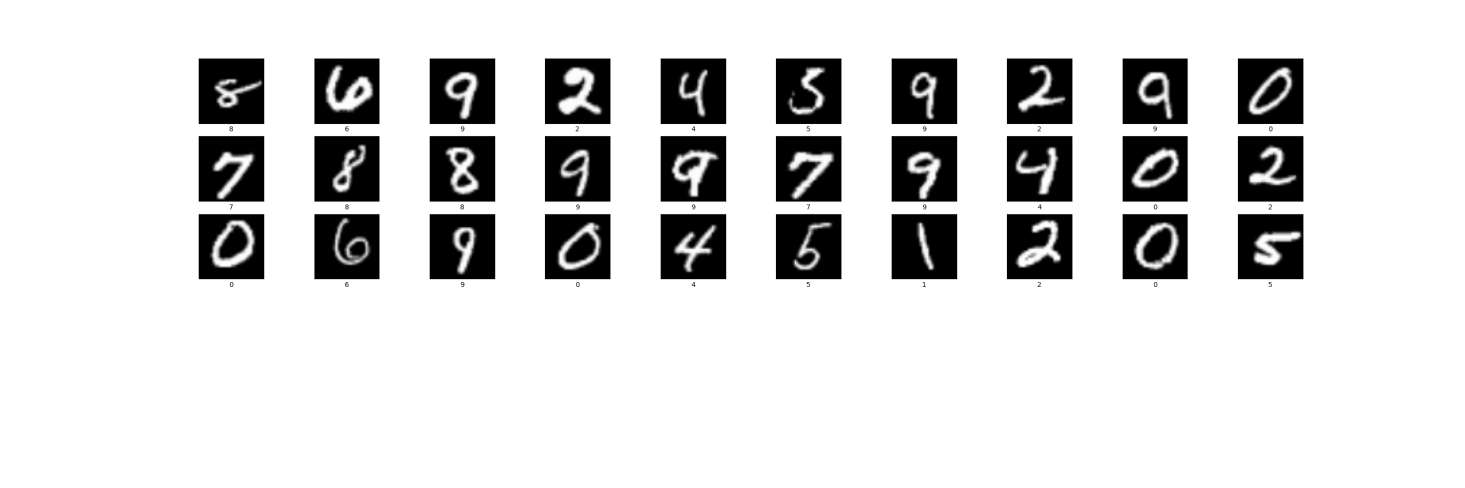
1. Click the button "Test 2" to test the ResNet model, and the detection image and results are shown in the following figure.

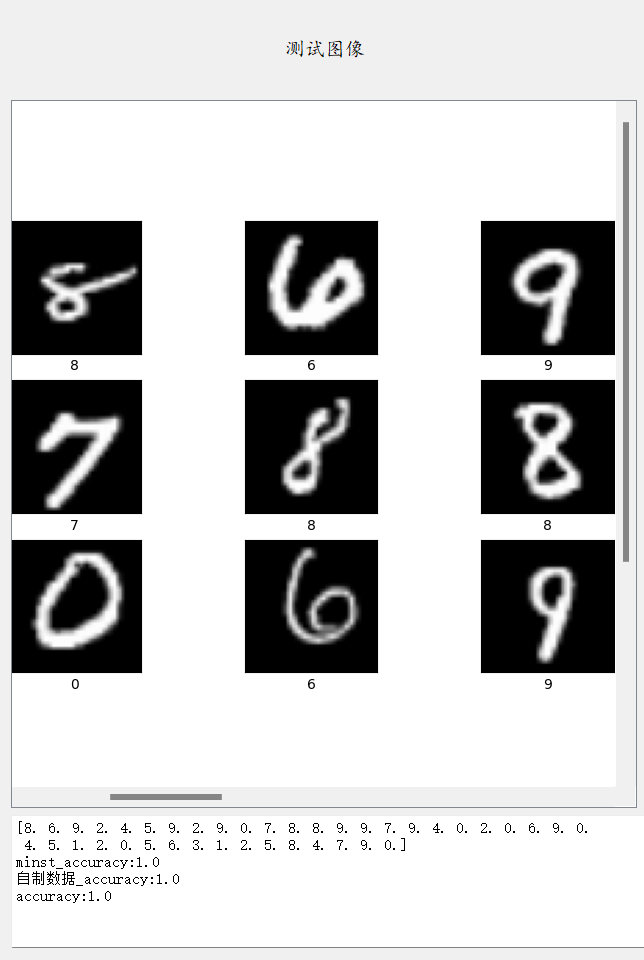




It can be seen that the ResNet network structure has almost the same testing effect on both the MINST dataset and the self-made dataset, with accuracies of 96.7% and 90%, respectively, resulting in an overall accuracy of 95%.

