**Task 2**

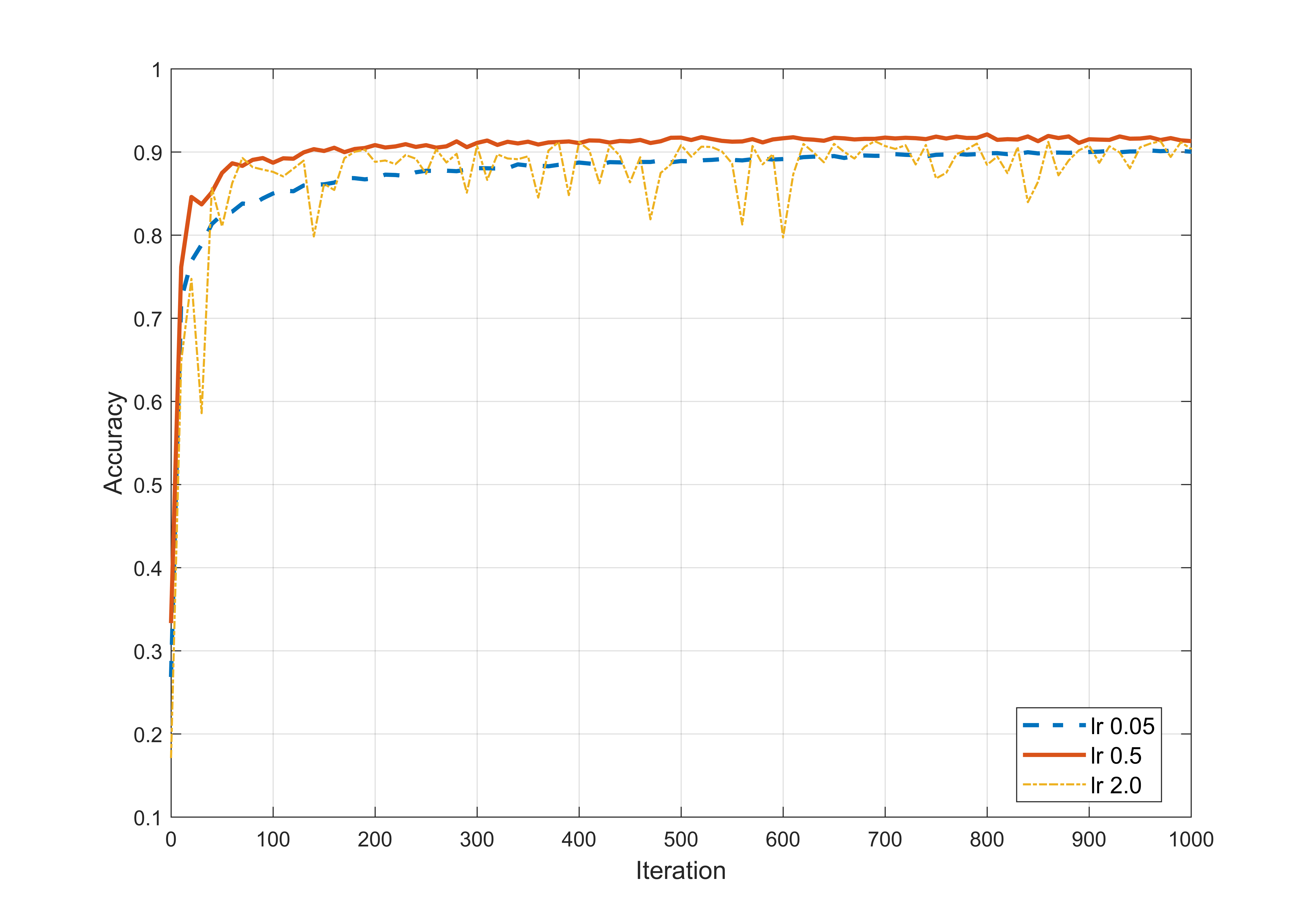


Figure 1. Accuracy vs. different learning rate

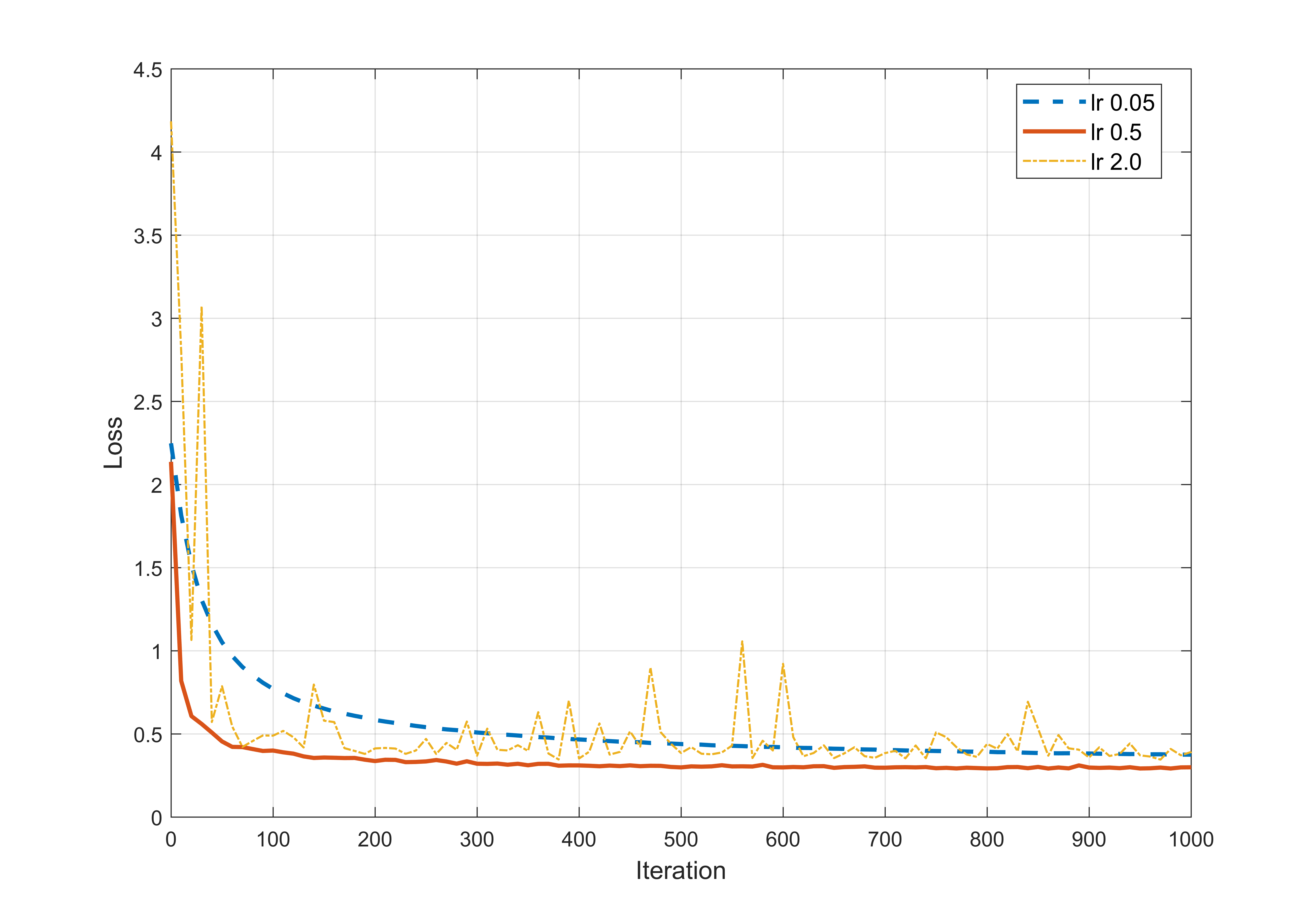


Figure 2. Loss vs. different learning rate

Figure 1 and 2 show how accuracy and loss change as iteration goes under different **learning rate** settings (tested on whole dataset every 10 iteration). Meanwhile other hyper-parameters are the same: batch size is 100; no hidden layer (only an input layer of 784 neurons and an output layer of 10 neurons).

