Machine Learning Principle HW4

Tianqi Li tl763

December 2024

1.1 Processing Dataset

From the DIGIT datasets the generation of 7x12 bitmap images using Nearest-neighbor interpolation. With the given images having only binary values and noise-free, there is no need for further image process approach like grayscaling or thresholding.

First of all, simplicity is a great factor, so I choose the Nearest-Neighbor interpolation. The resizing method offers computational efficiency compared to bilinear or bicubic interpolations. This is the reason in this case that used the Nearest-Neighbor interpolation.

1.2 Architecture

The architecture that the code uses is the LeNet-5 model, Consisting two convolutional layers followed by two pooling layers and three fully connected layers. The first convolutional layer takes a single-channel grayscale input and applies 6 filters of size 5×5, followed by batch normalization, a ReLU activation function, and a max-pooling operation with a 2×2 window to reduce spatial dimensions. The second convolutional layer applies 16 filters of size 5×5, followed by similar normalization, activation, and pooling operations. After the convolutional layers, the output is flattened and passed through three fully connected layers: the first with 120 neurons, the second with 84 neurons, and the final output layer with 10 neurons, corresponding to the number of classes in the MNIST dataset. ReLU activations are used after the first two fully connected layers to introduce nonlinearity, enabling the network to learn complex patterns.

1.3 Performance Evaluation

Performance evaluation of the model includes plotting the error rate across epochs, generating a confusion matrix, and analyzing the most confusing misclassified examples.

1.3.1 Error rate

The error rates over the 10 epochs highlight the model's learning progress displays a good performance of the model. Initially, the training error rate starts higher at around 4.5 percent and rapidly decreases as the model learns from the training data, dropping below 1 percent by the 5^{th} epoch. By the 10^{th} epoch, the training error reaches a very low value, indicating that the model fits the training data effectively. The testing error rate, on the other hand, starts at approximately 1.5 percent and follows a similar downward trend. However, it exhibits some minor fluctuations after the 6^{th} epoch, stabilizing around 0.75 percent by the 10^{th} epoch.

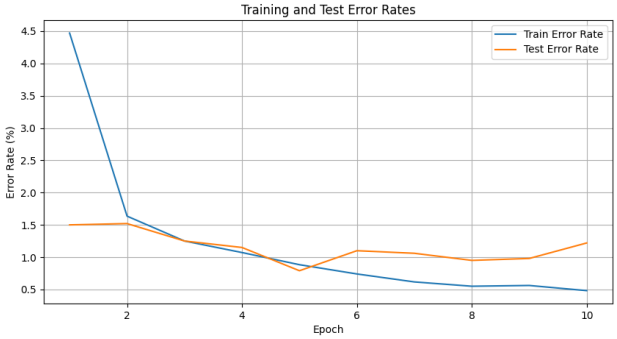


Figure 1: Error Rate in 10 Epochs Processing the Training and Testing Dataset

1.3.2 Confusion Matrix

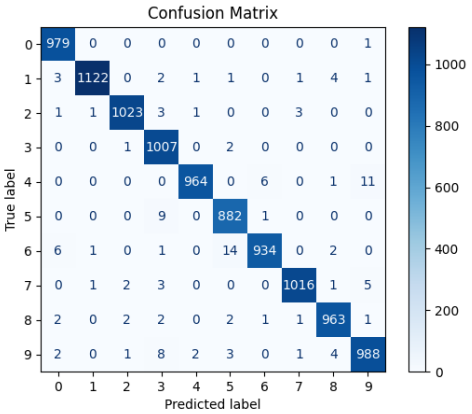
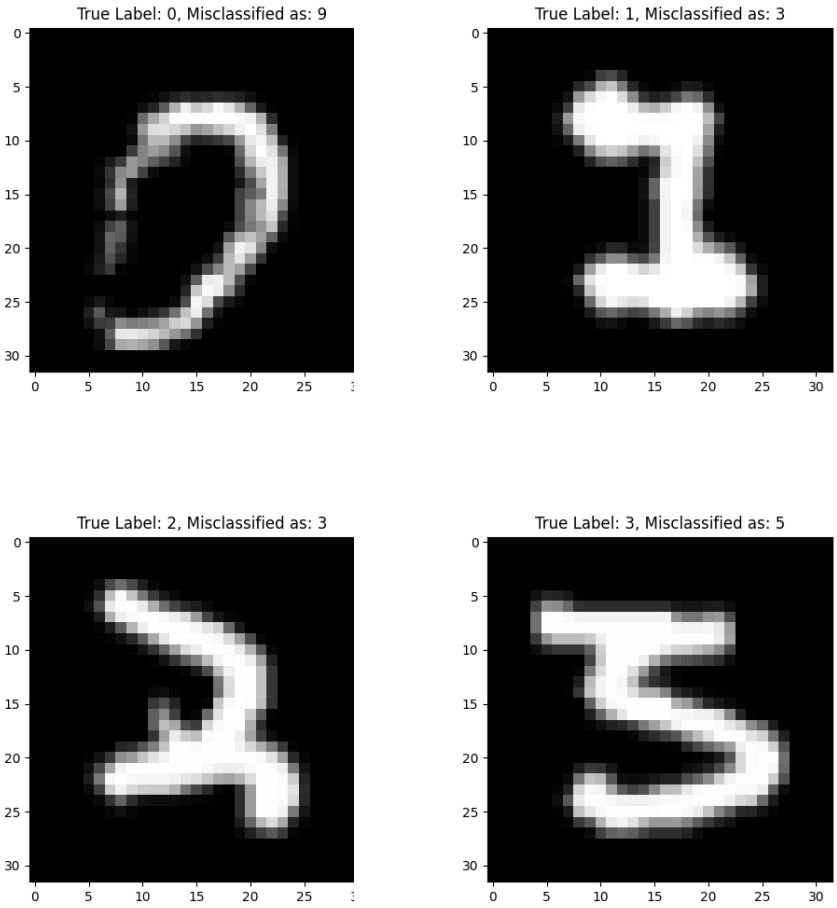


