Understanding the Adam Optimization Algorithm in Machine Learning

Oles Hospodarskyy[1,*,†], Vasyl Martsenyuk[2,†], Nataliia Kukharska[3,†], Andriy Hospodarskyy[4,†], Sofiia Sverstiuk[5,†]

[1] *Lviv Polytechnic National Universiy, Bandery St. 12, Lviv, 79001, Ukraine*
[2] *University of Bielsko-Biala, Willowa St. 2, Bielsko-Biala, 43-300, Poland*
[3] *Ternopil National Ivan Puluj Technical University, Rus'ka St. 56, Ternopil, 46001, Ukraine*
[4] *I. Horbachevsky Ternopil National Medical University, Maidan Voli, 1, Ternopil, 46002, Ukraine*
[5] *Ternopil National Pedagogical University, 2 Maxyma Kryvonosa St., Ternopil, 46027, Ukraine*

Abstract
Machine learning and artificial intelligence are significant areas of interest in both contemporary science and society. There are various optimization algorithms used. The algorithm's speed depends on the size of the dataset, the number of model parameters, and the number of iterations. Standard gradient descent requires computing the gradient of the cost function over the entire dataset, which can be resource-intensive, especially with large datasets. In Adam, a separate learning rate is maintained for each parameter weight, which is adapted and updated individually. The algorithm selects a smaller learning rate for frequently updated parameters and a larger one for parameters corresponding to rare features. To measure the effectiveness and universality of the Adam, we compared it with other optimization algorithms. Analysis of the experiment results conducted on various datasets, indicates a significant advantage of the Adam optimization algorithm. To make sure our model works well for our specific needs, we made a small dataset ourselves. The famous MNIST dataset, created by American researchers, might not match our handwritten numbers perfectly. The results appear promising, with the model achieving an accuracy of 97%, meaning it correctly predicted 97 out of 100 images. This level of accuracy suggests that the model is performing well on our custom dataset, demonstrating its effectiveness in recognizing and classifying our handwritten numbers. Experiments on various datasets showed that the Adam algorithm is capable of achieving good results across a wide range of machine learning tasks.

Keywords
Adam algorithm, machine learning, artificial intelligence, loss function, gradient descent

1. Introduction

Machine learning and artificial intelligence are significant areas of interest in both contemporary science and society [1]. They represent some of the most advanced

CITI'2024: 2nd International Workshop on Computer Information Technologies in Industry 4.0, June 12-14, 2024, Ternopil, Ukraine. ∗ Corresponding author. † These authors contributed equally.

oles.hospodarskyi.kb.2021@lpnu.ua (O. Hospodarskyy); vmartsenyuk@ath.bielsko.pl (V. Martsenyuk), nataliia.p.kukharska@lpnu.ua (N. Kukharska); hospodarskyy@tdmu.edu.ua (A. Hospodarskyy); khrystynasofia@gmail.com (S. Sverstiuk)
0009-0005-9088-3015 (O. Hospodarskyy); 0000-0001-5622-1038 (V. Martsenyuk); 0000-0002-0896-8361 (N. Kukharska); 0000-0002-9394-2675 (A. Hospodarskyy); 0000-0001-5595-4918 (S. Sverstiuk)

technologies applicable across various industries, including healthcare, finance, transportation and entertainment. The theoretical foundations of machine learning have been explored and expanded upon by some of the greatest figures in the field. Geoffrey Hinton, often revered as the "Godfather of Deep Learning," has laid the groundwork for many modern machine learning techniques with his groundbreaking research on neural networks

Yann LeCun's work on convolutional neural networks (CNNs) has revolutionized computer vision and pattern recognition, while Yoshua Bengio's contributions to neural network models have greatly advanced natural language processing and unsupervised learning. Ian Goodfellow's work on generative adversarial networks (GANs) has opened up new avenues in unsupervised learning and generative modeling, while Juergen Schmidhuber's contributions to recurrent neural networks (RNNs) and long short-term memory (LSTM) networks have propelled advancements in sequential learning and AI [2].

Each year the number of scientific publications dedicated to algorithms and machine learning methods continues to increase. However, despite significant progress made by researchers in this field, a range of unresolved and insufficiently studied issues persists. These include challenges related to optimization, which entail the search for optimal model parameters to achieve maximum prediction accuracy.

The main objective of this article is to investigate the Adam optimization algorithm, compare its effectiveness with other well-known algorithms on standard datasets such as MNIST and FashionMNIST, and assess its accuracy in recognizing and classifying handwritten characters [3].

2. General principles of optimization to find the best algorithm in machine learning

When an input signal is received by the machine model, it undergoes processing through a function, and after a series of computations, it transforms into an output value. Then the model compares the generated output with the actual output value and computes the loss function. The loss function is a measure of how well the model performs on a given task. For example, a popular loss function MSE computes squared difference between values:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2, \qquad (1)$$

Now we need some algorithm that will adjust the parameters of a model (w1, w2,...wn) to minimize (or maximize) the loss function. That’s the basic concept of optimization in machine learning.

