Analysis of Neural Network Training on MNIST Dataset with Different Optimizers (Task 4)

2101MC19, 2101MC29. 2101MC41 (Group 4)

September 2, 2024

1 Introduction

This report presents the analysis of a neural network model trained on the MNIST dataset using different optimization algorithms. The purpose of this analysis is to evaluate the performance of various optimizers and their impact on training loss and accuracy. The MNIST dataset, which contains images of handwritten digits, is used as a benchmark dataset for evaluating the model.

2 Dataset Description

The MNIST dataset consists of 60,000 training images and 10,000 test images of handwritten digits, ranging from 0 to 9. Each image is of size 28x28 pixels and is represented in grayscale. The dataset is widely used for training and testing image processing systems and machine learning models.

3 Model Architecture

The neural network model used in this experiment consists of multiple layers:

- Input Layer: Flattened 28x28 pixel image (784 neurons).
- Hidden Layers: Various configurations of hidden layers were used, ranging from 3 to 5 layers with decreasing neuron counts.
- Output Layer: 10 neurons representing the digit classes (0-9).

The ReLU activation function was applied after each hidden layer, and the output layer used the softmax function to produce a probability distribution over the 10 classes.

4 Training Procedure

The training procedure involved training the neural network model using different optimization algorithms, including SGD, SGD with Momentum, SGD with Nesterov Accelerated Gradient (NAG), RMSProp, and Adam. Each optimizer was tested with two configurations of hidden layers:

- Configuration 1: 3 hidden layers with 128, 64, and 32 neurons.
- Configuration 2: 5 hidden layers with 256, 128, 64, 32, and 16 neurons.

The loss function used for training was Cross-Entropy Loss, and the models were trained for a maximum of 100 epochs with an early stopping criterion based on the improvement in training loss.

5 Results

5.1 Training Loss vs. Epochs

The training loss for each optimizer was plotted over the epochs for both configurations to compare the convergence rates and stability of each optimization algorithm.

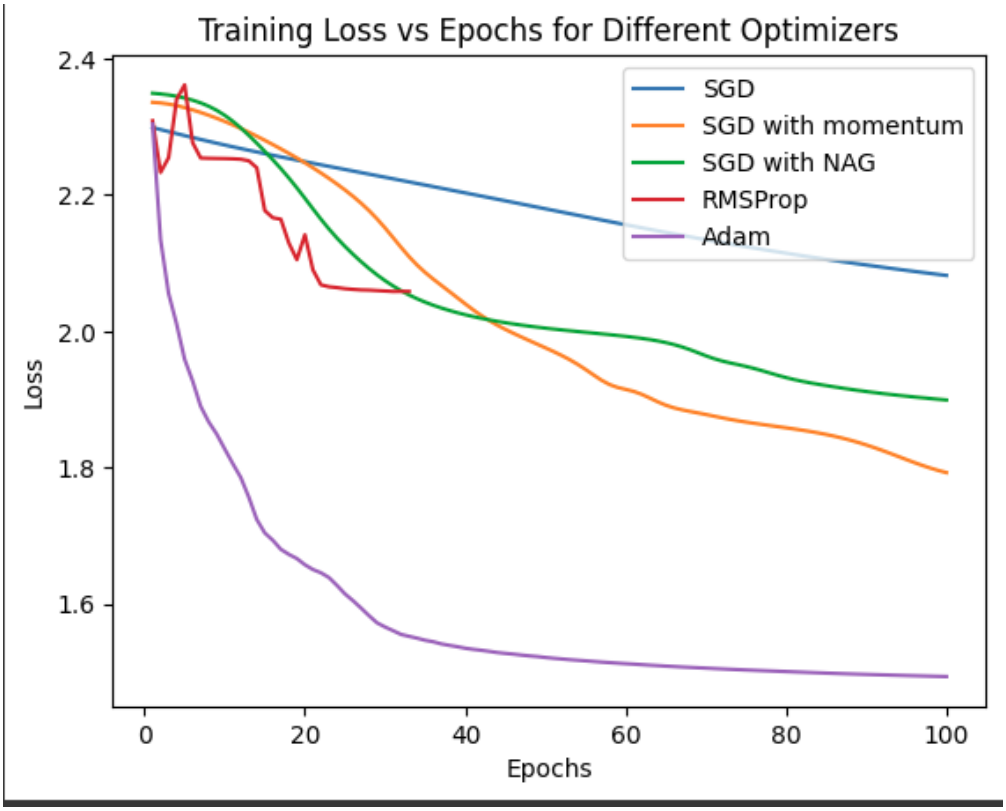

Figure 1: Training Loss vs. Epochs for Configuration 1 - Optimizer 1

Figure 2: Training Loss vs. Epochs for Configuration 2 - Optimizer 1

5.2 Test Accuracy

The test accuracy achieved by the models trained with different optimizers for both configurations is reported in the table below.

Optimizer	**Configuration 1 Accuracy**	**Configuration 2 Accuracy**
SGD	40.48	9.25
SGD with Momentum	69.58	10.52
SGD with NAG	57.32	48.31
RMSProp	40.57	10.09
Adam	95.75	74.25

Table 1: Test Accuracy for Different Optimizers and Configurations

5.3 Confusion Matrix

The confusion matrices for best optimizer(Adam) for each configuration were generated to visualize the number of correct and incorrect predictions made by the model for each digit class.