Figure 2: Confusion Matrix

Overall, the model performs well, with most predictions concentrated along the diagonal. The model shows strong performance for digits such as “0” and “7,” with 979 and 1016 correct predictions, respectively, and very few misclassifications. On the other hand, some digits, like “5” and “8,” present more challenges. For instance, 14 instances of the digit “5” were misclassified as “6” and 9 instances as “3” Similarly, digit “8” was misclassified as “2” “3” and “9” multiple times, suggesting that these digits might share similar features in the dataset, leading to confusion.

1.3.3 Most Confusing Example

The most confusing examples highlight instances where the model misclassified digits with high confidence, providing insights into its weaknesses. Below are the most confusing examples for selected digits:



1.3.4 20 Epochs

From the 20 Epochs the model continued showing a decreasing tendency for training error rate, but the testing error rate remain around 1 percent after the 6^{th} epoch. The differences between the training error and testing error displaying a potential risk of overfitting. But since the model used a large dataset of 60000 images, the overfitting will not cause drastic influence on the testing error.

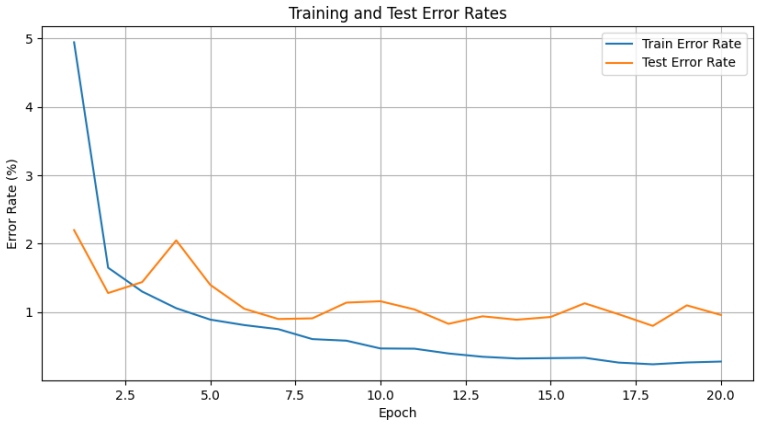


Figure 3: Confusion Matrix

2.1 Modified LeNet-5

To train the LeNet-5 model to handle unseen data it will be essential to include a pre-processing block. That is designed to enhance the input data before feeding it into the neural network, improving the model's robustness and performance.

Secondly, incorporating ReLU and Max pooling. With the cheaper computation of ReLU as a activation function. Also, the dimensional reduction from applying max pooling can reduce overfitting. Additionally increased batch size to 64 to reduce runtime.

From the result training and testing error rate we can tell that max pooling and ReLU provides faster converges. In order to process unseen data the model input processed images resulting a relatively high training and testing error rate in the first epoch.

Overall the modified model have strong performance to process modified images with a overall low testing error rate.

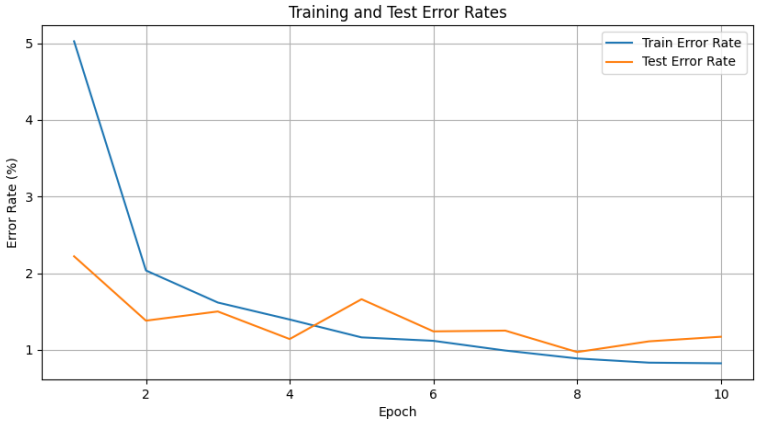


Figure 4: Training and Testing Error Rates

Codes: LeNet-5