There are various optimization algorithms used for this task. These algorithms are iterative, meaning they update the model parameters during each epoch during the training process.

3. The description of the gradient descent algorithm as a fundamental optimization technique

The idea of gradient descent is to update the parameters of the model (weights) by moving in the direction of the steepest descent of the loss function [5].

At first algorithm initializes random weights (or zeros). Then it calculates the loss function of the whole dataset, for example, MSE.

Then gradient descent calculates the derivative of the loss function concerning each model parameter (weight) to determine the direction of the update.

After computing the gradient of the loss function concerning each weight, the algorithm updates the weights by subtracting a fraction of the gradient from the current value of each weight. This fraction is known as the learning rate, denoted by α, and it controls the size of the step taken in the direction of the negative gradient.

$$w = w - \alpha\left[\frac{\partial Loss}{\partial w}\right], \tag{2}$$

This process is repeated iteratively, and with each iteration, the algorithm progressively approaches a local minimum of the loss function, where the weight values are optimal (Figure 1).

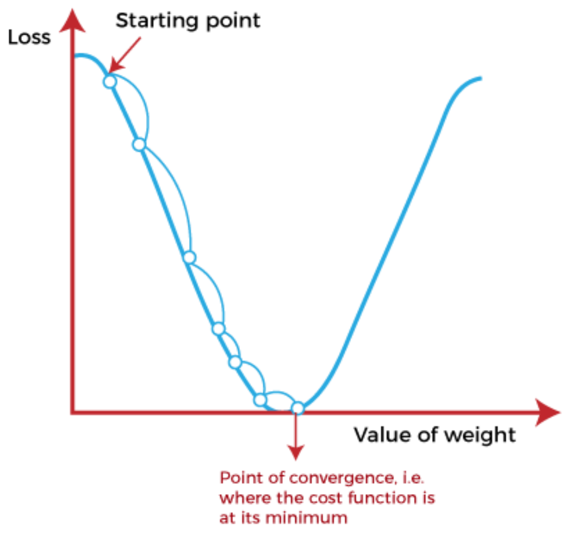

Loss
Starting point
Value of weight
Point of convergence, i.e. where the cost function is at its minimum

Figure 1: The relationship between the loss function and the weight value w
Source: photographed by the author

The algorithm's speed depends on the size of the dataset, the number of model parameters, and the number of iterations. Typically, larger datasets and more complex models require more time and resources for training. Additionally, the speed of the algorithm can be influenced by the choice of learning rate. Selecting a too large learning rate may cause the algorithm to move too quickly and fail to find the minimum of the weight function (Figure 2), while choosing a too small learning rate may prolong the training process.

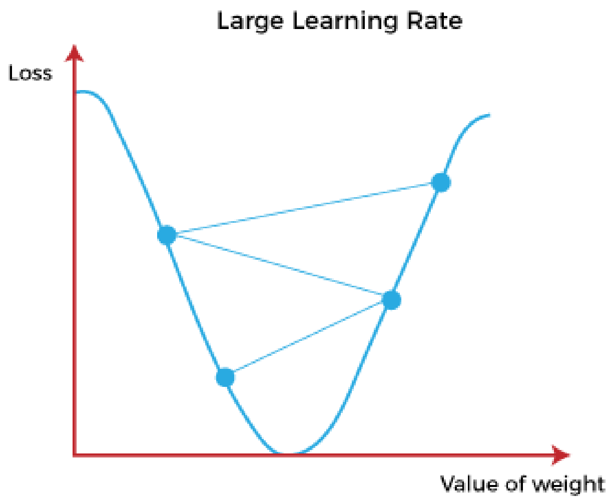


Figure 2: Demonstrative example where a learning rate that is too large.
Source: photographed by the author

4. Stochastic gradient descent, SGD

Standard gradient descent requires computing the gradient of the cost function over the entire dataset, which can be resource-intensive, especially with large datasets. Therefore, in such cases, stochastic gradient descent (SGD) is applied, which is more efficient for optimizing models with a large amount of data [4].

Standard gradient descent updates the model weights at each iteration using the entire dataset to compute the gradient of the loss function. However, stochastic gradient descent computes the gradient and updates the weights for each data sample in the dataset separately. That is, on each iteration, SGD uses only one data sample instead of the entire dataset, allowing for quick weight updates and more efficient processing of large datasets [5].

However, SGD can be sensitive to the initial values of the weights, causing it to get stuck in a local minimum and fail to find the global minimum of the loss function (Figure 3). Data normalization could help mitigate this issue for linear models, however, in more complex models like neural networks, normalization may not be sufficient.

