technologies

MDPI

Article

MAMGD: Gradient-Based Optimization Method Using Exponential Decay

Nikita Sakovich [1], Dmitry Aksenov [1], Ekaterina Pleshakova [2,*] and Sergey Gataullin [2]

[1] Financial University under the Government of the Russian Federation, Moscow 109456, Russia; 217556@edu.fa.ru (N.S.); daaksenov@fa.ru (D.A.)
[2] MIREA—Russian Technological University, 78 Vernadsky Avenue, Moscow 119454, Russia; sgataullin@cemi-ras.ru
* Correspondence: pleshakova@mirea.ru

Abstract: Optimization methods, namely, gradient optimization methods, are a key part of neural network training. In this paper, we propose a new gradient optimization method using exponential decay and the adaptive learning rate using a discrete second-order derivative of gradients. The MAMGD optimizer uses an adaptive learning step, exponential smoothing and gradient accumulation, parameter correction, and some discrete analogies from classical mechanics. The experiments included minimization of multivariate real functions, function approximation using multilayer neural networks, and training neural networks on popular classification and regression datasets. The experimental results of the new optimization technology showed a high convergence speed, stability to fluctuations, and an accumulation of gradient accumulators. The research methodology is based on the quantitative performance analysis of the algorithm by conducting computational experiments on various optimization problems and comparing it with existing methods.

Keywords: optimization; adaptive gradient methods; deep learning; neural networks; machine learning algorithms; gradient descent; comparative analysis

Citation: Sakovich, N.; Aksenov, D.; Pleshakova, E.; Gataullin, S. MAMGD: Gradient-Based Optimization Method Using Exponential Decay. *Technologies* **2024**, *12*, 154. https://doi.org/10.3390/technologies12090154

Received: 10 August 2024
Revised: 27 August 2024
Accepted: 31 August 2024
Published: 6 September 2024

1. Introduction

Deep learning is widely used to solve problems such as image recognition and classification [1–4], target localization [4,5], object detection [6–8], speech and action recognition [9,10], face recognition [11,12], and machine translation [13–15], improving the efficiency of agribusiness [16] and intelligent industrial robotics [17], as well as in the field of cybersecurity [18–21]. The model structure and optimization algorithm play a key role in the performance of a deep neural network. Based on this, the optimization algorithm significantly improves the performance and accuracy of deep neural network models. The choice of optimization algorithms is a priority in the model. In the case of a fixed network architecture and dataset, applying different optimization algorithms will give different training efficiencies.

Optimization methods, namely, gradient-based optimization methods, are a key part of training neural networks, which play an important role in various fields of science and business. With the development of deep learning and its widespread application in various tasks, there is a need for efficient optimization methods that can accelerate learning and improve the accuracy of models. Including optimization methods minimizes multivariate real functions. Because most mathematical models are some functions from several and usually from many variables, optimizers will help to find the extrema of such functions, which are solutions to problems that are applied in various fields of mathematical modeling. Neural networks are a branch of machine learning called deep machine learning. In deep learning, a neural network with a different number of layers and parameters is created, which generates a multidimensional function; in turn, the optimizer tries to change the parameters of the neural network so as to minimize the deviations of the predictions from

the training data. To measure such deviations, there are a large number of metrics, which are the target function for optimization and, more specifically, minimization. Currently, there are various optimization methods that combine the ideas of stochastic gradient descent, the adaptive learning rate, exponential smoothing, and the iterative correction of optimization algorithm parameters. Currently, the most popular optimization methods are SGD [22], Adagrad [23], RMSprop [24], and Adam [25], as well as Adam-based algorithms that apply various ideas to it, such as Nadam [22], AdaMax [25], and AdaFactor [26]. All these algorithms are implemented in the keras deep machine learning library and are used in training neural networks.

The authors of [27] propose a method that does not require computational costs and does not require backward iterations; this method is based on step-size adaptation, and the entire performance of the learning trajectory is considered as a function of the step size. Leslie N. Smith uses a cyclic learning rate method that does not need to find the best values and plan global learning rates, which reduces the number of iterations [28]. Zhibin Zhu et al. [29] propose a method that does not add additional computational effort, and the Powell restart strategy developed by the authors allows for improving numerical performance. The online adaptive gradient descent with communication event-triggered method [30] provides minimization of dynamic regret by a group of agents, which is a cumulative loss for agent evaluations of the optimal strategy.