After 1000 iterations, learning rate of 0.05 get 90.03%, learning rate of 0.5 get 91.39% and learning rate of 2.0 get 90.37%. According to figure 1 and figure 2, high learning rate (2.0) will result in strong oscillation and make it hard to converge, while low learning rate (0.05) need more time to learn things. Based on these plots, it’s fair to say the accuracy can still go up with low learning rate even after 1000 iterations. With medium learning rate (0.5), accuracy stays around 91% after 500 iterations.

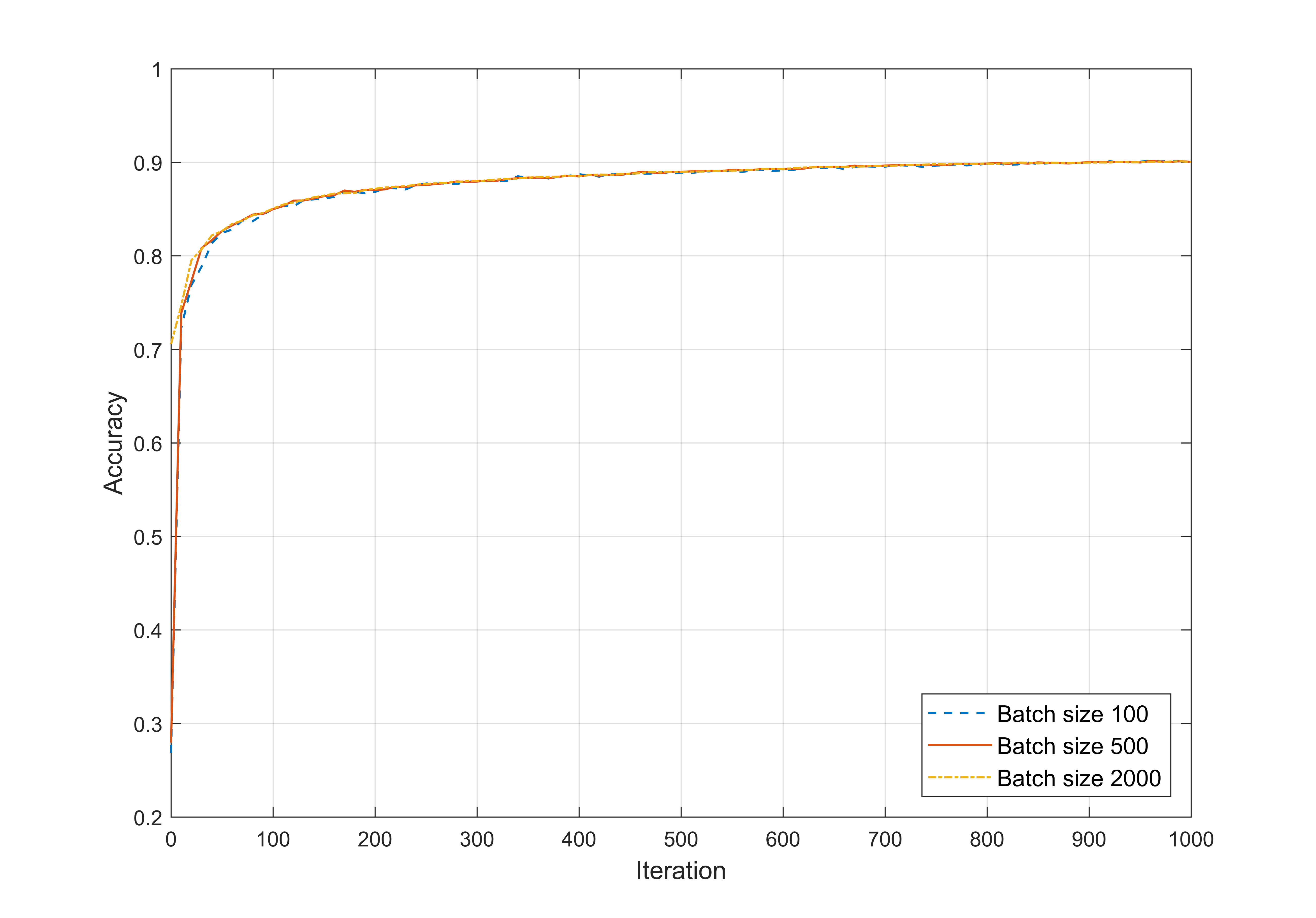


Figure 3. Accuracy vs. different batch size

Figure 3 shows how accuracy change as iteration goes under different **batch size** settings (tested on whole dataset every 10 iteration). Meanwhile other hyper-parameters are the same: learning rate is 0.05; no hidden layer (only an input layer of 784 neurons and an output layer of 10 neurons).

