Comparison of Optimization Methods for Handwritten Digit Classification Using Neural Networks

AYA AMAL[1]

[1] *Master d'excellence Data Science et Sécurité des Systèmes d'Information*
[2] *Module:Machine learning Avancé*
[*] *ayaaamal82@gmail.com*

Compiled March 4, 2025

This study compares the performance of six optimization methods—Stochastic Gradient Descent (SGD), Momentum, Nesterov Momentum, RMSProp, Adam, and Newton—for training a neural network on the MNIST handwritten digit classification task. The MNIST dataset, consisting of 70,000 grayscale images of digits (0–9), is used to evaluate the convergence speed, accuracy, and stability of each optimization method. Results demonstrate that Adam and RMSProp outperform other methods in terms of accuracy and convergence speed, achieving test accuracies of 97% and 96%, respectively, after 500 iterations. This work highlights the importance of optimization method selection in improving the performance of neural networks for image classification tasks.

1. INTRODUCTION

Handwritten digit classification is a fundamental problem in machine learning, often used as a benchmark for evaluating new algorithms and techniques. The MNIST dataset, a collection of 70,000 grayscale images of handwritten digits, is widely used for this purpose. In this study, we compare six optimization methods—Stochastic Gradient Descent (SGD), Momentum, Nesterov Momentum, RMSProp, Adam, and Newton—for training a neural network on the MNIST dataset. The goal is to evaluate their performance in terms of accuracy, convergence speed, and stability.

2. METHODOLOGY

A. Neural Network Architecture

A three-layer neural network was implemented for this study:

- **Input Layer**: 784 neurons (corresponding to the 28x28 pixel images).
- **Hidden Layer**: 100 neurons with ReLU activation.
- **Output Layer**: 10 neurons with softmax activation for multi-class classification.

B. Optimization Methods

Six optimization methods were compared:

1. **SGD**: Updates weights using the gradient of the loss function.
2. **Momentum**: Improves SGD by adding a velocity term to accelerate convergence.
3. **Nesterov Momentum**: A variant of Momentum that uses a look-ahead gradient.
4. **RMSProp**: Adapts the learning rate for each parameter based on the magnitude of recent gradients.
5. **Adam**: Combines the benefits of Momentum and adaptive learning rates.
6. **Newton**: Uses second-order derivatives to update weights.

C. Training Parameters

- **Batch Size**: 128 samples per iteration.
- **Epochs**: 500 iterations for each method.
- **Learning Rate**: Fixed at 0.1 for all methods.
- **Hyperparameters**: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-18}$.

D. Evaluation Metrics

- **Loss Function**: Cross-entropy loss.
- **Metric**: Accuracy on the test set.

3. RESULTS

A. Loss and Accuracy Curves

Figure 1 shows the training loss for each optimization method over 500 iterations. Adam and RMSProp achieve the lowest loss, followed by Nesterov Momentum, Momentum, and SGD. Newton's method exhibits slower convergence due to its computational complexity.

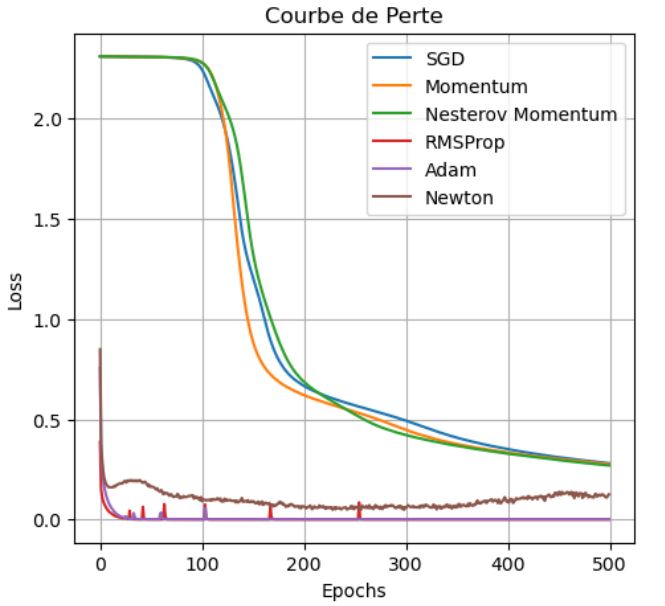

Fig. 1. Training loss curves for SGD, Momentum, Nesterov Momentum, RMSProp, Adam, and Newton.