SAM [31] is an efficient sharpness-aware minimization optimizer for training deep learning (DL) models. The main advantage of the optimizer is improved generalization for training deep learning models. The improvement is achieved by adding additional perturbation steps, which results in the flattening of the deep learning model landscape. The SAM optimizer is applied in various machine learning problems [32–35]. Recently, many new data-driven and network-based layout generation methods have been introduced in the field of deep learning, which provide performance improvements and automation after the training stage. The authors of [36] propose a new geometric representation that can encode correlations between objects, the ground, and the camera position.

Izuchukwu et al. [37] propose a method that demonstrates the improvement by including inertial terms in the reflected gradient projection methods, and the method includes one projection onto the feasible set and one cost operator estimate per iteration. Meruza Kubentayeva et al. [38] propose a method for solving dual optimization problems using accelerated gradient methods. They present a network equilibrium model, and this model is formulated as a convex minimization program. Xiaolin Zhou et al. [39] propose a reflected gradient projection method with a new step size to solve the variational inequality problem in Hilbert spaces. The authors of [40] propose a first-order gradient descent optimization algorithm based on NWM-Adam. The authors' proposed algorithm eliminates the undesirable convergence behavior of some optimization algorithms that use a fixed window of past gradients to scale gradient updates and improves the performance of Adam and AMSGrad.

Traditional pixel-based classifications face problems, such as misclassification, omissions, and noise. Object-based classifications effectively deal with these problems by taking into account the provided domain information and dividing the image into several objects based on the specified criteria. Regarding the importance of pixel-based and sub-pixel-based methods, the study by Valjarević et al. [41] provides a valuable insight.

This paper presents a new optimization method that combines the ideas of existing optimizers and new methods. The algorithm shows good convergence speed and accuracy results, including outperforming the current optimization methods with the proper selection of hyperparameters. The MAMGD optimizer uses an adaptive learning step, exponential smoothing and gradient accumulation, parameter correction, and some discrete analogies from classical mechanics. The MAMGD optimizer is designed to address the slow convergence and instability inherent in conventional gradient optimization methods, such as SGD and its variations. The main goal of MAMGD is to provide a high convergence speed and stability when dealing with different types of data, including those characterized

by high gradient variability. This is achieved by adaptively tuning the learning rate and using a discrete second-order representation of the derivatives to better account for the curvature of the loss function. Like all gradient-based optimization methods, the MAMGD method can be used for training neural networks, in optimization problems in which it is possible to take partial derivatives of the target function, and in mathematical modeling problems that are based on previous problems.

2. Materials and Methods

2.1. Classic Gradient Descent

The simplest and most basic algorithm is gradient descent [21], and all other optimization algorithms are formed based on this method. Classical gradient descent is easy to implement and understand, which makes it attractive to a wide range of users and researchers. Also, this method guarantees convergence to the local minimum of the loss function if certain conditions, such as the convexity and continuity of the loss function, are met (Algorithm 1).

Algorithm 1 Gradient descent

$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$
$\theta_t \leftarrow \theta_{t-1} - \eta g_t$

The classical gradient descent algorithm (Algorithm 1 Gradient descent) is simple to implement and can work well with a good choice of initial approximations and the learning rate parameter. However, gradient descent is very dependent on the choice of initial values and may be subject to build-ups in the local minima. All subsequent optimization methods are based on the idea of gradient descent and are improvements of its idea. To improve the gradient descent, derived optimization algorithms implement adaptive stepping and gradient accumulation methods to eliminate the problem of building up in the local minimum.

2.2. Momentum

The Momentum optimization method [22] eliminates some problems in classical gradient descent and allows for converging to a minimum faster. The idea of the optimizer is to accumulate gradients, i.e., the algorithm uses information about previous iterations of the algorithm (Algorithm 2).

Algorithm 2 Momentum

$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$
$m_t \leftarrow \beta m_{t-1} + g_t$
$\theta_t \leftarrow \theta_{t-1} - \eta m_t$

At the same time, the method may be subject to the overflight of the global minima due to the accumulation of gradients, which can also be strongly affected by explosive gradients. This will greatly increase the accumulation of gradients and the optimizer will jump the global minimum. The problem of sensitivity to the choice of the learning rate remains, just as with classical gradient descent; at high learning rates, the optimizer overshoots the global minima, and at low learning rates, it may converge to them more slowly but still faster than classical gradient descent because of the gradient accumulator. In any case, if the learning rate value is too small, learning may be slow and there is a risk of getting stuck in a local minimum. If the value is too large, oscillation or divergence of the algorithm may occur.

