Common Optimizer Comparison

Haochen Ji

***Abstract*—In this report, I applied the Lenet-5 convolutional neural network model to the MNIST dataset with different optimizers, and compared their accuracy, training time, and loss with each other and different hyperparameter. Optimizers covered are SGD, AdaGrad, and RMSProp.**

I. Introduction

IN the current era of machine learning, numerous optimizers are available for different models, particularly in deep learning. When implementing a convolutional neural network (CNN) model, a substantial amount of computation is required due to the abundance of parameters in each layer. Consequently, selecting an appropriate optimizer that facilitates faster convergence to a satisfactory level of accuracy is essential. Furthermore, each optimizer entails specific hyperparameters that need to be carefully calibrated. Hence, the current project aims to evaluate the performance of various optimizers applied to the Lenet-5 CNN model by experimenting with different hyperparameters. The comparison of their performances will be conducted on the MNIST handwritten digit dataset.

II. Related Work

In the paper "Performance comparison of the convolutional neural network optimizer for photosynthetic pigments prediction on plant digital image" by Prilianti, Brotosudarmo, Anam, and Suryanto [1], they compared the performance of seven different optimizers: SGD, AdaGrad, Adadelta, RMSProp, Adam, Adamax, and Nadam when they applied Lenet model on predictions of three main photosynthetic pigments by comparing the Mean Square Error (MSE) for different optimizer under different batch size and after certain numbers of epochs.

III. Data Description

In this project, I use MNIST dataset that contains 60000 training samples and 10000 testing samples, where each sample is 28*28*1 image. All images are handwritten digit number, and the label of each image is the number that is written in image, so all classes are number from 0 to 9. In order to fit Lenet-5 model, I reshape the original image to 32*32*1. As the example sample in Fig. 1, its label is 7.

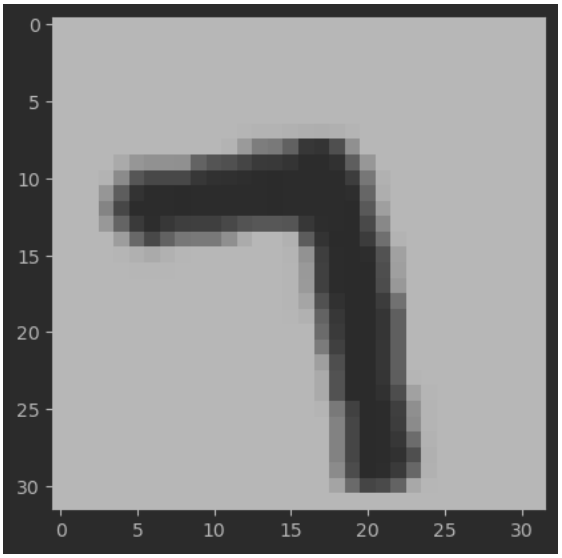

Fig. 1. An example of sample data in dataset. The label of it is 7.

IV. Method Description

In this project, I compared three different optimizers:

A. Stochastic gradient descent (SGD) Optimizer

SGD is an iterative algorithm for optimizing an objective function with smoothness properties, using a stochastic approximation of gradient descent optimization. Instead of calculating the gradient from the entire data set, SGD estimates the gradient from a randomly selected subset, reducing the computational burden for high-dimensional problems at the cost of a lower convergence rate.

B. Adaptive Gradient (AdaGrad) Optimizer

AdaGrad is a variant of SGD algorithm, and it adjusts the learning rate of each feature based on the estimated problem geometry, assigning higher learning rates to infrequent features. This prioritizes the relevance of features over their frequency, leading to parameter updates that are less dependent on feature frequency.

C. Root Mean Square Propagation (RMSProp) Optimizer

RMSProp is also a variant of SGD that adapts the learning rate for each parameter based on the root mean squared (RMS) of the gradients.

The algorithm maintains a moving average of the squared gradients, which is used to normalize the gradient values during training. This normalization helps to improve the convergence of the optimizer and to prevent it from getting stuck in steep or

narrow valleys.

V. Model Description

In this project, I used Lenet-5 CNN model and detail of each layer can be seen in Table 1.

TABLE I
LAYERS IN LENET-5

Hidden Layer	Description
Conv 1	Convolution Layer with 6 filters, kernel size 5*5, and ReLU activation function
Pooling 1	Max Pooling Layer with kernel size 2*2 and stride of 2
Conv 2	Convolution Layer with 16 filters, kernel size 5*5, and ReLU activation function
Pooling 2	Max Pooling Layer with kernel size 2*2 and stride of 2
Conv 3	Convolution Layer with 120 filters, kernel size 5*5, and ReLU activation function
Flatten	Flatten Layer convert 2D matrix to 1D array
FC 1	Fully Connection Layer with input size 120 and output size 84 performs a linear transformation.
FC 2 / Output Layer	Fully Connection Layer with input size 84 and output size 10 performs a linear transformation.