After 1000 iterations, they all get about 90% accuracy. And their curves almost overlap, so in this case it’s hard to tell how batch size affect accuracy and convergence.

Since it seems like changing learning rate and batch size doesn’t help increasing accuracy in this case, I tried to add **hidden layer** between input layer and output layer.

C:\Users\czeng8\AppData\Local\Microsoft\Windows\INetCache\Content.Word\add_layer.tif

Figure 4. Accuracy of 784-300-10 network

Figure 4 shows how accuracy changes as iteration goes after adding an additional hidden layer of 300 neurons. Learning rate is 0.1 and batch size is 100.

After 20,000 iterations, it get 93.12%. It gets a slightly higher accuracy than before, but it takes much more time to converge.

Besides these hyper-parameters, there is another important issue regarding training neural network: how parameters are initialized. I noticed that the original code sets them to zero at beginning, which is not good. So I decided to use Gaussian distribution to initialize them.

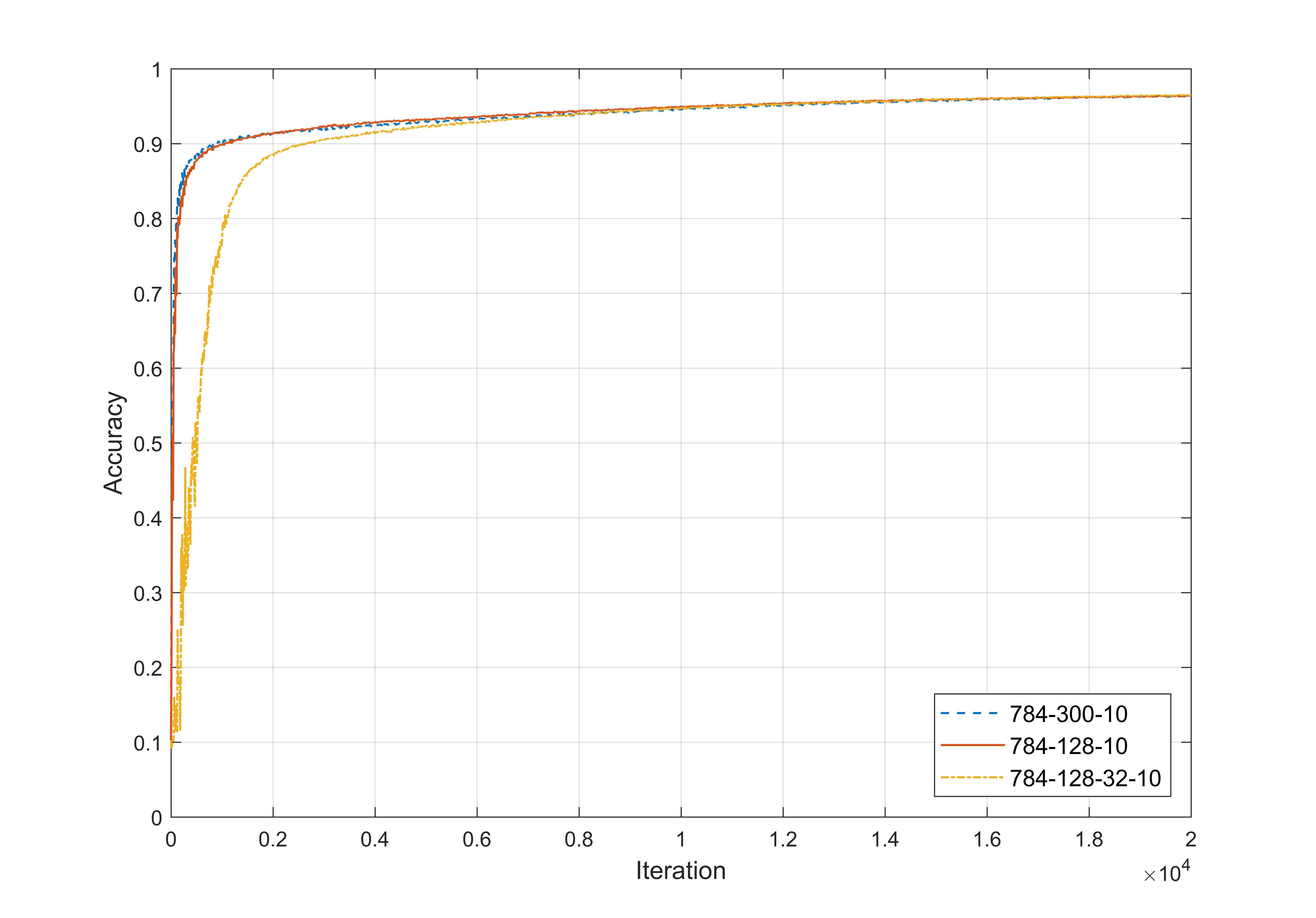


Figure 5. Accuracy of Gaussian distribution initialized networks

3 different networks were tested using Gaussian distribution initialization, whose mean is 0 and standard deviation is 0.1. And learning rate is set to 0.1.

After 20,000 iterations, they get 96.04%, 96.33% and 96.40%. These results are much better than before, which means using Gaussian distribution is better than setting to zero. Compared with figure 4, these curves don’t suffer from early oscillation and just go up at the very beginning. Besides, there’s another point: 2 hidden layers network learns slower than 1 hidden layer network, but it can achieve higher accuracy at last.

What’s more, based on this figure, it seems like learning rate of 0.1 might be too small because it looks like accuracy is still going up even after 20,000 iterations. So I tried 0.5 and I got accuracy of 97.70% with that 4-layer network.

