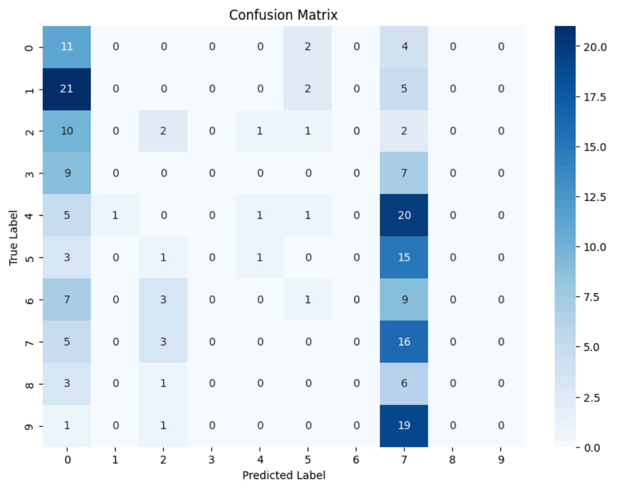
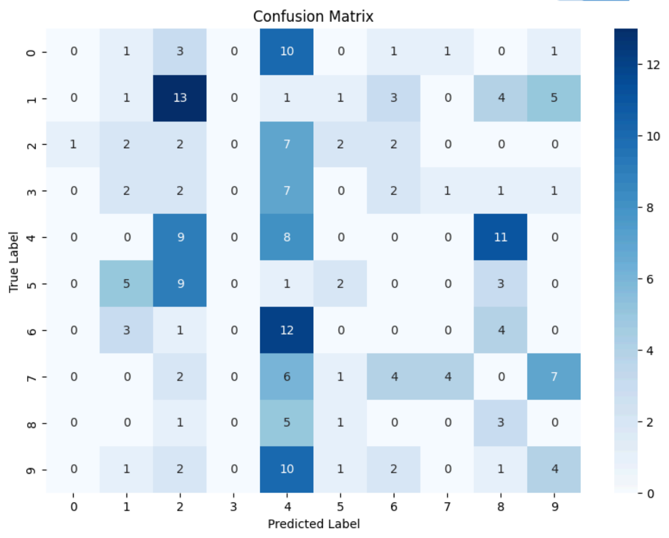
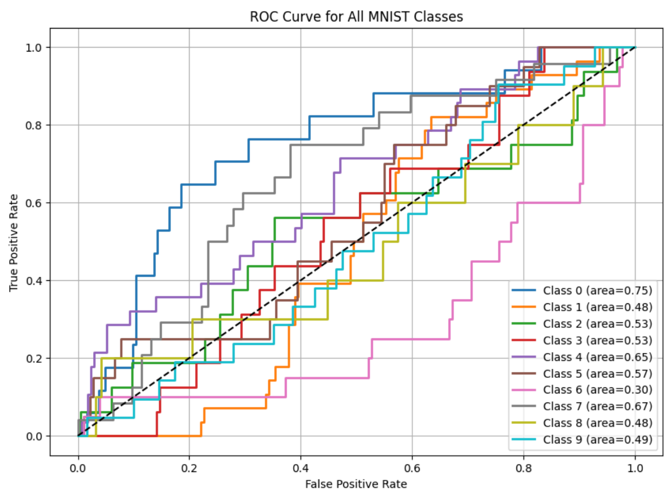
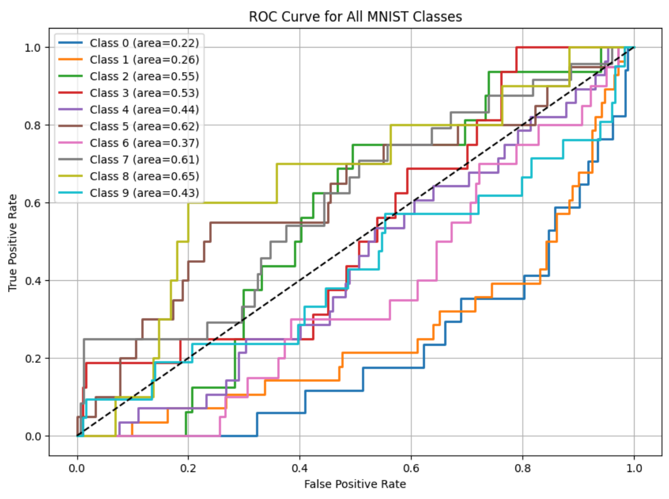
Put Your Code Results Here:

First test	**Second time test**
	
The number 1 is easily predicted to be a 0, possibly because certain characteristics of handwritten digits 1 and 0 are easily confused. The number 7 was also more incorrectly judged as other numbers.	The number 4 is an obvious center of confusion, with many numbers easily mistaken for four. The numbers 7 and 9 are also somewhat confused with each other.
	
input_size = 784 hidden1 = 64 hidden2 = 32 output_size = 10 **Use `ReLU` For activation Function**	**input_size = 784 hidden1 = 128 hidden2 = 64 hidden3 = 32 output_size = 10** **Use `tanh` For activation Function**

```
              precision    recall  f1-score   support

           0     0.1467    0.6471    0.2391        17
           1     0.0000    0.0000    0.0000        28
           2     0.1818    0.1250    0.1481        16
           3     0.0000    0.0000    0.0000        16
           4     0.3333    0.0357    0.0645        28
           5     0.0000    0.0000    0.0000        20
           6     0.0000    0.0000    0.0000        20
           7     0.1553    0.6667    0.2520        24
           8     0.0000    0.0000    0.0000        10
           9     0.0000    0.0000    0.0000        21

    accuracy                         0.1500       200
   macro avg     0.0817    0.1474    0.0704       200
weighted avg     0.0923    0.1500    0.0714       200
```

```
              precision    recall  f1-score   support

           0     0.0000    0.0000    0.0000        17
           1     0.0667    0.0357    0.0465        28
           2     0.0455    0.1250    0.0667        16
           3     0.0000    0.0000    0.0000        16
           4     0.1194    0.2857    0.1684        28
           5     0.2500    0.1000    0.1429        20
           6     0.0000    0.0000    0.0000        20
           7     0.6667    0.1667    0.2667        24
           8     0.1111    0.3000    0.1622        10
           9     0.2222    0.1905    0.2051        21

    accuracy                         0.1200       200
   macro avg     0.1482    0.1204    0.1058       200
weighted avg     0.1636    0.1200    0.1114       200
```

Adding one more layer (32 neuro units) makes the neural network deerper. And change activation function from ReLU to tanh but look like doesn't make ACC value higher

---

1. Explain how you designed your model (number of layers, neurons, and activation functions).
What changes did you make to improve the accuracy, and how did those changes affect the results?
*Please attach the performance results before and after your improvements.

At first, I made my model with a pretty simple structure. It had an input layer with 784 neurons, then two hidden layers with 64 and 32 neurons, and an output layer with 10 neurons. For the activation functions, I used ReLU in the hidden layers and Softmax in the output layer.

To make the accuracy better, I tried some changes. I increased the number of neurons in the hidden layers to 128 and 64, and I even added one more hidden layer with 32 neurons. I also changed the activation function from ReLU to tanh because I thought it might help the model find more complex patterns.

But the changes didn't work as I hoped. Surprisingly, the accuracy went down from 15% to 12%. I think this happened because the model wasn't trained with backpropagation and was just using random weights. Adding more layers just made the model more complicated without actually learning anything useful. The extra layers probably just added more noise instead of helping the model recognize the digits better.

2. Based on your evaluation results (confusion matrix, ROC, etc.), how well did the model perform? Which classes are harder to predict? Why do you think that happened?

Honestly, the model didn't do well at all. The accuracy was really low, only around 12–15%, which is just a little bit better than random guessing. The confusion matrix shows that the model had a hard time telling apart most of the numbers. For example, it often got confused with 3, 6, 8, and 9—they were almost always guessed wrong, and the precision and recall were close to zero.

The main problem is that the model didn't really learn how to recognize the numbers because it was just using random weights without training. That's why it mostly guessed instead of making accurate predictions. Also, some numbers like 1 and 7 were often mistaken for 0. I think this is because handwritten 1 and 7 can look a bit like 0 if written in a certain way, which makes the model confused.

One thing that stood out is that the numbers 0 and 9 were recognized a bit better than the others. I think it’s because 0 has a unique round shape, and 9 often has a distinct tail, so they are a bit easier to tell apart even when using random weights.

Overall, just adding more layers without training didn’t help—in fact, it made things worse. The model needs proper training with backpropagation to actually learn the features instead of just guessing randomly.