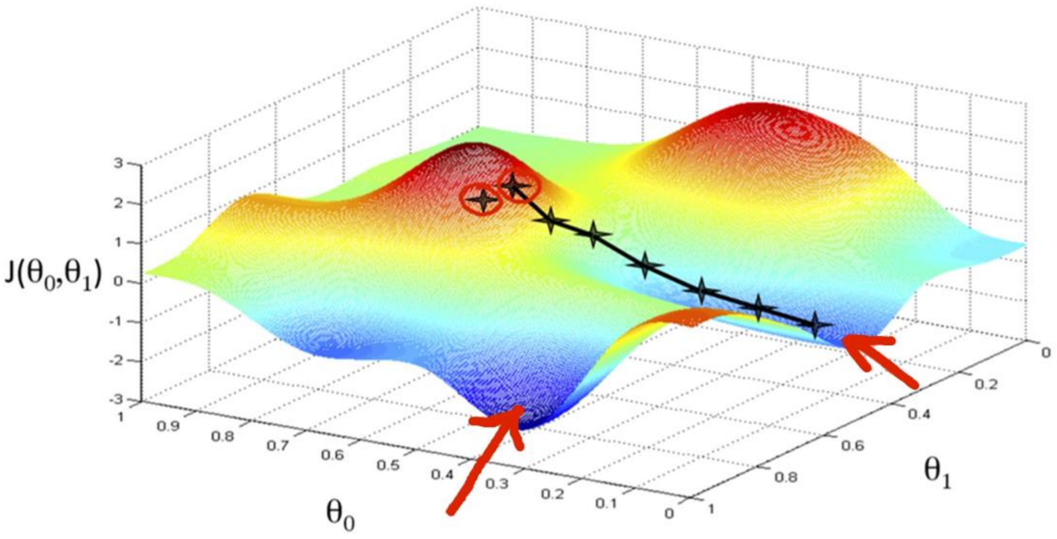


Figure 3: Graphical example where the algorithm failed to find the global minimum of the function
Source: photographed by the author

That's where Adam comes in handy. It uses the history of previous gradients to adaptively adjust the learning rates for each parameter, helping to overcome the limitations of SGD.

5. Adam

Adam was presented by Diederik Kingma from OpenAI and Jimmy Ba from the University of Toronto in their 2015 ICLR paper (poster) titled "Adam: A Method for Stochastic Optimization". The name Adam is derived from adaptive moment estimation.

Adam differs from classical stochastic gradient descent. Standard stochastic gradient descent uses a single learning rate (alpha) for updating weights, and this learning rate remains constant throughout training [6].

In Adam, a separate learning rate is maintained for each parameter weight, which is adapted and updated individually. The algorithm selects a smaller learning rate for frequently updated parameters and a larger one for parameters corresponding to rare features.

The authors describe Adam as combining the advantages of two other extensions of stochastic gradient descent. Specifically:

- The Adaptive Gradient Algorithm (AdaGrad) which is particularly effective with sparse gradients, such as in natural language processing tasks (NLP) and computer vision. It employs a method that maintains an individual learning rate for each parameter, facilitating efficient updates for rarely used parameters. However, AdaGrad may encounter the issue of rapidly decreasing learning rates, which can prematurely halt the learning process.

- The Root Mean Square Propagation (RMSProp) algorithm which, unlike AdaGrad, mitigates the problem of decreasing learning rates. It utilizes a method that sustains individually adjusted learning rates for each parameter, adapted based on the recent average of gradient magnitudes for weights. This algorithm performs effectively on online tasks and tasks where parameters may change over time (non-stationary tasks).

The hyperparameters of Adam include:

1. Learning rate: Determines the size of the step by which the model weights will change during each iteration. A large learning rate can lead to unstable model training, while a too small value can slow down the learning process. Typically, an initial learning rate is chosen, but Adam automatically adapts it over time.
2. Beta1 and Beta2: These parameters control the exponential smoothing of previous gradients and their squares, respectively. Beta1 is responsible for smoothing gradients, while Beta2 handles the smoothing of gradient squares. Typically, values such as Beta1 = 0.9 and Beta2 = 0.999 work well, but they can be manually adjusted if needed.
3. Epsilon: A small numerical value added to the denominator in the Adam formula to avoid division by zero.

The Adam algorithm computes the exponential moving average of the gradient (first moment) and the squared gradient (second moment) of the weights, where the parameters beta1 and beta2 control the smoothing rates of these moving averages [7].

In the context of exponential moving average (Figure 4), smoothing occurs by assigning more weight to newer data. Thus, the model responds more to the recent changes in data than to older values, allowing it to adapt more quickly to any new data trends.