2.3. Adagrad

The optimization algorithm Adagrad [23] applies the idea of the adaptive learning rate, which can solve the problem of the very fast or very slow convergence of the method to a minimum. Adagrad uses an accumulator of gradient squares to accumulate velocities and adjusts the step depending on the gradient vector (Algorithm 3).

Algorithm 3 Adagrad

$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$
$n_t \leftarrow n_{t-1} + g_t^2$
$\theta_t \leftarrow \theta_{t-1} - \eta \frac{g_t}{\sqrt{n_t + \epsilon}}$

The method adjusts the gradient vector by removing the influence of large gradients and increasing the influence of small values. This idea solves the problem of discharged models with a large number of different values of weights and increases the convergence rate by changing the learning rate. But it also introduces the problem of explosive gradients, where the values can greatly increase the accumulator, leading to moving away from the global minimum or converging to a local minimum.

Also, Adagrad tends to have slow convergence and may even stop learning in some cases. This is due to the fact that weights that are rarely updated have small gradient values, as well as being due to the accumulation of the sum of squares of gradients in the denominator. Because of this, the method may "freeze" on the way to a local minimum.

2.4. RMSProp

In order to reduce the influence of explosive and damped gradients, the RMSProp optimization method [24] uses the idea of exponential smoothing. This helps to increase the influence of the accumulator of gradient squares or gradients at the current iteration step. At the same time, a new parameter of the optimizer, the smoothing factor, appears, which is responsible for adjusting the influence of the accumulator of gradients (Algorithm 4).

Algorithm 4 RMSProp

$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$
$n_t \leftarrow \beta n_{t-1} + (1 - \beta) g_t^2$
$\theta_t \leftarrow \theta_{t-1} - \eta \frac{g_t}{\sqrt{n_t + \epsilon}}$

The algorithm, like Adagrad, uses an adaptive learning step based on accumulators of gradient squares. The improvement of RMSProp compared to Adagrad is there is less influence of outliers on the optimization progress due to exponential smoothing. RMSProp is less prone to overshooting the minima and movement due to accumulated gradients but is also subject to oscillations. There is also a new exponential smoothing hyperparameter that can affect the convergence rate, which requires more parameter experiments when training the model, but can significantly speed up training or reduce oscillations and overshoots of global minima.

2.5. Adam

The Adam optimizer [25] combines the Momentum and RMSProp optimization methods. The method uses an accumulator of gradients, gradient squares, and exponential smoothing while adding a correction to the exponential smoothing coefficients to increase the impact of the first iteration steps for faster convergence to a minimum (Algorithm 5).

At the moment, the Adam optimizer is the most popular in training neural networks; this is due to its high convergence rate and low susceptibility to oscillations. At the same time, Adam is highly dependent on initial parameters, which, if properly selected experimentally, can significantly increase convergence to the global minimum. In any case, Adam

does not completely solve the problem of large gradient accumulations and damping, which, in turn, is greatly helped by the correction of the exponential damping coefficients. Generalizing, Adam combines the methods of adaptive step learning, accumulator gradients, exponential smoothing, and coefficient correction, which helps to solve the problems of a low convergence rate, overflight of the global minima, and susceptibility to oscillations.

Algorithm 5 Adam

$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$
$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
$n_t \leftarrow \beta_2 n_{t-1} + (1-\beta_2) g_t^2$
$\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$
$\hat{n}_t \leftarrow \frac{n_t}{1-\beta_2^t}$
$\theta_t \leftarrow \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{n}_t}+\epsilon}$

3. Algorithm

The MAMGD optimization method combines the ideas of the Adam algorithm and supplements them with the ideas of exponential damping and discrete acceleration, which is a discrete form of the second derivative of the gradient. These methods are used to reduce the influence of accumulated gradient accumulators when the number of iterations of the algorithm increases. So, if the damping hyperparameter is successfully chosen, the learning rate can be reduced in time and the minimum that can be passed when the accumulated gradients are large cannot be missed (Algorithm 6).