VI. Experimental Procedure and Results

For each optimizer, I tried different parameters:

A. SGD Optimizer

As shown in Table II, I used 5 different set of hyper-parameters for SGD optimizer, where "lr" represents learning rate, and "weight_decay" is parameter for L2 regularization.

TABLE II
HYPER-PARAMETER SETS FOR SGD OPTIMIZER

Set	Hyper-parameters
SGD0	lr = 0.1, momentum = 0.9, weight_decay = 5e-4
SGD1	lr = 0.05, momentum = 0.9, weight_decay = 5e-4
SGD2	lr = 0.01, momentum = 0.9, weight_decay = 5e-4
SGD3	lr = 0.005, momentum = 0.9, weight_decay = 5e-4
SGD4	lr = 0.001, momentum = 0.9, weight_decay = 5e-4

As in Fig. 2.1 and Fig. 2.2, SGD2 hyper-parameter set has better performance comparing to other 4 sets.

Fig. 2.1. Accuracy comparison between different hyper-parameter sets for SGD Optimizer

Fig. 2.2. Loss comparison between different hyper-parameter sets for SGD Optimizer

B. AdaGrad Optimizer

As shown in Table III, I used 5 different set of hyper-parameters for AdaGrad optimizer, where "lr" represents learning rate, and "weight_decay" is parameter for L2 regularization.

TABLE III
HYPER-PARAMETER SETS FOR ADAGRAD OPTIMIZER

Set	Hyper-parameters
AdaGrad0	lr = 0.1, weight_decay = 5e-4
AdaGrad1	lr = 0.05, weight_decay = 5e-4
AdaGrad2	lr = 0.01, weight_decay = 5e-4
AdaGrad3	lr = 0.005, weight_decay = 5e-4
AdaGrad4	lr = 0.001, weight_decay = 5e-4

As in Fig. 3.1 and Fig. 3.2, AdaGrad2 hyper-parameter set has better performance comparing to other 4 sets.

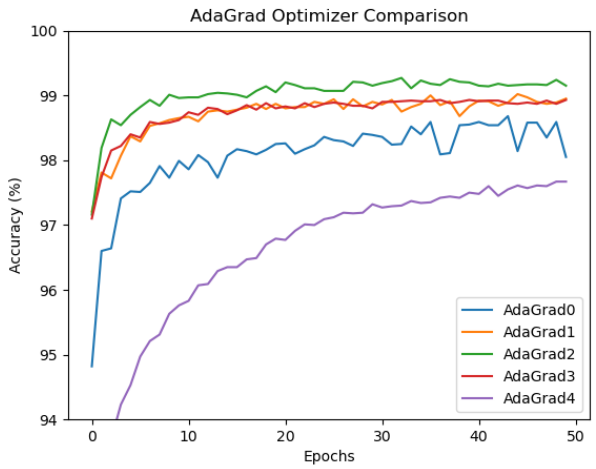

Fig. 3.1. Accuracy comparison between different hyper-parameter sets for AdaGrad Optimizer

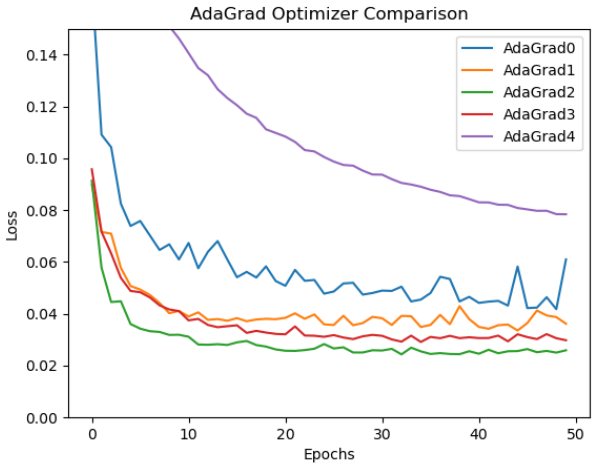

Fig. 3.2. Loss comparison between different hyper-parameter sets for AdaGrad Optimizer

C. Root Mean Square Propagation (RMSProp) Optimizer

As shown in Table IV, I used 9 different set of hyper-parameters for RMSProp optimizer where "lr" represents learning rate, and "weight_decay" is parameter for L2 regularization, and I split into three different groups by their alpha value.

