Machine Problem: Neural Network

Bren Jay Magtalas

June 23, 2023

1 The Dataset

The MNIST handwritten digits dataset is used for training and evaluating the neural network implementation. The data, originally stored in a `.mat` format was parsed using the `scipy.io` library. A total of 70,000 samples are in the dataset with their corresponding labels. The features are represented as a 784-sized array of pixel intensity values per sample, ranging from 0 as darkest and 255 as brightest. The labels are the digit identification ranging from 0 to 9. A sample can be seen in Figure 1 with label 2, which is a 28 by 28 image visualized from resizing the feature array. Because of the large number of data, the dataset was undersampled to be just 1,000 samples per class. To randomly sample the dataset, ensuring a balanced number per digit, `RandomUnderSampler` from `imblearn.under_sampling` was utilized. With the 10 digits classes, the total resulting dataset is reduced to 10,000 samples. The labels are then one-hot encoded to be suitable for classification tasks. As a convention, the dataset is transformed in a way that the rows are features and the columns are samples, with a size of 784 by 10,000. Similarly, the label rows are one-hot encoded classes, and the columns are sample labels, with a size of 10 by 10,000. The dataset was then randomly split into 60% training, 20% validation, and 20% test set.

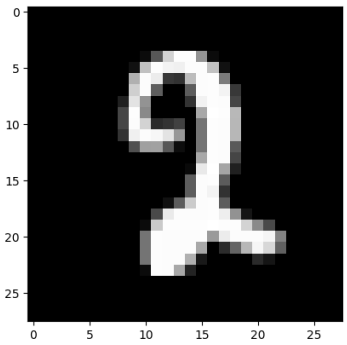

Figure 1: Samle Data with Label 2

2 Implementation

The neural network implementation is made scalable by using the concept of computational graphs. The one tested specifically for the problem is composed of 4 hidden layers with ReLU activation function and 1 output layer of Logistic. The model is trained using stochastic gradient descent, which uses random mini-batches of the data, and momentum is also utilized to speed up the training. The layers and their corresponding number of neurons are stored in a list to set the model architecture. The neural network architecture consists of multiple layers. The first layer is the input layer, which has a number of neurons equal to the input features. There are four hidden layers with a user-defined number of neurons. Lastly, there is an output layer with a number of neurons equal to the number of classes. The names of activation functions used are also specified in a list of strings with a length of less than 1 because the input layer does not have activation. The input arguments to the main MLP training function are the training set, validation set, neurons per layer list, activations list, mini-batch size, learning rate, momentum rate, and epoch.

The training starts by initializing the error in the training set and the error in the validation set to an array of zeros, which will be used for monitoring the performance of the model. Because the training is done per batches of the training set, to easily separate the data, the batch indices are obtained based on the batch size. The weights, bias, previous change in weight, and previous change in bias are then initialized. The weights are set to be random uniformly distributed small values while the rest are set to zero. In each epoch, the training data is randomized and the batch indices are utilized to loop through the entirety of data. In each batch, forward propagation is performed, storing the intermediate values calculated per layer. The error is then calculated using the binary cross entropy or log loss. The total cost and the local gradient of the output layer is passed through the backpropagation step. The backpropagtion function would then calculate the local gradient per layer. The update of weights and bias in the neural network is performed by utilizing the local gradients and the values obtained during the forward step. These values are scaled by the learning rate and adjusted based on the momentum of the previous weights and bias. The cost per batch is summed up and divided by the number of batches. The validation error using the weights and bias obtained for the current epoch is also calculated for comparison using the validation set. The training would be done when the maximum epoch is reached. The average batch cost error and validation error are recorded per epoch. The final weights and biases are also returned. The model is then evaluated using the test set.