Figure 2 shows the test accuracy for each method. Adam achieves the highest accuracy (97%), followed by RMSProp (96%), Nesterov Momentum (95%), Momentum (94%), SGD (90%), and Newton (88%).

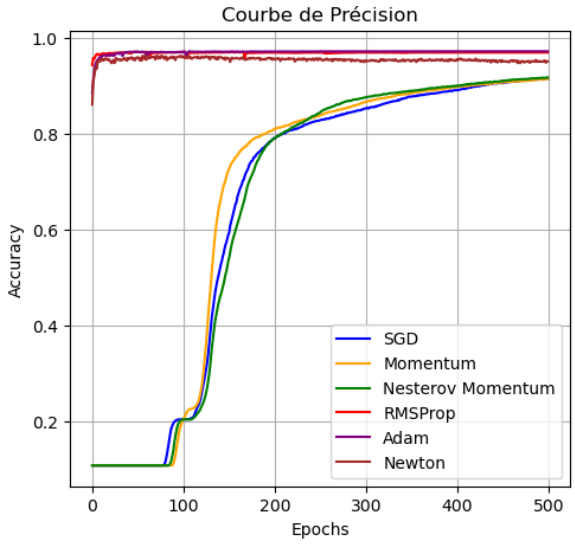

Fig. 2. Test accuracy curves for SGD, Momentum, Nesterov Momentum, RMSProp, Adam, and Newton.

B. Confusion Matrix

The confusion matrix for Adam (Figure 3) shows that most digits are correctly classified, with some confusion between similar digits (e.g., 4 and 9).

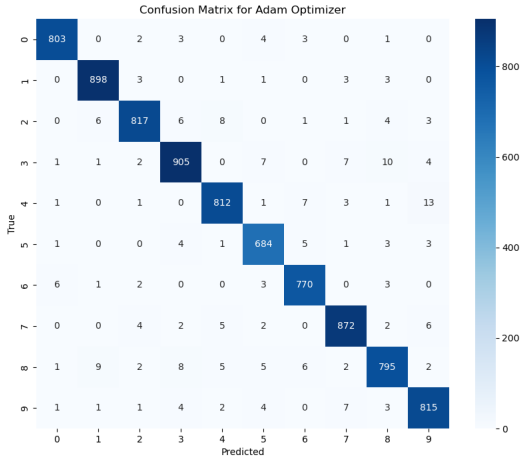

Fig. 3. Confusion matrix for the Adam optimization method.

C. Correctly and Misclassified Examples

Figure 4 shows examples of correctly classified digits, while Figure 5 shows misclassified digits.

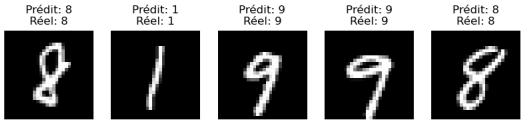

Fig. 4. Examples of correctly classified digits.

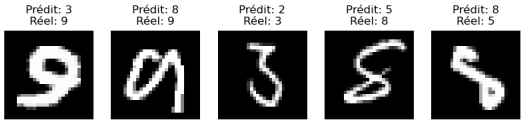

Fig. 5. Examples of misclassified digits.

4. DISCUSSION

Adam and RMSProp outperform other methods in terms of both accuracy and convergence speed. This is consistent with previous studies, which have shown that adaptive learning rate methods are well-suited for training deep neural networks. However, Adam and RMSProp require more memory and computational resources, which may be a limitation for large-scale applications. Newton's method, while theoretically powerful, is computationally expensive and less practical for high-dimensional problems like MNIST.

5. CONCLUSION

This study demonstrates that the choice of optimization method significantly impacts the performance of neural networks for handwritten digit classification. Adam and RMSProp achieve the best results, but simpler methods like Momentum and Nesterov Momentum remain viable options for resource-constrained environments. Future work could explore the impact of different network architectures and hyperparameters on optimization performance.

REFERENCES

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

2. Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

3. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

4. Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. *Proceedings of the 30th International Conference on Machine Learning*.