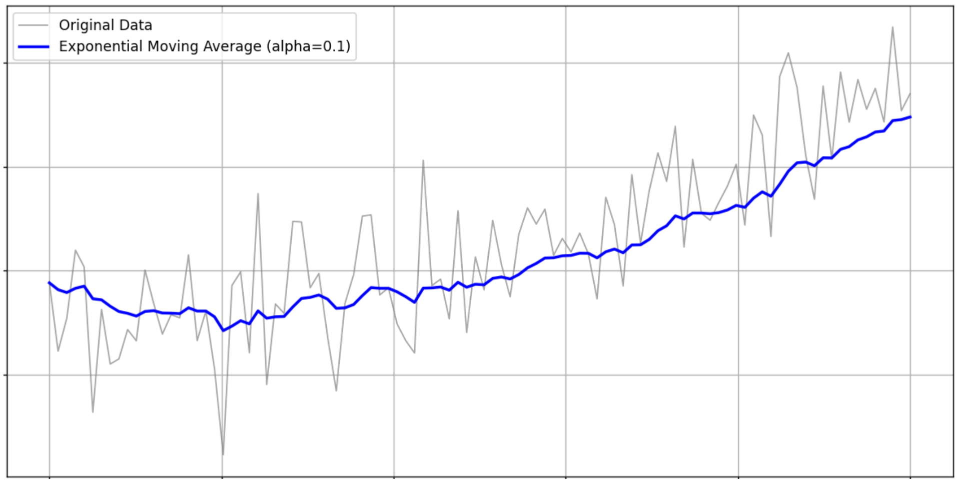


Figure 4: Example of exponential moving average (blue line)
Source: photographed by the author

The first and the second moments are statistical concepts. The first moment of data is their mean value, and the second moment is the variance, which indicates how spread out the data is around the mean value.

In the context of the Adam optimization algorithm, the first moment of gradients is used to estimate the mean value of gradients (which can be viewed as the "rate of change" of model parameters), and the second moment of gradients is used to estimate the variance of gradients (reflecting how gradients are spread out around the mean value).

The main idea behind using moments in the Adam algorithm is to provide the algorithm with additional information about previous weight updates and the gradient direction, enabling better control over the optimization process.

$$\begin{gathered} m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{gathered} \tag{3}$$

As mt and vt are initialized as vectors of zeros, they tend to be biased towards zero, especially during the initial time steps, and especially when the decay rates are small (i.e. β_1 and β_2 are close to 1). In the Adam algorithm, a bias correction is done by adjusting the estimates m_t and v_t by dividing them by $(1-\beta^t)$, where t represents the current step. This reduces the bias towards zero, ensuring that the initial parameter updates are more accurate.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{4}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{5}$$

Taking this correction into account, the parameter update rule takes the following form:

$$w_{t+1} = w_t - \frac{n}{\sqrt{v_t} + e} m_t, \tag{6}$$

6. Simple Adam example

In this section, we will provide an example of how the Adam algorithm works in its simplest form. We will consider a scenario where we have a simple function that needs to be optimized, and we will apply the Adam algorithm to find its minimum.

Firstly, we need to define the loss function. We will use a simple two-dimensional function that squares the input data and defines the range of input data from -1.0 to 1.0:

```
def loss_function(x, y):
  return x ** 2.0 + y ** 2.0
```

To visually observe the progress of the function, let's create a 2D plot:

```
bounds = asarray([[-1.0, 1.0], [-1.0, 1.0]])
xaxis = arange(bounds[0,0], bounds[0,1], 0.1)
yaxis = arange(bounds[1,0], bounds[1,1], 0.1)
x, y = meshgrid(xaxis, yaxis)
```

```
results = objective(x, y)
pyplot.contourf(x, y, results, levels=50, cmap='jet')
pyplot.show()
```

Executing this code snippet generates a two-dimensional contour plot of the objective loss function (Figure 5). This plot will serve as a visual representation of the points investigated throughout the search for the local minimum of the function.

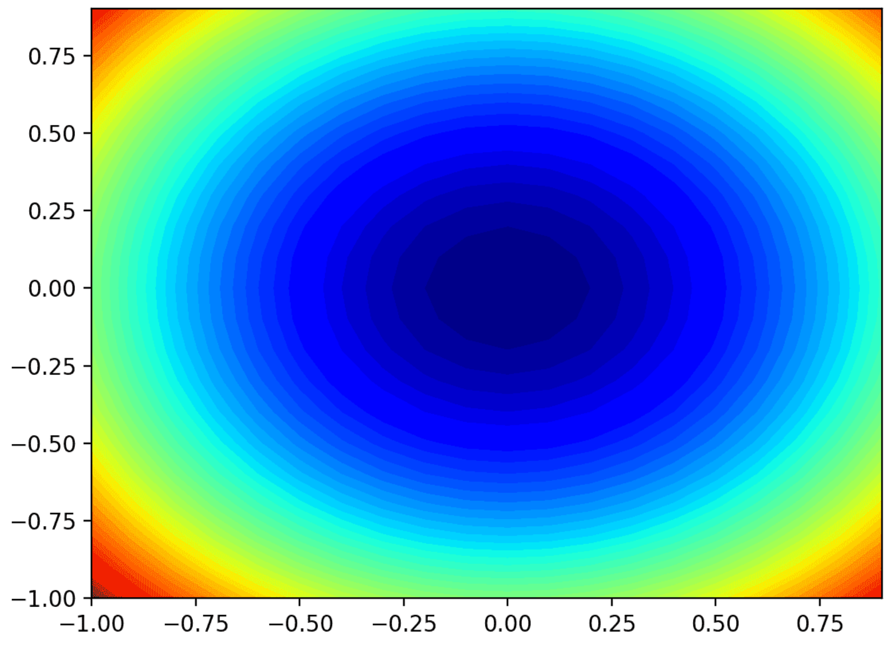


Figure 5: Two-dimensional plot of the loss function using Adam
Source: photographed by the author