Algorithm 6 MAMGD

$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$
$\hat{\beta_1} \leftarrow \beta_1 exp(-kt)$
$\hat{\beta_2} \leftarrow \beta_2 exp(-kt)$
$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
$n_t \leftarrow \beta_2 n_{t-1} + (1-\beta_2) g_t^2$
$a \leftarrow \sqrt{(\frac{g_t - g_{t-1}}{\theta_t - \theta_{t-1} + \epsilon})^2 + n_t}$
$\hat{m}_t \leftarrow \frac{m_t}{1-\hat{\beta}_1^t}$
$\hat{a} \leftarrow \frac{a}{1-\hat{\beta}_2^t}$
$\theta_t \leftarrow \theta_{t-1} - \eta \frac{\hat{m}_t}{\hat{a}+\epsilon}$

The hyperparameters used are the learning rate η (usually taken as $\eta = 0.001$ in the initial approximation), the exponential smoothing coefficients $\beta_1, \beta_2 \in [0; 1)$, the exponential damping factor $k \in [0, 1)$, and $\epsilon = 1 \times 10^{-7}$, a small number for the prerotation of the division to zero. In most cases, to determine a successful attenuation coefficient is a separate problem. For very small values, the algorithm can still fly over the global minima as well as other optimization methods, and for very large values, very early damping may occur. As can be seen at each iteration, the smoothing coefficients are updated using the rule $\hat{\beta} \leftarrow \beta exp(-kt)$, and the more iterations that pass, the greater the influence of the current gradient and less that of the accumulator of gradients. This is performed in order to reduce the speed. Because the optimizer cannot know in advance when the global minimum will occur, the task of selecting a successful damping parameter is a purely heuristic problem.

MAMGD achieves a high convergence rate and stability through a combination of several methods. Exponential decay gradually reduces the influence of old gradients and accumulated values, contributing to a more accurate adaptation to current optimization conditions. The use of second-order derivative discretization to estimate the curvature of the loss function allows for a more precise adjustment of the optimization step, avoiding overly large or small corrections, which improves stability. The adaptive learning rate

adjusts according to the current state of gradients and accumulated values, enabling more efficient progress toward the minimum of the loss function.

The discrete second derivative of gradients in MAMGD is computed by calculating the difference between the current and previous gradients $g_t - g_{t-1}$, which is then divided by the difference in the corresponding parameters $\theta_t - \theta_{t-1}$. The result is squared to account for the magnitude of the gradient change and added to the accumulated value of n_t, which also accounts for past gradient squares. This approach allows us to take into account not only the current gradient change but also the accumulated effects, which provides a more accurate estimate of the curvature of the loss function and hence a more efficient adaptation of the optimization step.

The parameter correction in MAMGD is performed through normalization of the moments m_t and a (analogous to n_t). The correction by dividing by $(1 - \hat{\beta}_1^t)$ and $(1 - \hat{\beta}_2^t)$ compensates for the initial bias of the moments m_t and a, which avoids their underestimation in the early stages of optimization. This is important to ensure the correct adaptation of the learning rate in the initial iterations.

The MAMGD optimizer is less oscillatory due to the hyperparameter of exponential damping, which reduces the influence of accumulated gradients with increasing iterations. The difference from the other presented optimizers is the finite form of the weights change, which uses the idea of discrete acceleration, which gives some similarity to the Newton–Raphson method, but without searching for the Hessian of the error function; hence, the optimizer can be partially called quasi-Newtonian, with the difference in the formula and in the absence of additional memory and computation costs. On the other hand, due to discreteness, we have to store weights and gradients for the previous iteration, which will have a negative impact when the number of weights of the neural network is large.

The main parameters of MAMGD include the following:

1. The attenuation coefficients β_1 and β_2: They decrease exponentially with each iteration according to the formula $\hat{\beta_1} \leftarrow \beta_1 \exp(-kt)$ and $\hat{\beta_2} \leftarrow \beta_2 \exp(-kt)$, where k is the attenuation coefficient.
2. The learning rate coefficient η: This parameter is chosen similarly to other optimizers, such as Adam, and can be tuned empirically.
3. The smoothing parameter ϵ: This is used to avoid division by zero and is usually chosen to be very small (10^{-7}).

The parameters are chosen based on experiments with different datasets to ensure a balance between the convergence speed and stability of the optimization. One option for a parameter search is a grid search.

