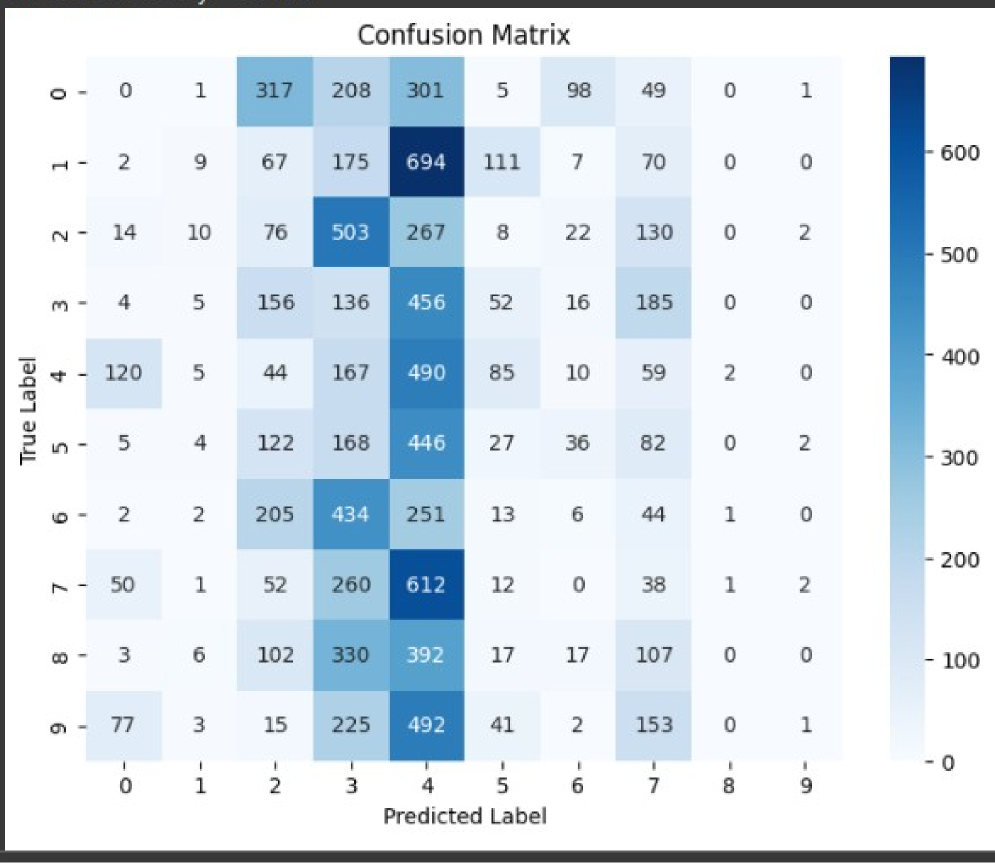
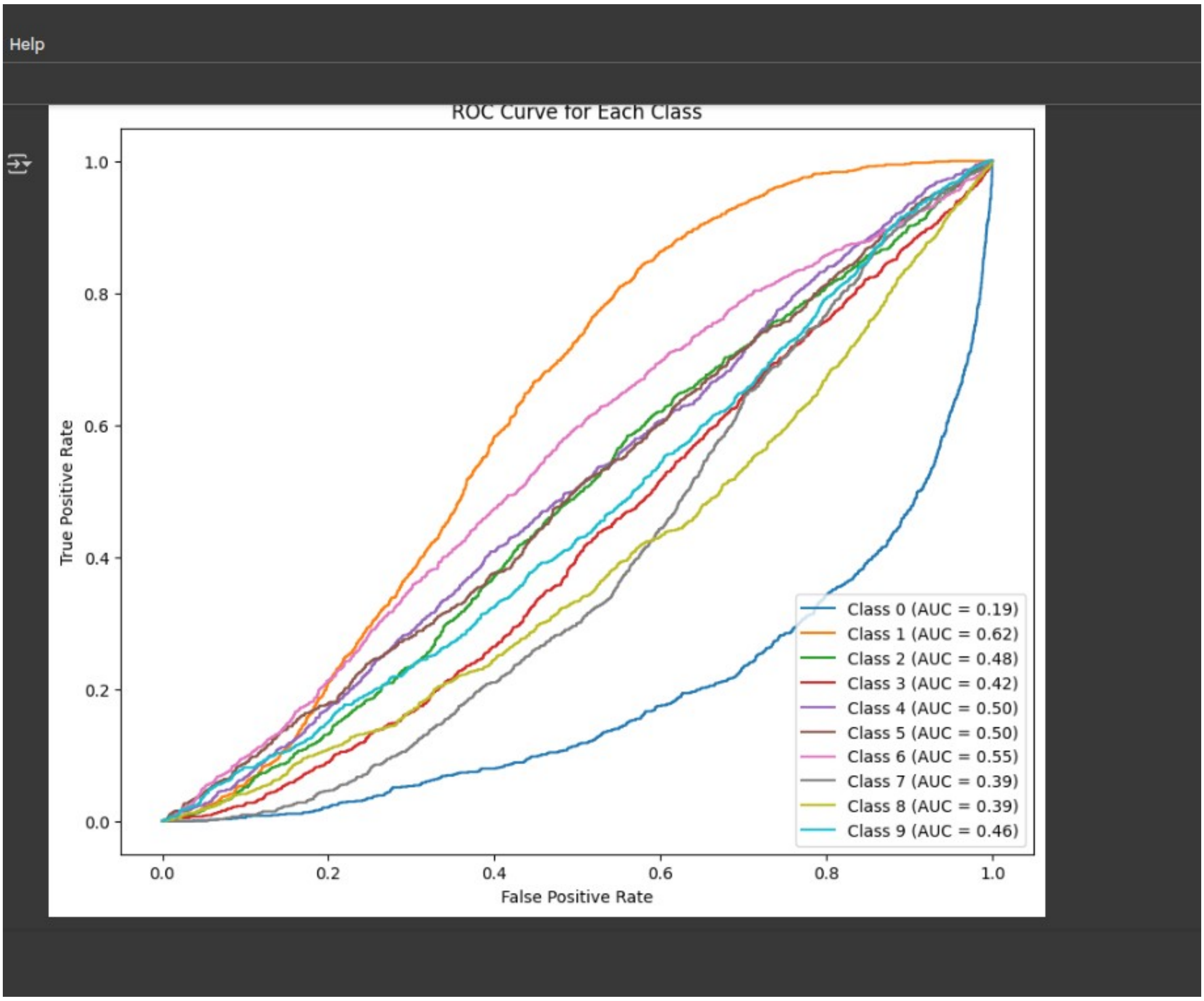
Lab3