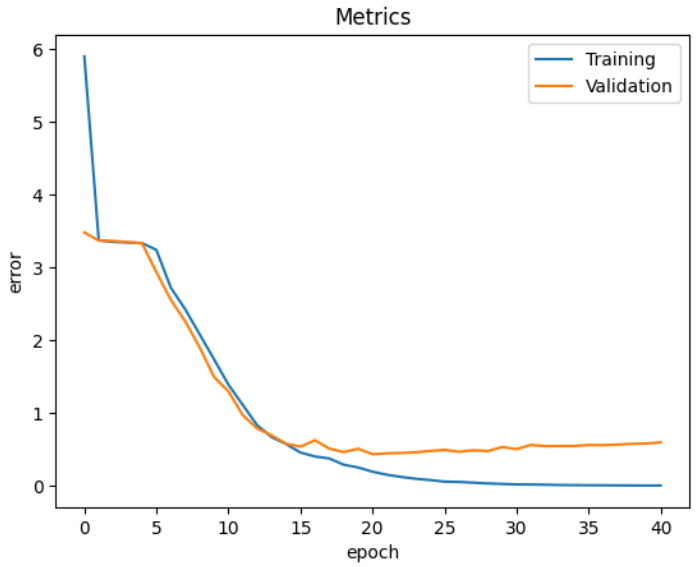

Figure 2: Error Plot

3 Hyperparameter Tuning

The hyperparameters were initially set to a very small number, 0.01 for the learning rate, and 0.9 for the momentum rate. The number of epochs is set to 50 iterations and the batch size is set to 8. The number of nodes in the hidden layers was selected based on the convergence of the error plots and the confusion matrix. At the start, the number of nodes is set to be the same as the number of output nodes. The confusion matrix shows the number of misclassified data that served as a baseline when adjusting the number of neurons. In tuning the hyperparameter, the general shape of the error plot is observed. The initial learning rate proved to be too big for the data because of the instability in the error plots. Setting the learning rate to 0.0009 smoothened the curve without sacrificing a lot of training time duration. The momentum rate of 0.9 is chosen and maintained based on the favorable characteristics observed in the plot, indicating its effectiveness. Decreasing the nodes in the hidden layer resulted in an increase in misclassified data. Increasing the number of nodes decreases the misclassification but only up to a certain point. The discovered optimal configuration consists of four hidden layers with 256 neurons in the first layer, 128 neurons in the second layer, 64 neurons in the third layer, and 32 neurons in the fourth layer. After testing various batch sizes, it was determined that a batch size of 64

offered a favorable balance between training duration and classification performance. It was observed that as the number of epochs increased, the plots of training and validation error produced a greater divergence. The epoch was reduced to 40 to prevent overfitting the training data. The error plot is shown in Figure 2. The error plot reveals an initial rapid decrease followed by a plateau, occurring around 1-5 epochs. This observation suggests that the network may have encountered a local minimum during this period, leading to the flattened error curve. It was followed by a gradual decrease in error until the curves diverge around 15 epoch.

4 Evaluation Metrics

The generalization performance of the model is measured using the test set. Accuracy is the ratio of correctly predicted samples and the total number of samples. It indicates how well the model can classify positive and negative classes. Precision is the ratio of correctly predicted positive samples and the total number of predicted positive samples. It indicates how well the model deal with false positives. Recall is the ratio of correctly predicted positive samples and all actual positive samples. It indicates how well the model deal with false negatives. The F1 score is the harmonic mean of precision and recall. It indicates how well the model deal with both false positive and false negative. The confusion matrix is in Figure 3 and the performance metrics are summarized in Table 1.