**Task 3**

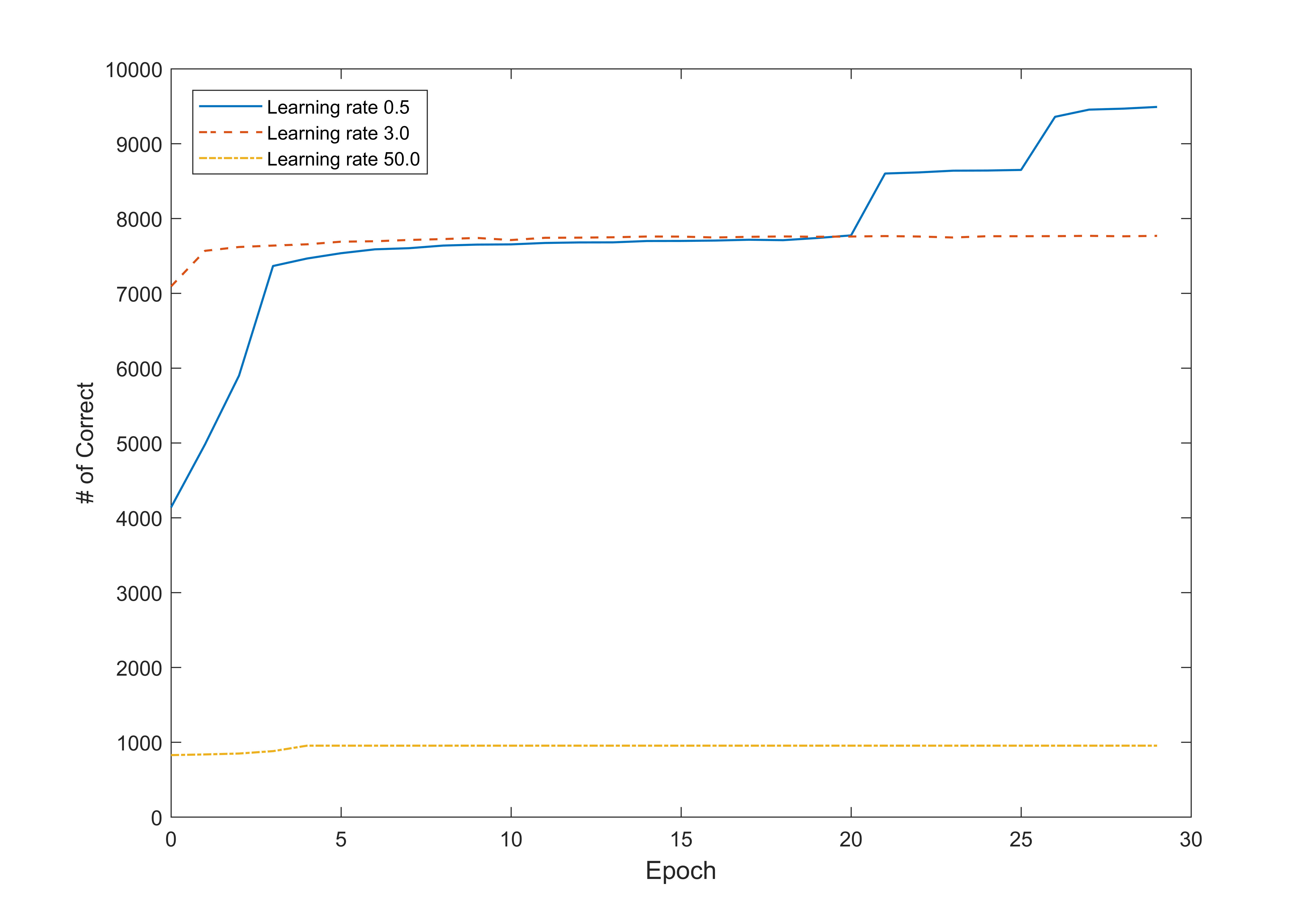


Figure 6. Different learning rates

Figure 6 shows how the number of correct classified image changes as epoch goes up under different learning rates. Three layers network, with a hidden layer of 128 neurons, is used in this test, and batch size is 10.

High learning rate as 50 leads to failure to learn; learning rate of 3.0 works well at first but stuck into local optima later on; learning rate of 0.5 learns slowly but it will jump