1. Click the button "Test 3" to test the VGG model, and the detection image and results are shown in the following figure.





It can be seen that the VGG16 network structure achieved good training results on the MINST dataset and good testing results on the self-made dataset, with accuracies of 100% for both, resulting in an overall accuracy of 100%.

From this, it can be seen that the U-Net has different training effects on different datasets. The U-Net has better training results on the MNIST dataset but poorer results on self-made datasets. In contrast, ResNet18 and VGG16 have almost the same training effects on different datasets, so we believe that ResNet18 and VGG16 have better adaptability to different datasets.

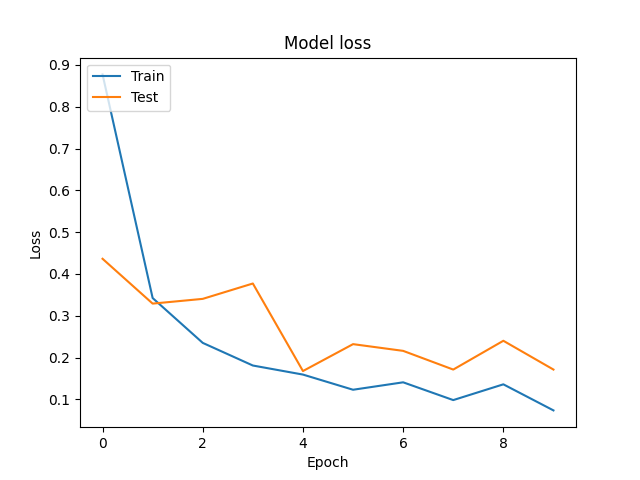
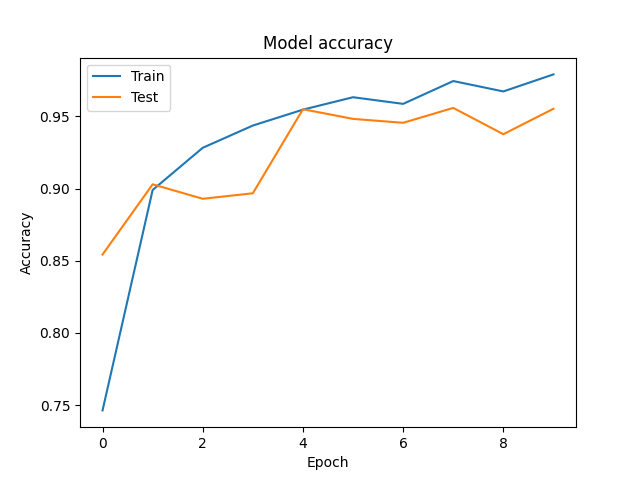
1. **Improvements**

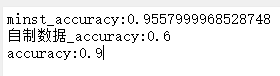
**Based on some possible issues mentioned earlier, I have made the following improvement plan, and the results are as follows.**

1. **Without changing the type of input images for the original models, i.e., ResNet18 still maintains a 28\*28\*1 grayscale image input and VGG16 maintains a 48\*48\*3 RGB image input, the test data is expanded to the entire MNIST test dataset. In the following training and testing, the test results of the U-Net network structure were not as good as those of ResNet18 and VGG16, so the U-Net results will not be further discussed, only the test results of ResNet18 and VGG16 will be compared.**