TABLE IV
HYPER-PARAMETER SETS FOR RMSPROP OPTIMIZER

Set	Hyper-parameters
RMSProp0	lr = 0.001, alpha = 0.9, momentum = 0.1
RMSProp1	lr = 0.0005, alpha = 0.9, momentum = 0.1
RMSProp2	lr = 0.0001, alpha = 0.9, momentum = 0.1
RMSProp3	lr = 0.001, alpha = 0.95, momentum = 0.1
RMSProp4	lr = 0.0005, alpha = 0.95, momentum = 0.1
RMSProp5	lr = 0.0001, alpha = 0.95, momentum = 0.1
RMSProp6	lr = 0.001, alpha = 0.99, momentum = 0.1
RMSProp7	lr = 0.0005, alpha = 0.99, momentum = 0.1
RMSProp8	lr = 0.0001, alpha = 0.99, momentum = 0.1

As in Fig. 4.1.1, Fig. 4.1.2, Fig. 4.2.1, Fig. 4.2.2, Fig. 4.3.1, and Fig. 4.3.2, RMSProp5 hyper-parameter set has better performance comparing to all other 8 sets.

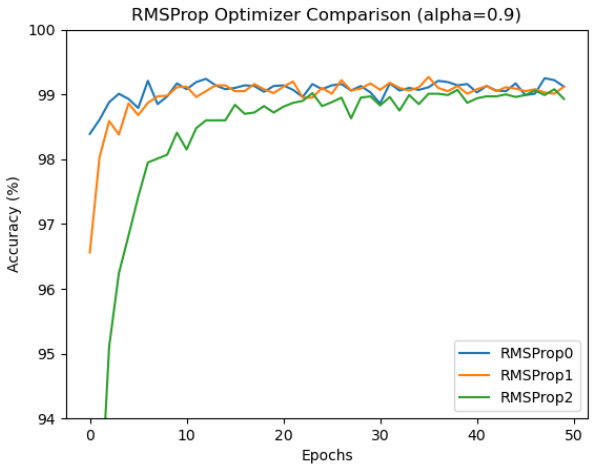

Fig. 4.1.1. Accuracy comparison between different hyper-parameter sets for RMSProp Optimizer when alpha=0.9

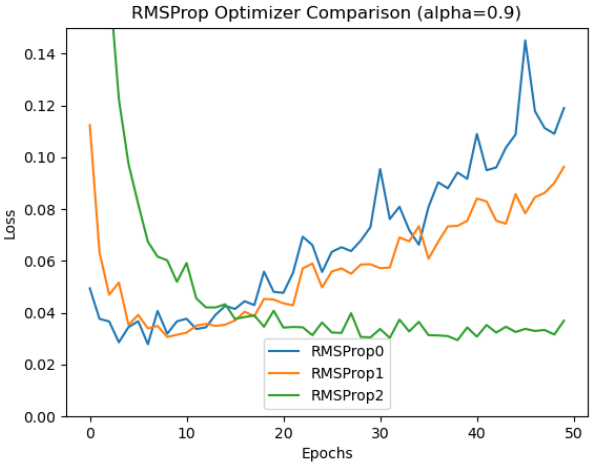

Fig. 4.1.2. Loss comparison between different hyper-parameter sets for RMSProp Optimizer when alpha=0.9

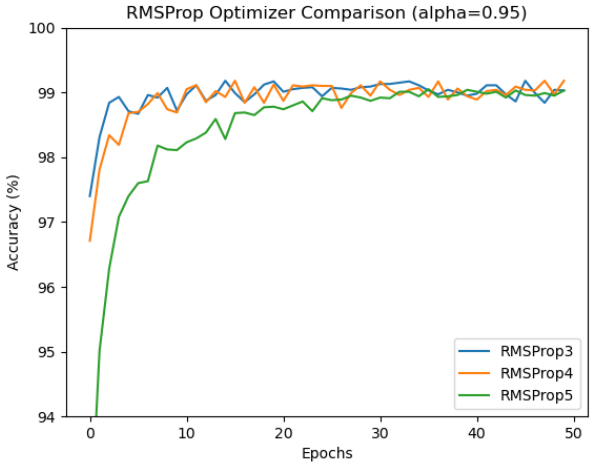

Fig. 4.2.1. Accuracy comparison between different hyper-parameter sets for RMSProp Optimizer when alpha=0.95

