# Comparing Different Optimizers When Training the LeNet-5 Model on MNIST Data

Zeeshan Qureshi
Zq40@scarletmail.rutgers.edu

## Abstract

In this study, four different optimization algorithms are tested to determine which one does the best on training handwritten digits from the MNIST dataset on the LeNet-5 convolutional neural network model. The learning rate was set to 0.01 for all the optimizers with no other parameters passed in. It is found that in terms of accuracy, the Adagrad optimizer does the best with an accuracy of 99.81%, whereas the RMSProp is the fastest optimizer with a speed of 157.83 seconds.

## Introduction

As deep learning has grown in popularity in recent times, interest in deep learning optimizers has increased as well. In the corporate world, choosing the correct optimizer can be the difference between generating large sums of revenue and wasting funds when it could have been avoided. In the research world, choosing the correct optimizer can determine the feasibility of a research project and can be the difference between having the best paper or not in an academic conference.

In the context of machine learning, optimizers are used to update the weights of a neural network. Different optimizers were developed over time to address problems encountered by using previously developed optimizers. In this study, four optimizers are implemented to update the weights of the LeNet-5 model using MNIST data to train and test the network. The first two optimizers studied are Stochastic Gradient Descent and Mini Batch Stochastic Gradient Descent. The second two optimizers studied are Adagrad and RMSProp. The purpose of this paper is to conduct a comparative study to determine which of the four optimizers perform the best in accuracy and execution time.

## Related Work

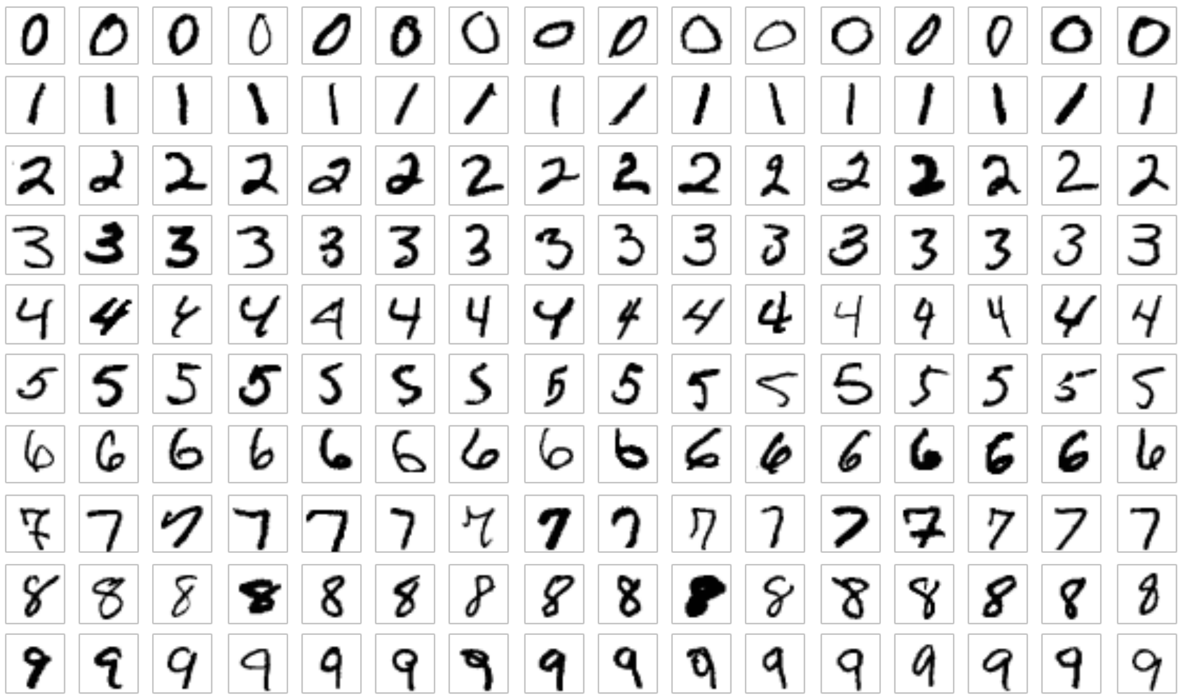
Research in comparing optimizers on a model for a problem has been done before. For instance, Mustika et al. conducted a study where they compared the AdaDelta, AdaGrad, Adam, Adamax, RMSprop, and Stochastic Gradient Descent optimizers on a CNN model and an LSTM model trained over the Stanford Earthquake Dataset (STEAD) [1]. After conducting their study, they concluded that the RMSProp optimizer performed the best over both the models, with a model 1 accuracy of 99.2% and a loss of 0.102, and a model 2 accuracy of 99% and a loss of 0.107.