out from local optima, which gives it the best result out of three, 9490 of 10000 are correct.

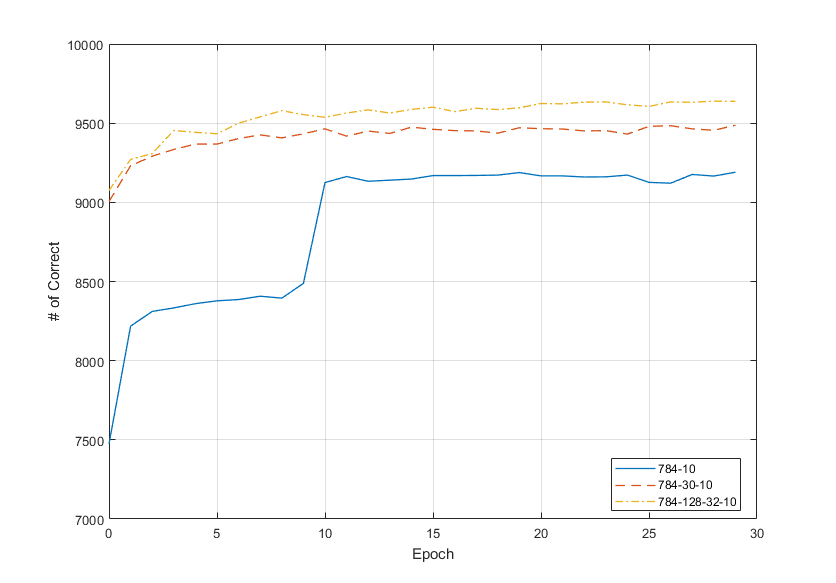


Figure 7. Different network structures

Figure 7 shows the effects of different structure of networks. After 30 epochs, two-layer network (784-10) got 9190 correct, three-layer network (784-30-10) got 9487 correct, and four-layer network got 9638 correct, which can be regarded as that network with more layers can do better than shallow network with less.