Let's move on to the Adam algorithm. First, we initialize the first and second moments as zeros:

```
m = [0.0 for _ in range(bounds.shape[0])]
v = [0.0 for _ in range(bounds.shape[0])]
```

After that, we compute the gradient (derivative) of the data:

```
gradient = derivative(w[0], w[1])
```

Now we need to apply the Adam parameter update rule. While in practice, a matrix method is typically utilized for computation, for the sake of clarity in this example, we'll employ an iterative approach. Given we have two parameters, we'll use a loop to update both of them:

```
for i in range(x.shape[0]):
  m[i] = beta1 * m[i] + (1.0 - beta1) * g[i]
  v[i] = beta2 * v[i] + (1.0 - beta2) * g[i]**2
```

Then we apply the bias correction:

```
mhat = m[i] / (1.0 - beta1**(t+1))
vhat = v[i] / (1.0 - beta2**(t+1))
```

In the end we update the parameters of the model and calculate the loss:

```
w[i] = w[i] - alpha * mhat / (sqrt(vhat) + eps)
score = loss_function(w[0], w[1])
```

The Figure 6 illustrates the outcome of executing the code. The "Score" indicates the value of the loss function.

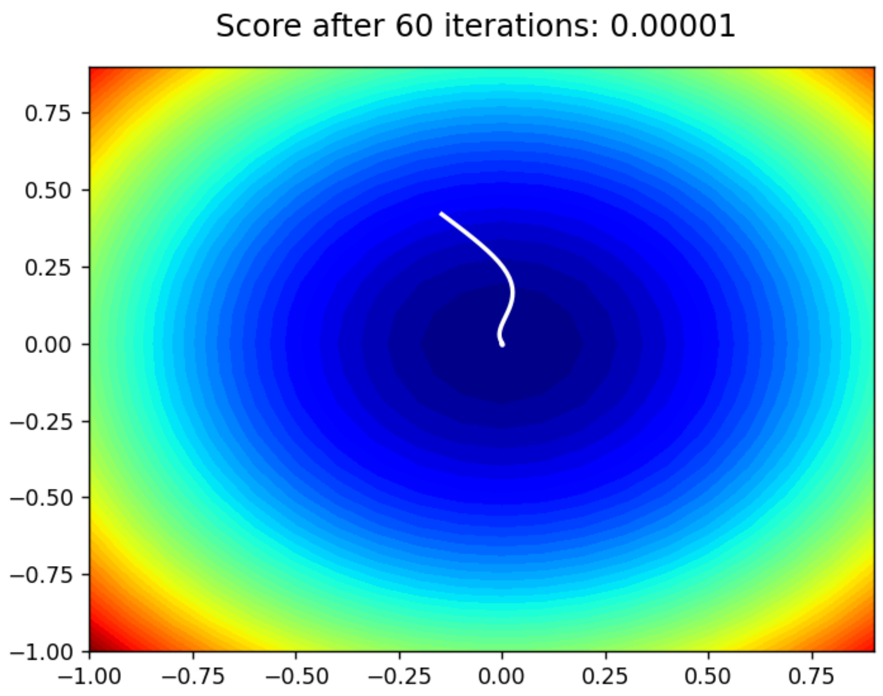


Figure 6: Two-dimensional graph of the loss function using Gradient Descent
Source: photographed by the author

For comparison, the gradient descent algorithm, with the same function and the same number of iterations, achieved significantly worse results (Figure 7).