The advantages of the MAMGD optimization method may be its rather high convergence speed, resistance to oscillations, and accumulations of gradient accumulators, with these factors being closely related to exponential damping and the discrete analog of the second derivative of the gradients. The disadvantages are the possible high computational and memory costs, as it is necessary to store the results of gradient and weight calculations from previous iterations.

4. Experiments and Analysis of Results

For the experiments, the MAMGD optimization method was implemented using the deep machine learning libraries tensorflow and keras. The matplotlib library was used for plotting the function graphs. The Optuna library was used to select the optimal hyperparameters. The experiments included the minimization of multivariate real functions, approximation of functions using multilayer neural networks, and training of neural networks on popular classification and regression datasets.

4.1. Minimization of Functions

In this chapter, we will review the results of the MAMGD gradient method for minimizing various functions and compare it with the other popular methods discussed in the

previous chapters. The MSE (mean squared error) metric was used as an error estimate. The MSE is expressed as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where n is the number of observations, y_i is the actual value, and $\hat{y}_i$ is the predicted value.

The MSE (mean squared error) is a metric used to assess model quality in regression problems. It measures the mean square deviation of predicted values from actual values.

4.1.1. Quadratic Function of One Variable

A quadratic function of one variable is one of the most trivial variants for minimization testing, so it was chosen as the first test. In this case, we considered a function of the following form: $f(x) = x^2 - x$ (Figure 1).

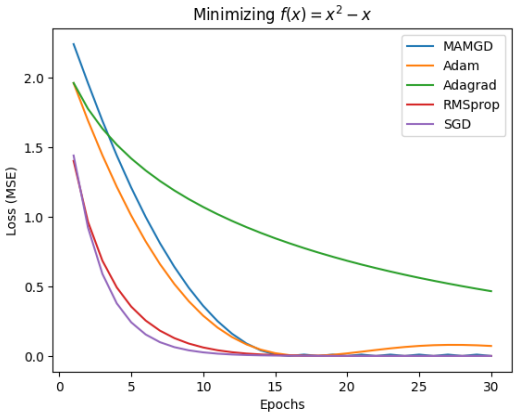

Figure 1. Quadratic function of one variable minimization.

The results of the graph and table (Table 1) of values show that the classical gradient descent and RMSprop method converged to the minimum the fastest, and the speed of the MAMGD and Adam methods were almost the same, but MAMGD was able to reach the minimum faster.

Table 1. Quadratic function.

Epoch	MAMGD Loss	Adam Loss	Adagrad Loss	RMSprop Loss	SGD Loss
2	1.959703	1.690626	1.775573	0.963417	0.921600
4	1.439806	1.214850	1.517669	0.492081	0.377487
6	0.999839	0.823812	1.333152	0.254789	0.154619
8	0.639871	0.516539	1.188714	0.128099	0.063332
10	0.359903	0.289402	1.070271	0.061027	0.025941
12	0.159935	0.135676	0.970315	0.027027	0.010625
14	0.039968	0.045391	0.884294	0.010932	0.004352
16	0.000000	0.005672	0.809204	0.003968	0.001783
18	0.000000	0.001759	0.742943	0.001268	0.000730
20	0.000000	0.018631	0.683974	0.000350	0.000299
22	0.000000	0.042904	0.631133	0.000081	0.000123
24	0.000000	0.064453	0.583516	0.000015	0.000050
26	0.000000	0.077261	0.540403	0.000002	0.000021
28	0.000000	0.079330	0.501213	0.000000	0.000008
30	0.000000	0.071832	0.465465	0.000000	0.000003

4.1.2. Sphere Function

The sphere function is a function of two variables and shows how gradient methods deal with multivariate optimization. In this case, we considered a function of the following form: $f(x) = x^2 + y^2$ (Figure 2).

