ASSIGNMENT REPORT

Assignment 2 (NN)

Fahim Wayez
21-44499-1 | Section: B

Code/Process explanation: The NeuralNetwork class is defined in this assignment to implement a neural network for multi-class classification. It initializes the network's architecture, including the number of input neurons, hidden layer neurons, and output neurons. The learning rate is set to 0.2 for controlling the speed of weight updates during training. Weights for each layer are initialized randomly using np.random.randn().

The class defines two activation functions: sigmoid and softmax. A sigmoid activation function for hidden layers, which converts the input to a value between 0 and 1, and a softmax activation function for the output layer, which converts the raw output into class probabilities.

The cross_entropy_loss function is written to calculate the cross-entropy loss between the predicted probabilities and the true labels. It measures the difference between predicted probabilities and true class probabilities.

The feedforward method performs forward propagation through the neural network. It calculates the activation of hidden layers and the output layer using the sigmoid and softmax activation functions, respectively. And the backpropagation method updates the weights of the network based on the prediction error. It calculates the error and delta for each layer and updates the weights accordingly.

The train method trains the neural network using the given training data. It performs forward and backward propagation for a specified number of epochs, storing the training error for each epoch.

A synthetic dataset with five classes is generated using random normal distributions. Each class has a specified number of samples, and input features are generated accordingly. Then the synthetic dataset is split into training and testing sets using train_test_split. The neural network is initialized, trained using the training set, and the training error is plotted. The trained network is then tested using the testing data, and predictions are obtained. Predictions are converted to class labels, and evaluation metrics (accuracy, precision, recall, F1-score) are calculated. The confusion matrix is also plotted to visualize the model's performance across different classes.