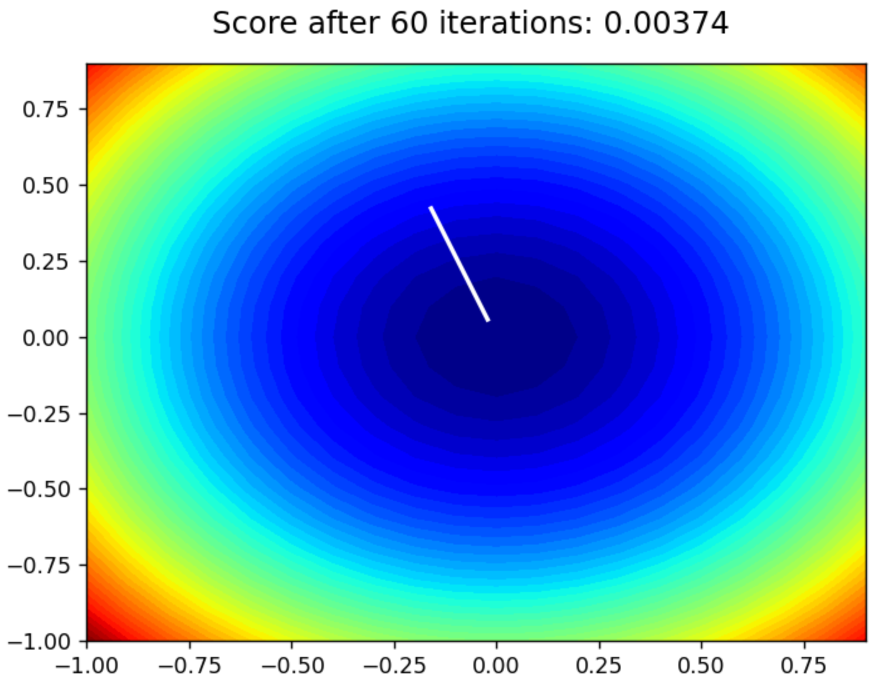


Figure 7: Graph illustrating the performance using Gradient descent algorithm
Source: photographed by the author

7. Comparing Adam with other algorithms

To measure the effectiveness and universality of the Adam, we compared it with other optimization algorithms on two datasets, MNIST and FashionMNIST. We chose them because they are often used as a benchmark for testing new machine learning algorithms and models.

MNIST (Modified National Institute of Standards and Technology) is a classic dataset consisting of 60,000 black and white images of handwritten digits in the training set and 10,000 images in the test set.

FashionMNIST is another popular dataset that contains 60,000 images in the training set and 10,000 images in the test set. The images represent various types of clothing items such as T-shirts, dresses, trousers, etc. This dataset is created for use in image classification tasks.

We chose to test the Adam algorithm on both MNIST and FashionMNIST datasets because they contain different kinds of data. MNIST has images of handwritten digits, while FashionMNIST consists of more complicated pictures of clothing items. By evaluating Adam's performance on these diverse datasets, we can see how well it works across different types of data, showing its usefulness in various situations.

Figure 8 illustrates the training process of models on the MNIST dataset using various optimization algorithms. The results indicate that, despite the task not being very complicated, Adam successfully demonstrated the best performance among all the algorithms. This is further evidenced in Figure 9, where the test accuracy of Adam surpasses that of all other algorithms.

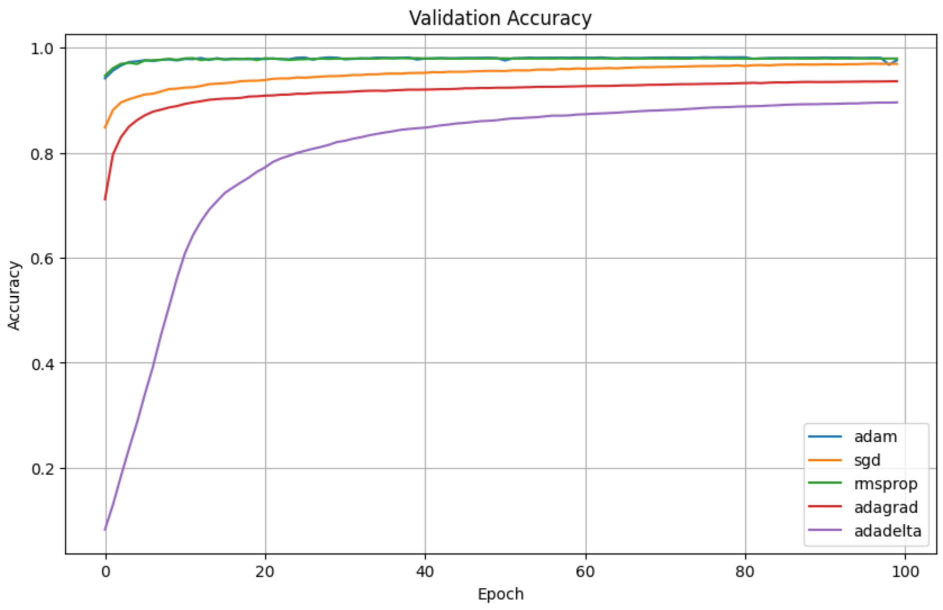
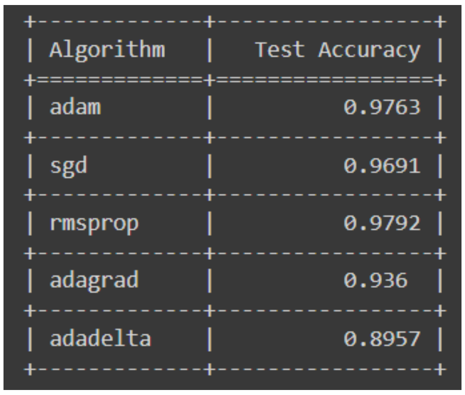