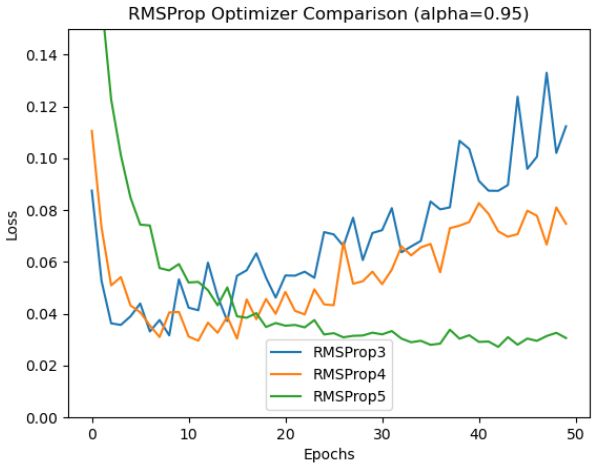

Fig. 4.2.2. Loss comparison between different hyper-parameter sets for RMSProp Optimizer when alpha=0.95

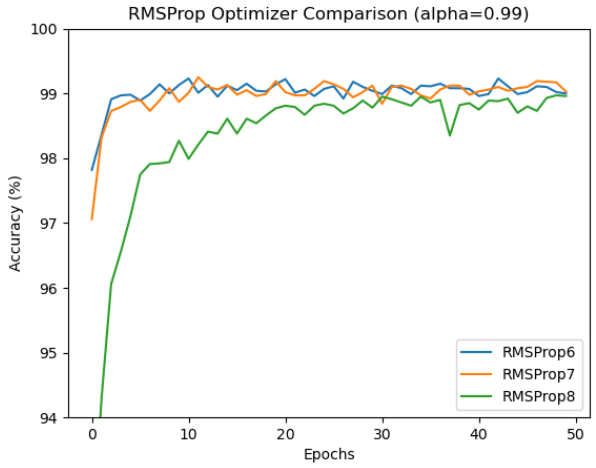

Fig. 4.3.1. Accuracy comparison between different hyper-parameter sets for RMSProp Optimizer when alpha=0.99

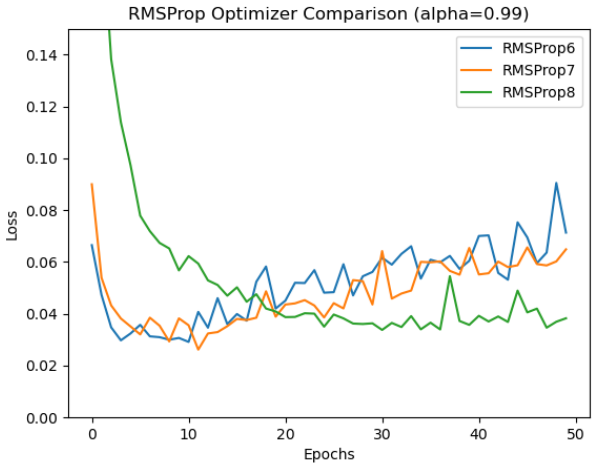

Fig. 4.3.2. Loss comparison between different hyper-parameter sets for RMSProp Optimizer when alpha=0.99

VII. Conclusion

By comparing three optimizers with their best hyper-parameter set in Fig. 5.1 and Fig. 5.2, we can see that AdaGrad Optimizer with lr = 0.01 and weight_decay = 5e-4 has best output, and from Fig. 5.3 we can see the time for all three optimizers are close.

Fig. 5.1. Accuracy comparison between three optimizers

Fig. 5.2. Loss comparison between three optimizers

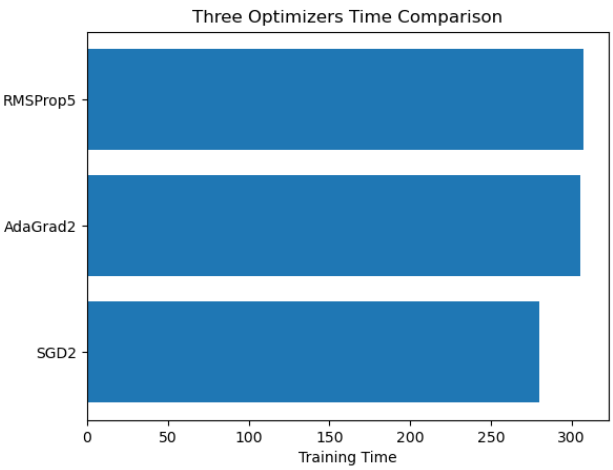

Fig. 5.3. Training time comparison between three optimizers

REFERENCES

[1] G. O. Young, “Synthetic structure of industrial plastics (Book style with paper title and editor),” in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.