**1）The result of Resnet18**

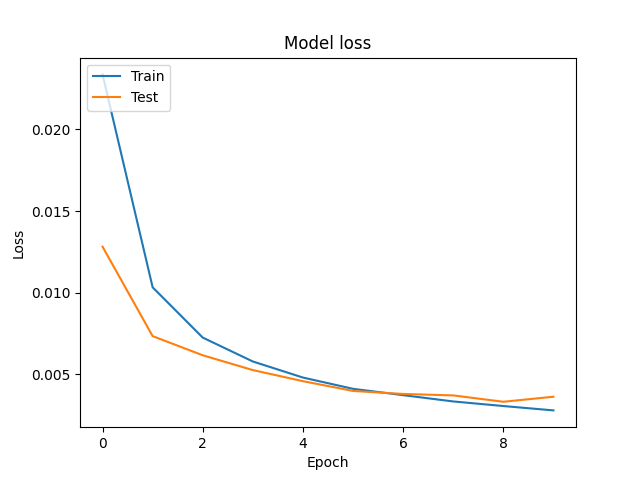
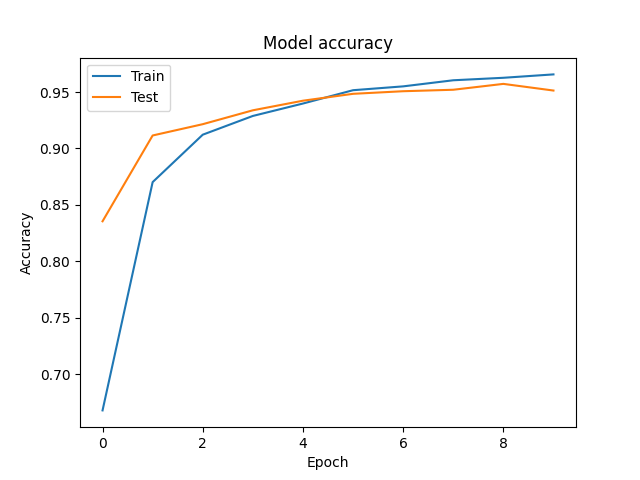


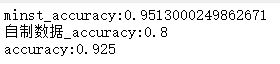




1. **The result of VGG16**





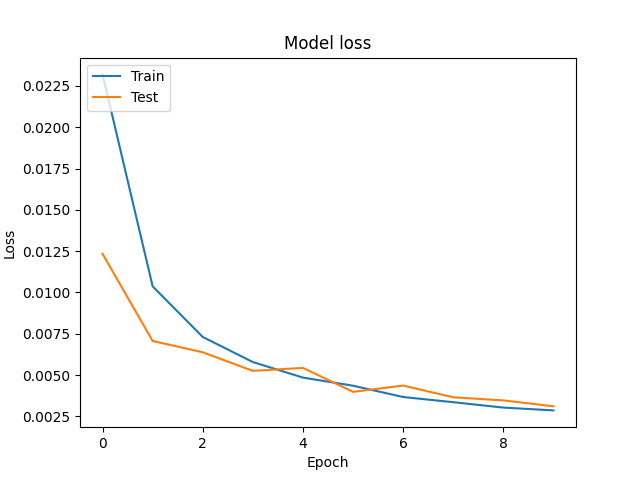
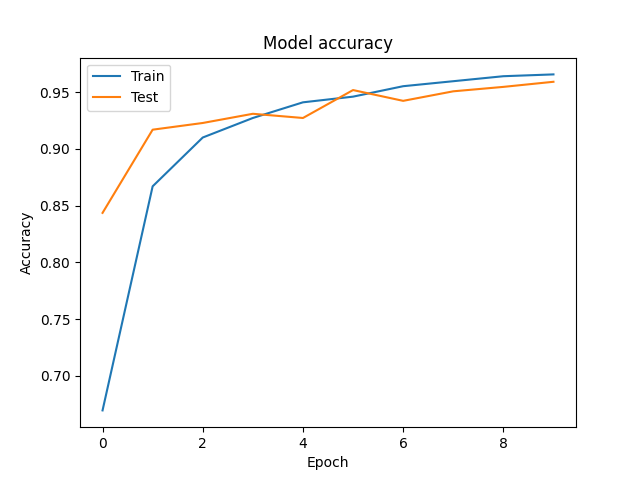


**From the above results, it can be seen that when the test set is expanded to the entire MNIST test set, the recognition accuracy of the ResNet18 network has exceeded that of the VGG16 network, while for the self-made dataset, the recognition performance of ResNet18 is still not as good as VGG16.**

**2、Further, the input image type of Unet and Resnet18 was also modified to 48\*48\*3 RGB image input, with the test data remaining as the entire MNIST test dataset. The test results are shown below.**