Figure 8: Graph illustrating the performance of different algorithms on MNIST dataset
Source: photographed by the author



```
+-------------+-----------------+
| Algorithm   |   Test Accuracy |
+=============+=================+
| adam        |          0.9763 |
+-------------+-----------------+
| sgd         |          0.9691 |
+-------------+-----------------+
| rmsprop     |          0.9792 |
+-------------+-----------------+
| adagrad     |          0.936  |
+-------------+-----------------+
| adadelta    |          0.8957 |
+-------------+-----------------+
```

Figure 9: Test accuracy of different algorithms on MNIST dataset
Source: photographed by the author

Figure 10 illustrates the training process of models on the FashionMNIST dataset, which is slightly more complex. Despite this complexity, Adam managed to outperform other algorithms, demonstrating its effectiveness even for more challenging tasks. This is further supported by Figure 11, where it is shown that the test accuracy of Adam exceeds that of all other algorithms.

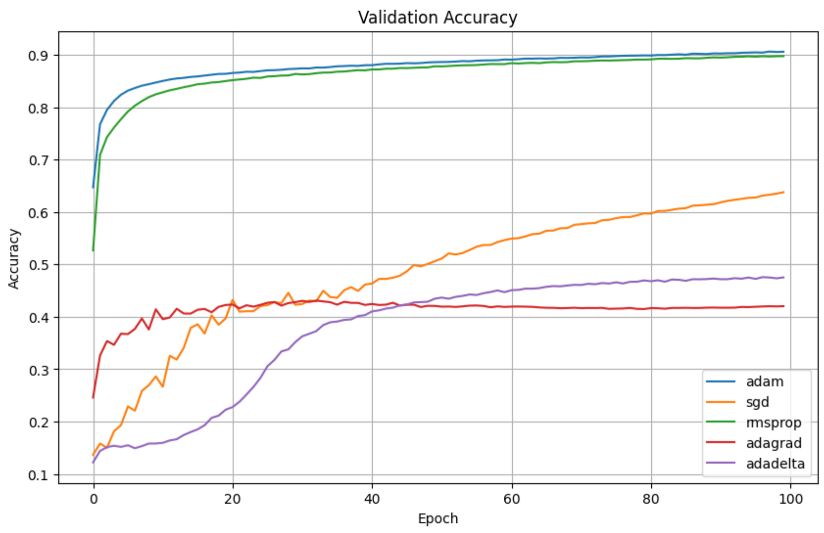
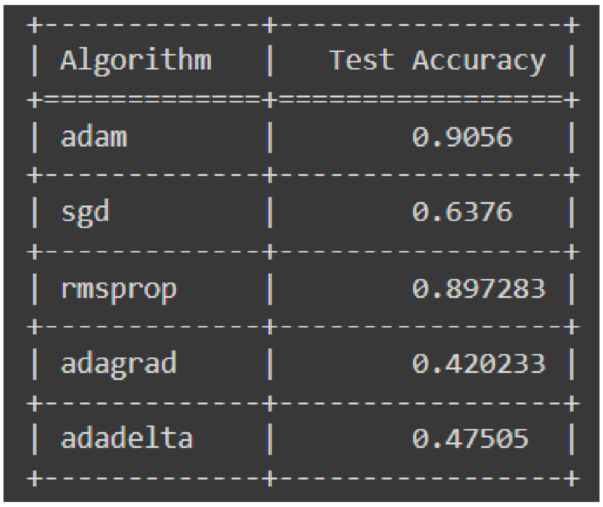


Figure 10: Graph illustrating the performance of different algorithms on FashionMNIST dataset
Source: photographed by the author



```
+-------------+-----------------+
| Algorithm   |   Test Accuracy |
+=============+=================+
| adam        |        0.9056   |
+-------------+-----------------+
| sgd         |        0.6376   |
+-------------+-----------------+
| rmsprop     |        0.897283 |
+-------------+-----------------+
| adagrad     |        0.420233 |
+-------------+-----------------+
| adadelta    |        0.47505  |
+-------------+-----------------+
```

Figure 11: Test accuracy of different algorithms on FashionMNIST dataset
Source: photographed by the author

Analysis of the experiment results conducted on various datasets, including MNIST and FashionMNIST indicates a significant advantage of the Adam optimization algorithm. Its effectiveness was demonstrated regardless of the complexity of object structures and the diversity of classes in the datasets. Interestingly, while some algorithms may have shown slightly better results on datasets with simpler structures and fewer classes, Adam proved to be more efficient in all modeled scenarios. Overall, Adam provided faster and higher-quality solutions to classification tasks compared to most other algorithms, confirming its advantages in machine learning.