In another study, Zohrevand et al. studied the efficacy of four optimizers, namely SGD, Adam, Adadelta, and Adagrad, on two convolutional neural networks, namely VGG11 and AlexNet [2]. These models are trained on the CIFAR-10, MNIST, and Fashion MNIST datasets. This study concluded that the Adam and Adagrad optimizers have the best performance compared to the other two in terms of training cost and recognition accuracy.

A third study conducted by Pomerat et al. involved testing various optimizers on a neural network architecture for regression on randomly generated polynomial datasets [3]. The study concluded that SGD is more accurate than Adam and RMSprop for "regression tasks on low-featured polynomial data. Furthermore, the study concluded that SGD is more efficient with fewer epochs for finding local minimums than the other two optimizers. However, in the long term, the performance differences vanished.

## Data Description

The data used for training and testing the LeNet-5 model is the famous MNIST dataset. It is a set of 70,000 handwritten digits, all of which are square 28x28 pixel grayscale images. 60,000 of these images are for training and 10,000 are for testing [4]. Below are examples of some of these images [5]:

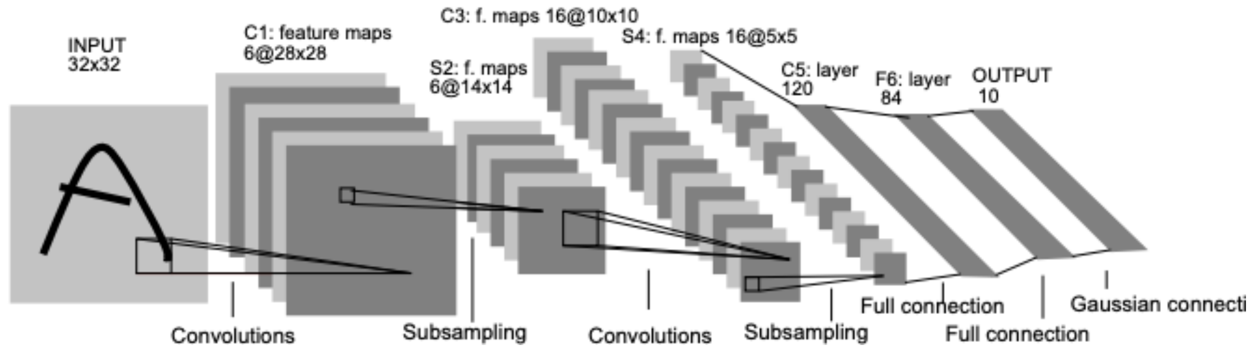


## Method Description

PyTorch was utilized to construct the LeNet-5 model, and for training and testing the model. The four optimizers mentioned previously were applied to this singular architecture, with a standard learning rate set to 0.01. The number of epochs was set to five and the batch size was set to 128 (except for plain SGD, which has a batch size of 1). To ensure a fair comparison, momentum was set to 0 for all optimizers, and weight decay was set to 0 as well for the optimizers that have that parameter. Epsilon was set to 0 as well for Adagrad and RMSprop.