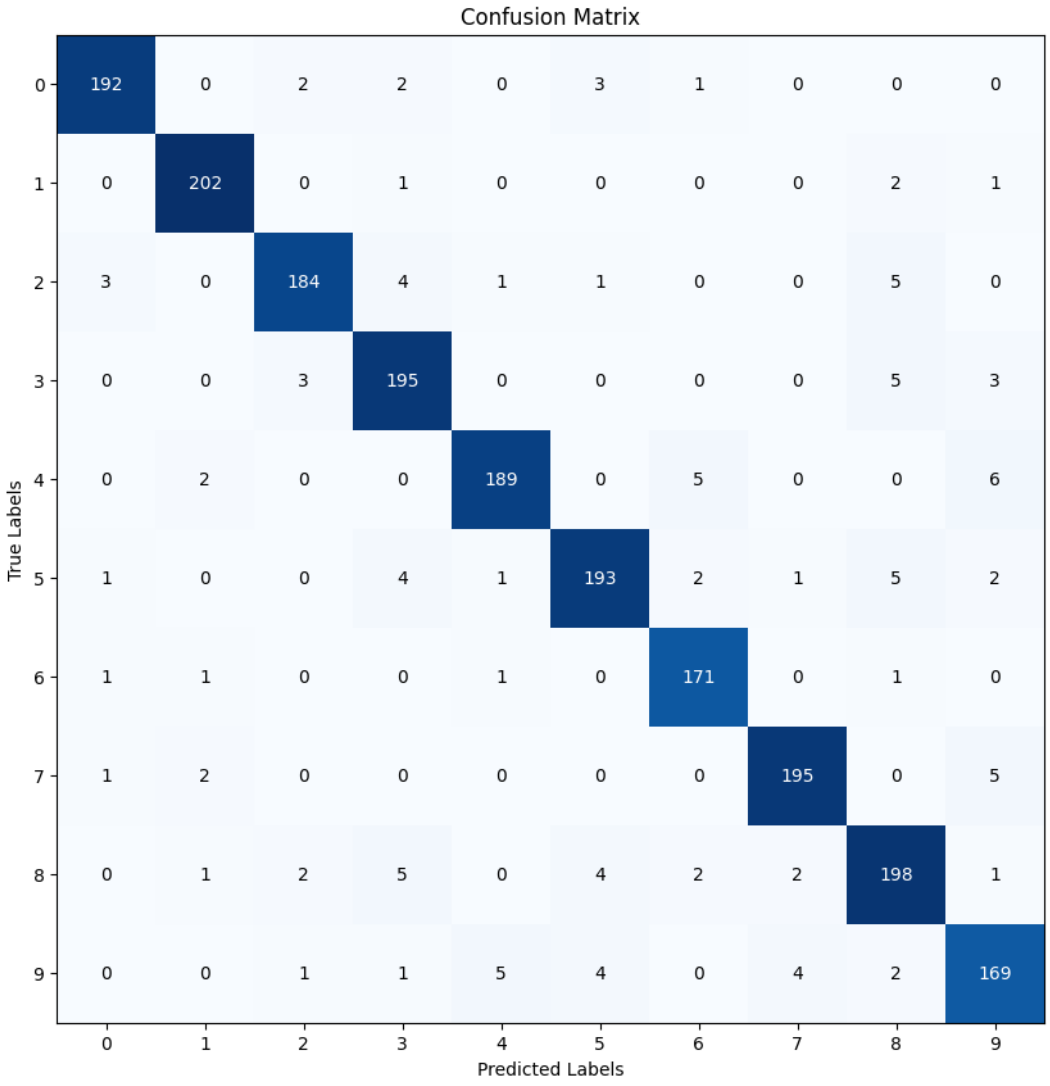

Figure 3: Confusion Matrix

Metrics	0	1	2	3	4	5	6	7	8	9	Avg
Accuracy	0.993	0.995	0.989	0.986	0.990	0.986	0.993	0.993	0.982	0.983	0.989
Precision	0.970	0.971	0.958	0.920	0.959	0.941	0.945	0.965	0.908	0.904	0.944
Recall	0.960	0.981	0.929	0.947	0.936	0.923	0.977	0.961	0.921	0.909	0.944
F1-Score	0.965	0.976	0.944	0.933	0.947	0.932	0.961	0.963	0.915	0.906	0.944

Table 1: Performance Metrics

Among all the class labels, the digit 9 yields the least favorable performance. Both precision and recall are approximately 90%. As evident from the confusion matrix, numbers like 4 and 7 are occasionally misclassified as 9, and vice versa. The precision of digit 8 is also only around 90%, with numbers 5, 3, and 2 sometimes misclassified as 8. The model produced an overall good performance based on the average of all the metrics.

5 Conclusion

A neural network model with 4 hidden layers was trained using MNIST digits dataset to classify handwritten number images and produced a good classification performance. The learning rate and momentum rate are 0.0009 and 0.9 respectively. Although the alpha is very small, the training time is not affected much because of the momentum term. The optimal number of neurons is tested to be 256 for the first, 128 for the second, 64 for the third, and 32 for the fourth hidden layer. Increasing the number of neurons provides no significant improvement in the model and decreasing it would make the model less accurate. A batch size of 64 provided balance in training time and accuracy. To prevent overfitting or significant divergence between training and validation error, it is found that a suitable number of epochs is 40. The recall for the digit 9 is the lowest, while both digits 9 and 8 yield the lowest precision. Overall, the model achieved an excellent classification performance when considering the average metrics for each class. The neural network demonstrated a strong ability to identify the majority of the digits in the test set, indicating its good generalization.