To make sure our model works well for our specific needs, we made a small dataset ourselves. The famous MNIST dataset, created by American researchers, might not match our handwritten numbers perfectly. So, we wanted to see if our model could still understand and categorize our handwritten characters correctly. This way, we could check if our model is flexible and reliable for our purposes, not just for standard datasets.

The dataset consists of 100 handwritten numbers from 0 to 9 (Figure 12).

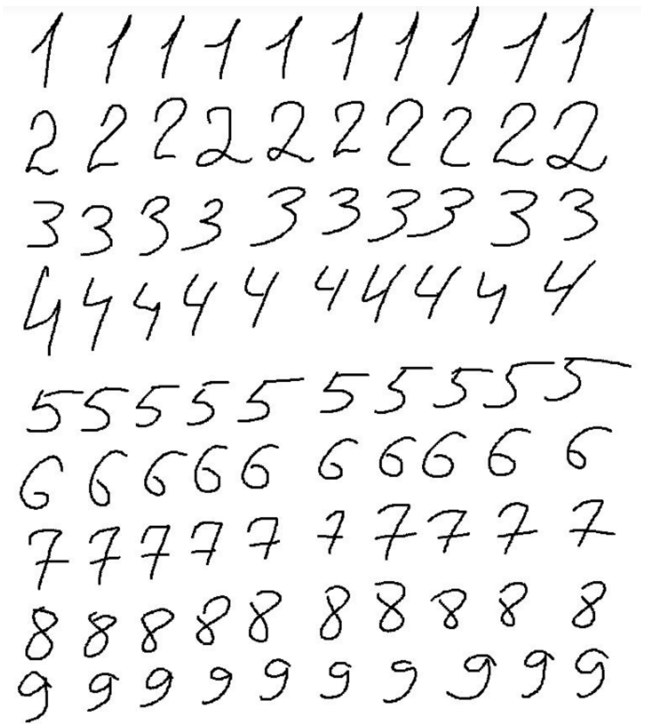


Figure 12: An example of our handwritten dataset
Source: photographed by the author

The results appear promising, with the model achieving an accuracy of 97%, meaning it correctly predicted 97 out of 100 images. This level of accuracy suggests that the model is performing well on our custom dataset, demonstrating its effectiveness in recognizing and classifying our handwritten numbers.

In further research, it is planned to use Adam's algorithm to analyze the data of cyberphysical systems [8, 9], biosensors [10] and the results of cardiac signal processing [11].

8. Conclusions

In this study, the Adam algorithm was investigated in the context of optimization in machine learning. The main conclusions and results of the study are as follows:

1. The Adam algorithm is an effective optimization method that combines ideas from other algorithms such as RMSProp and AdaGrad.
2. Experiments on various datasets, such as MNIST and FashionMNIST showed that the Adam algorithm is capable of achieving good results across a wide range of machine learning tasks.

3. The Adam algorithm is effective for optimizing tasks involving both large and small datasets, as demonstrated by experimental results.

9. References

[1] D. P. Kingma and J. L. Ba, Adam: a method for stochastic optimization, arXiv:1412.6980v9 [cs.LG], 2015.

[2] R. Zaheer and H. Shaziya, A Study of the Optimization Algorithms in Deep Learning, March 2020.

[3] H. Xiao, K. Rasul, and R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, arXiv preprint arXiv:1708.07747, 2017.

[4] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747, 2016.

[5] S. Wang, C. Li, X. Ding, Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis, arXiv:1802.09941v2 [cs.LG], 15 Sep 2018.

[6] J. Brownlee Gentle Introduction to the Adam Optimization Algorithm for Deep Learning, 2017, https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[7] J. Brownlee Code Adam Optimization Algorithm From Scratch, 2021, https://machinelearningmastery.com/adam-optimization-from-scratch/

[8] V. Martsenyuk, A. Sverstiuk, A. Klos-Witkowska, N.Kozodii, O. Bagriy-Zayats, I. Zubenko, Numerical analysis of results simulation of cyber-physical biosensor systems. CEUR Workshop Proceedings, 2019, 2516, pp. 149–164.

[9] V. Martsenyuk, A. Sverstiuk, O. Bahrii-Zaiats, A. Kłos-Witkowska, Qualitative and Quantitative Comparative Analysis of Results of Numerical Simulation of Cyber-Physical Biosensor Systems. (2022) CEUR Workshop Proceedings, 3309, pp. 134 – 149.

[10] V. Martsenyuk, A. Klos-Witkowska, S. Dzyadevych, A. Sverstiuk, Nonlinear Analytics for Electrochemical Biosensor Design Using Enzyme Aggregates and Delayed Mass Action. Sensors, 2022, 22(3), 980.

[11] V. Trysnyuk, A. Zozulia, S. Lupenko, I. Lytvynenko, A. Sverstiuk, Methods of rhythm-cardio signals processing based on a mathematical model in the form of a vector of stationary and stationary connected random sequences. CEUR Workshop Proceedings, 2021, 3021, pp. 197–205.