## Model Description

The model used to conduct this study is an adaption of the well-known LeNet-5, introduced by Yann LeCun et al. in a paper from 1998 [6]. The model utilized in this study consists of three convolutional layers, which are all followed by a ReLU layer. A max pooling layer follows the first two convolutional layers. The max pool layers have a kernel size of 2 and a stride of 2. The first convolutional layer connects the input image to six feature maps with 5x5 convolutions. The second convolutional layer connects the 6 input channels to 16 output channels with 5x5 convolutions [7]. The third convolutional layer connects 16 input channels to 120 output channels with 5x5 convolutions. Once the convolutional operations, as well as the ReLU and max pool operations, are conducted, the data is passed to a fully connected layer connecting 16 feature maps to 120 output units. This is then followed by a ReLU layer. The result is passed to another fully connected layer which connects 84 input units to 10 output units. This result is then passed to a softmax function. Note: the softmax function is not explicitly defined in the code because we use the PyTorch implementation of Cross Entropy Loss which defines the softmax layer implicitly. Below is a visual representation of the original LeNet-5 model [6]:



## Experimental Procedure and Results

As mentioned in the Method Description section, four optimizers were utilized to update parameters in this model. To reiterate, the learning rate was set as 0.01, and no other parameters were passed to the optimizers. The batch size was set as 128 (except for plain SGD which has a batch size of 1). The code for loading the MNIST data was taken from https://blog.paperspace.com/writing-lenet5-from-scratch-in-python/ .

The results show that the Adagrad optimizer did the best with an accuracy of 99.81% (9881/10000). The plain SGD had the second-best accuracy with an accuracy of 99.55% (9855/10000). The RMSProp was third in accuracy with 97.39% (9739/10000) and mini batch SGD was last in accuracy with 96.88% (9688/10000). In terms of execution time, RMSProp performed the best with a time of 157.83 seconds. Second fastest was Mini-Batch Gradient Descent with an execution time of 157.90 seconds. Third fastest was Adagrad with a time of 159.77 seconds. Last was plain SGD with an execution time of 738.49 seconds. The results are summarized below.

| Optimizer | Accuracy | Execution Time |
|---|---|---|
| SGD | 99.55% | 738.49 |
| Mini-Batch Gradient Descent | 96.88% | 157.90 |
| RMSProp | 97.39% | 157.83 |
| Adagrad | 99.81% | 159.77 |

## Conclusion

In this study, four different optimizers were used for the training of the LeNet-5 network using MNIST data. Stochastic Gradient Descent, Mini-Batch Stochastic Gradient Descent, RMSProp, and Adagrad were implemented with the same learning rate, and with no other parameters passed into the optimizers. The results showed that Adagrad had the best accuracy and RMSProp had the best execution time. Further studies can focus on comparing optimizers while passing other parameters besides the learning rate, such as momentum, weight decay, alpha, and epsilon for Adagrad and RMSProp.

## References

[1] I. W. Mustika, H. N. Adi and F. Najib, "Comparison of Keras Optimizers for Earthquake Signal Classification Based on Deep Neural Networks," *2021 4th International Conference on Information and Communications Technology (ICOIACT)*, 2021, pp. 304-308, doi: 10.1109/ICOIACT53268.2021.9563990.

[2] A. Zohrevand and Z. Imani, "An Empirical Study of the Performance of Different Optimizers in the Deep Neural Networks," *2022 International Conference on Machine Vision and Image Processing (MVIP)*, 2022, pp. 1-5, doi: 10.1109/MVIP53647.2022.9738743

[3] Pomerat, John , Segev, Aviv, Datta, Rittuparna, "On Neural Network Activation Functions and Optimizers in Relation to Polynomial Regression", IEEE International Conference 2019

[4] https://selectstar-ai.medium.com/what-is-mnist-and-why-is-it-important-e9a269edbad5

[5] Wikipedia contributors. (2022, April 17). MNIST database. In *Wikipedia, The Free Encyclopedia*. Retrieved 02:44, May 13, 2022, from https://en.wikipedia.org/w/index.php?title=MNIST_database&oldid=1083252903

[6] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791

[7] https://github.com/lychengr3x/LeNet-5-Implementation-Using-Pytorch/blob/master/LeNet-5%20Implementation%20Using%20Pytorch.ipynb