Results:

```
Epoch 30/30, Loss: 2.3026
Class 0: TP=0, TN=8743, FP=277, FN=980
       Precision=0.0000, Recall=0.0000, F1=0.0000, Accuracy=0.0000
Class 1: TP=9, TN=8828, FP=37, FN=1126
       Precision=0.1957, Recall=0.0079, F1=0.0152, Accuracy=0.0009
Class 2: TP=76, TN=7888, FP=1080, FN=956
       Precision=0.0657, Recall=0.0736, F1=0.0695, Accuracy=0.0076
Class 3: TP=136, TN=6520, FP=2470, FN=874
       Precision=0.0522, Recall=0.1347, F1=0.0752, Accuracy=0.0136
Class 4: TP=490, TN=5107, FP=3911, FN=492
       Precision=0.1113, Recall=0.4990, F1=0.1821, Accuracy=0.0490
Class 5: TP=27, TN=8764, FP=344, FN=865
       Precision=0.0728, Recall=0.0303, F1=0.0428, Accuracy=0.0027
Class 6: TP=6, TN=8834, FP=208, FN=952
       Precision=0.0280, Recall=0.0063, F1=0.0102, Accuracy=0.0006
Class 7: TP=38, TN=8093, FP=879, FN=990
       Precision=0.0414, Recall=0.0370, F1=0.0391, Accuracy=0.0038
Class 8: TP=0, TN=9022, FP=4, FN=974
       Precision=0.0000, Recall=0.0000, F1=0.0000, Accuracy=0.0000
Class 9: TP=1, TN=8984, FP=7, FN=1008
       Precision=0.1250, Recall=0.0010, F1=0.0020, Accuracy=0.0001

Overall Accuracy: 0.0783
```





QA:

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?

Model structure:

Input Layer: 784 neurons (each MNIST image is 28x28 pixels)

Hidden Layer 1: 128 neurons, using ReLU activation function

Hidden Layer 2: 64 neurons, using ReLU activation function

Output Layer: 10 neurons, using Softmax activation function

Improvements:

From the results, the accuracy of the model is very low (about 7.83%), indicating that the network failed to effectively learn the MNIST data. Therefore, possible directions for improvement include:

Increase the number of layers and neurons: The current network is shallow and may not be sufficient to learn the features of MNIST.

Use Batch Normalization or Dropout: This can help stabilize training and prevent overfitting.

Improve weight initialization: currently uses np.random.randn() * 0.01, may need a better initialization method such as Xavier or He initialization.

Adjust learning rate: Currently use learning_rate = 0.01, maybe need to test a more appropriate learning rate.

Use a more advanced optimizer: Currently not using Adam or RMSprop, just plain gradient descent.

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?

(1) Confusion Matrix Analysis

The confusion matrix shows that many numbers are misclassified, especially some categories where the data is highly concentrated in a few wrong classifications. For example:

Class 1 is often misclassified as Class 4.

Class 4 is often misclassified as Class 3 or Class 5.

Class 7 is often misclassified as Class 4.

This could be because some of the digits have similar shapes or the model fails to distinguish these features effectively.

(2) ROC curve analysis

The AUC for Class 0 is only 0.19, indicating that this class is almost impossible to classify correctly.

The AUC for Class 1 is 0.62, which is relatively high, indicating that this class has good discrimination.

The AUCs for other categories ranged from 0.39 to 0.55, indicating poor overall classification performance with no clearly high-performance category.

(3) Which categories are more difficult to predict? Why?

Class 0 (AUC = 0.19) and Class 7, 8 (AUC ≈ 0.39) are more difficult to predict.

Possible causes:

Insufficient dataset features learned: More layers may be needed to learn MNIST features.

Underfitting: Since the network is shallow, it may not be able to learn the data effectively.

Impact of weight initialization: Improper weight initialization may lead to poor learning results.

Data imbalance: Some categories have less data, resulting in insufficient model learning.