**1）The result of VGG16**



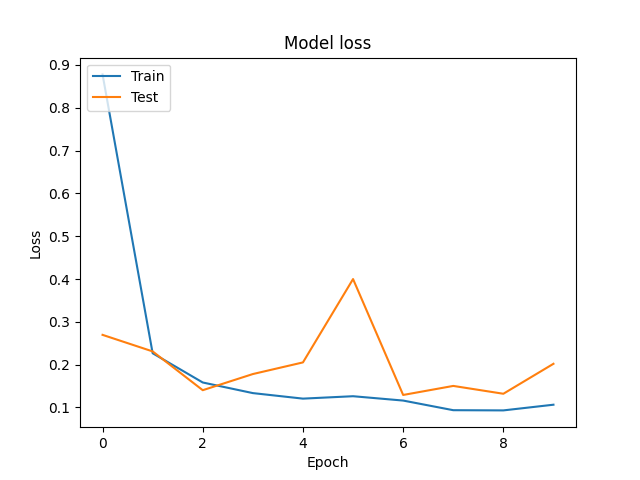
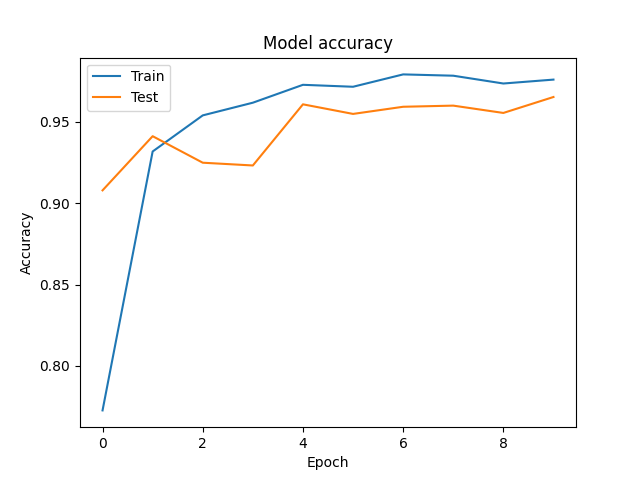




**It can be seen that the recognition accuracy of VGG16 on the mnist test set is around 95.91%, and it performs well in recognizing self-made datasets, achieving a recognition accuracy of 0.9.**

**2）The result of Resnet18**







**It can be seen that the Resnet18 network's test results have greatly improved after changing the input image type to 48\*48\*3 RGB images. The recognition accuracy on the mnist test set reached 96.67%, which is a 1% improvement compared to when using 28\*28\*1 grayscale images, and also 1% better than VGG16. The recognition accuracy on self-made datasets also increased from 60% to 80%, which means only two digits were misclassified, while VGG16 misclassified only one digit. The gap in recognition accuracy between the two models has been reduced, but Resnet18 is still not as good as VGG16.**

**Through the above improvements and re-experiments, we believe that under the same input and test set conditions, the recognition accuracy of Resnet18 is higher than that of VGG16. There may be several possible reasons for the difference in recognition performance on the self-made handwritten digit dataset compared to the MNIST dataset:**

1. **The self-made handwritten digit dataset has a smaller sample size compared to the MNIST dataset, which may lead to larger errors.**
2. **Due to differences in writing habits, the self-made handwritten digit dataset may have significant differences from some digit images in the MNIST dataset, which may result in Resnet18, which has better performance on the MNIST test set, being less effective than VGG16 at recognizing the self-made handwritten digit dataset.**

**However, overall, Resnet18 achieved improved recognition accuracy for handwritten digit recognition compared to VGG16 by adding residual network structures and increasing training parameters.**

**四、Project Summary**

The difficulty of handwriting recognition itself in this project is not high, but there were many aspects that were not fully considered during the experiment, resulting in less than ideal results during the presentation. After communicating with the teacher and listening to the suggestions, I had some ideas for improving the project and achieved some good results after some attempts. Through this project, I realized that I still have shortcomings in my ability to discover and solve different problems, and my grasp of the combination of theory and practice is not yet sufficient. Therefore, I am unable to make a sufficiently accurate analysis of the experimental results, which is an area that I need to continue to work on in the future.