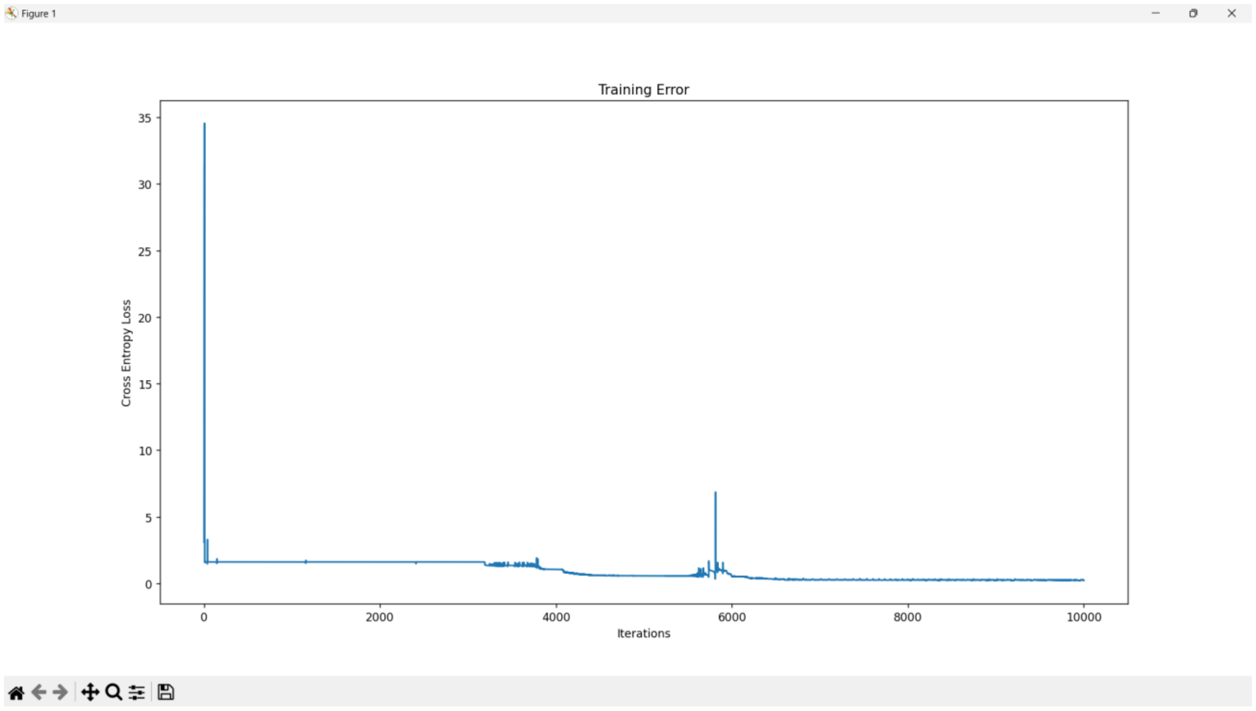


Figure 1: Training Error

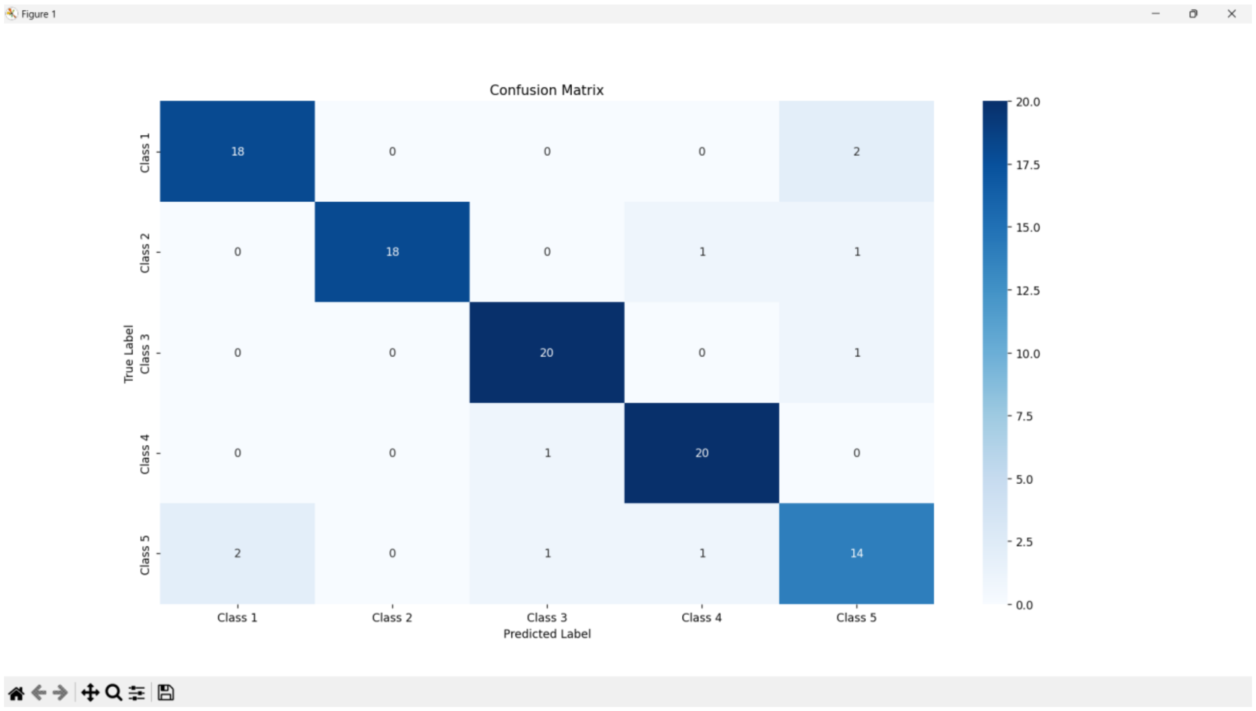
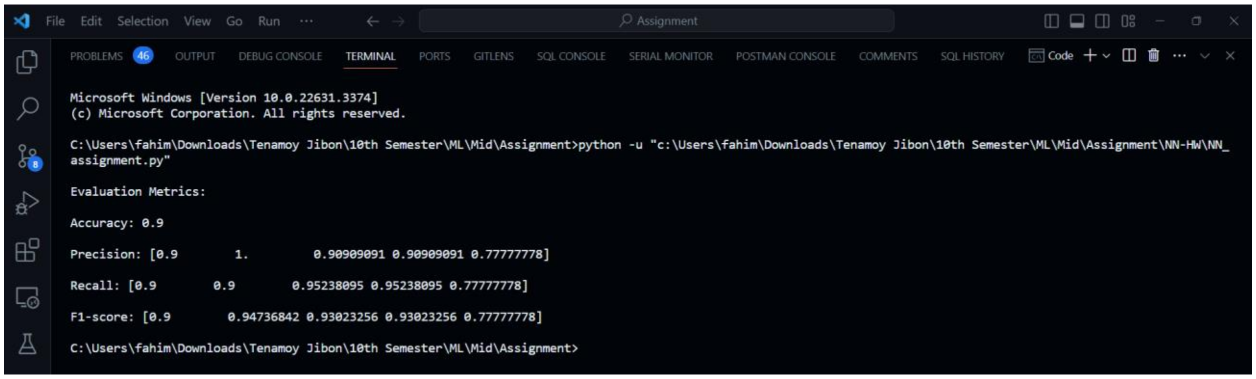


Figure 2: Confusion Matrix



```
Microsoft Windows [Version 10.0.22631.3374]
(c) Microsoft Corporation. All rights reserved.

C:\Users\fahim\Downloads\Tenamoy Jibon\10th Semester\ML\Mid\Assignment>python -u "c:\Users\fahim\Downloads\Tenamoy Jibon\10th Semester\ML\Mid\Assignment\NN-HW\NN_
assignment.py"

Evaluation Metrics:

Accuracy: 0.9

Precision: [0.9        1.         0.90909091 0.90909091 0.77777778]

Recall: [0.9        0.9        0.95238095 0.95238095 0.77777778]

F1-score: [0.9        0.94736842 0.93023256 0.93023256 0.77777778]

C:\Users\fahim\Downloads\Tenamoy Jibon\10th Semester\ML\Mid\Assignment>
```

Code Segment 1: Evaluation Metrics

Discussion: The multi-class classification model achieved an accuracy of 90%, indicating its effectiveness in classifying the synthetic dataset into five distinct classes. Precision, recall, and F1-score were calculated for each class, providing insights into the model's performance across different classes. The confusion matrix visualizes the performance of the model, showing the distribution of true positive, true negative, false positive, and false negative predictions.

Modifying the neural network architecture for multi-class classification required careful consideration of the number of neurons in the output layer and activation functions. Preparing the synthetic dataset with five distinct required understanding the data generation process and ensuring balanced representation of each class. Analyzing evaluation metrics and the confusion matrix helped in understanding the strengths and weaknesses of the model and identifying areas for improvement.

In case of potential improvements, experimenting with different learning rates, number of neurons in hidden layers, and training epochs could enhance the model's performance. Introducing variations in the synthetic dataset or incorporating real-world data could improve the model's generalization capability. Also, exploring more complex neural network architectures such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs) could lead to better performance especially for datasets with spatial or sequential dependencies.