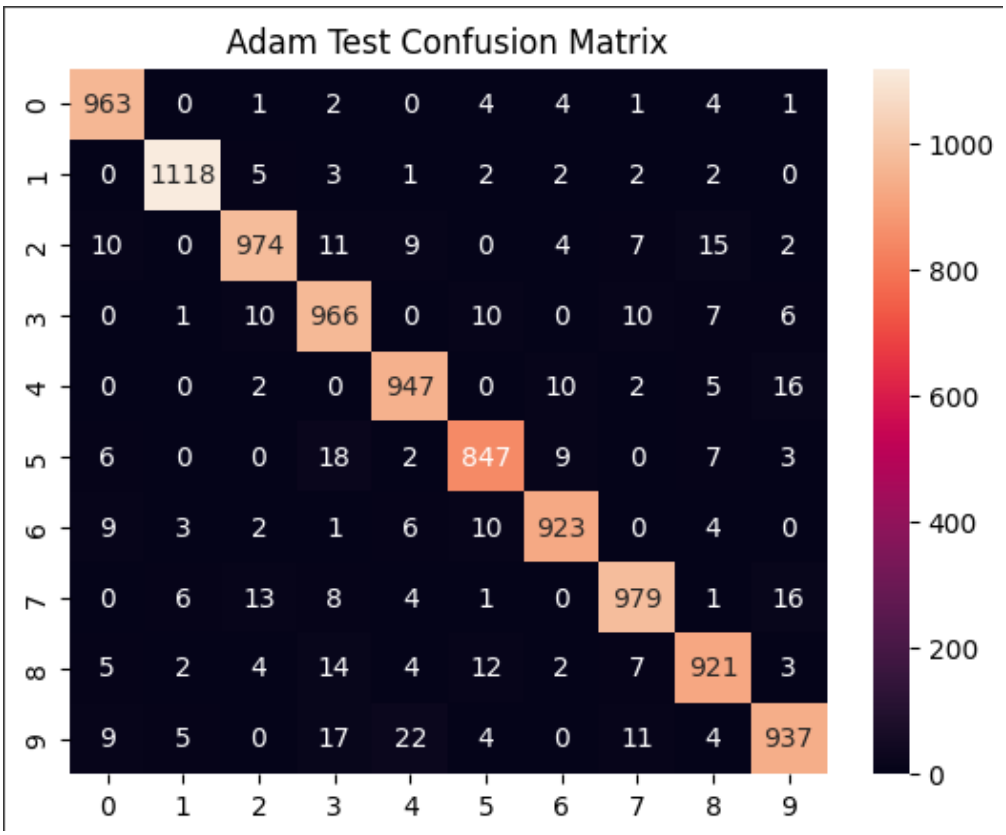

Figure 3: Confusion Matrix for Configuration 1 - Adam

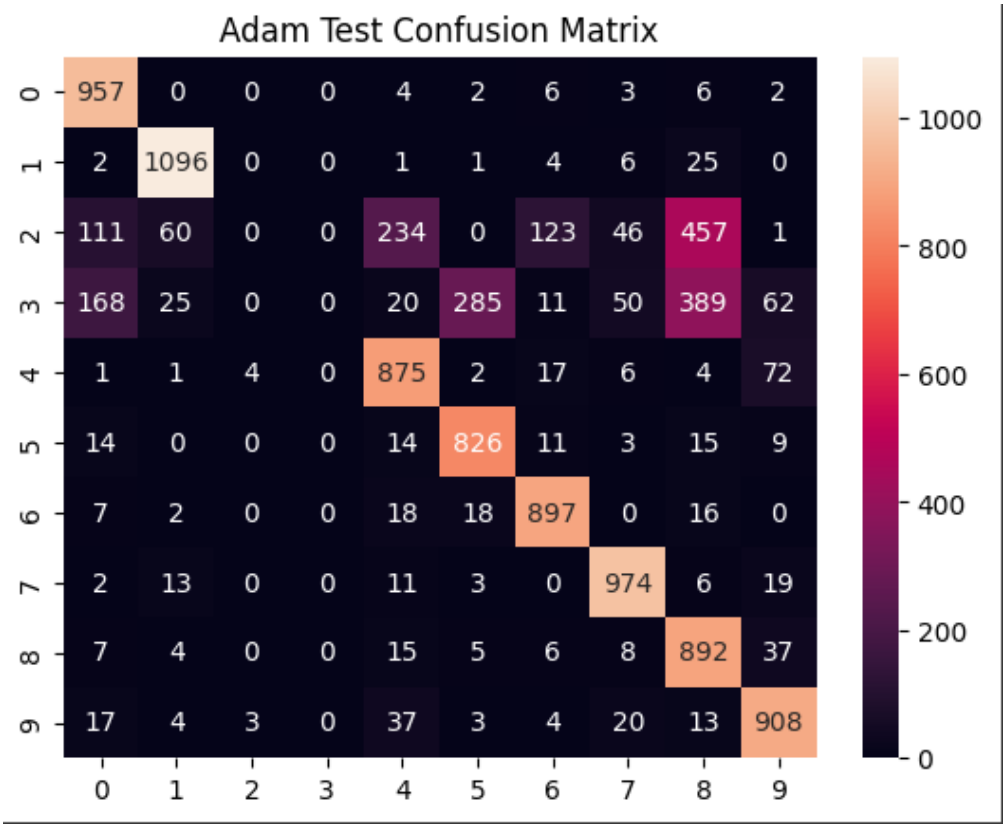

Figure 4: Confusion Matrix for Configuration 2 - Adam

6 Conclusion

In this report, we compared the performance of different optimizers on a neural network trained on the MNIST dataset. The results show that the choice of optimizer significantly impacts the training convergence and the final accuracy of the model. Based on the results, Adam performed the best for both configurations, demonstrating superior accuracy compared to other optimizers. This suggests that Adam is highly effective for the MNIST classification task and may be a preferred choice for similar classification tasks.

7 Future Work

Future work can explore other optimization algorithms, such as Adagrad, Adadelta, and variations of Adam (e.g., AdamW). Additionally, experimenting with different neural network architectures and regularization techniques could provide further insights into improving model performance.