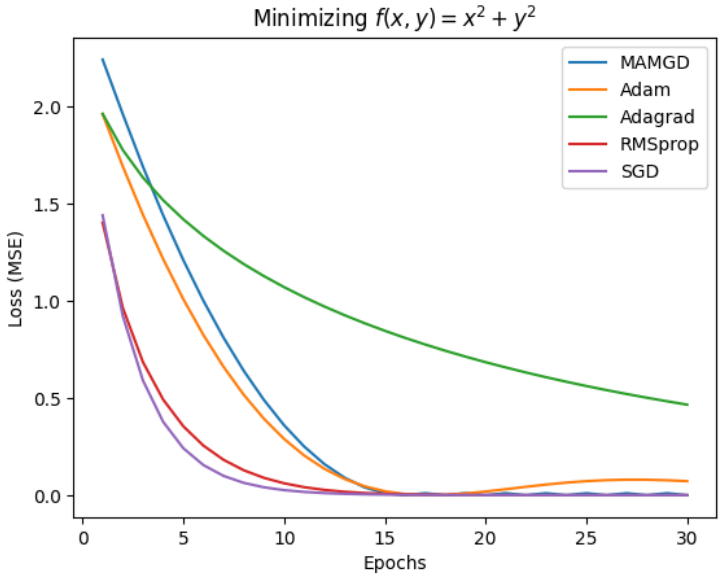

Figure 2. Sphere function minimization.

As in the previous case, the measurement results show that the MAMGD method quickly reaches a minimum when the exponential decay hyperparameter is properly chosen (Table 2). At the same time, the SGD and RMSprop methods are most effective at the initial stages.

Table 2. Spherical function.

Epoch	MAMGD Loss	Adam Loss	Adagrad Loss	RMSprop Loss	SGD Loss
2	1.959738	1.690627	1.775573	0.963417	0.921600
4	1.439837	1.214850	1.517669	0.492081	0.377487
6	0.999864	0.823812	1.333151	0.254789	0.154619
8	0.639892	0.516539	1.188714	0.128099	0.063332
10	0.359919	0.289402	1.070271	0.061027	0.025941
12	0.159946	0.135676	0.970315	0.027027	0.010625
14	0.039973	0.045391	0.884294	0.010932	0.004352
16	0.000000	0.005672	0.809204	0.003968	0.001783
18	0.000000	0.001759	0.742942	0.001268	0.000730
20	0.000000	0.018631	0.683973	0.000350	0.000299
22	0.000000	0.042904	0.631133	0.000081	0.000123
24	0.000000	0.064453	0.583516	0.000015	0.000050
26	0.000000	0.077261	0.540403	0.000002	0.000021
28	0.000000	0.079330	0.501213	0.000000	0.000008
30	0.000000	0.071832	0.465465	0.000000	0.000003

4.1.3. Three-Hump Camel Function

The three-hump camel function was chosen as the next function to be optimized, which is described as follows: $f(x, y) = 2x^2 - 1.05x^4 + \frac{x^6}{6} + xy + y^2$. It has several local minima and one global minimum. The main characteristics are as follows:

1. The area of definition is usually considered on $(x, y \in [-5, 5])$.

2. The global minimum is $(f(0,0) = 0)$.
3. It has three local minima, which gave the name of the function.

It can be seen from the data that the SGD and RMSprop methods quickly converged to a minimum, as in the previous cases (Figure 3). The Adam method underwent correction. The MAMGD method converged slower than the Adam method but was not corrected and obtained better results (Table 3).

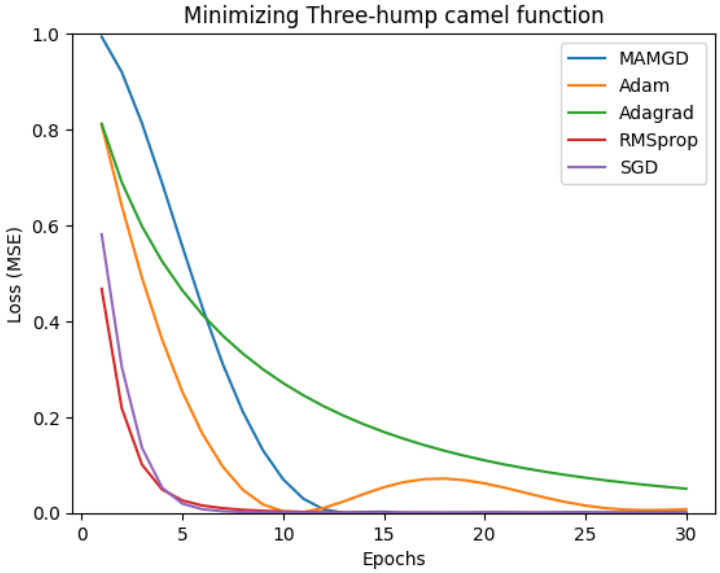

Figure 3. Three-hump camel function minimization.

Table 3. Three-hump camel function.

Epoch	MAMGD Loss	Adam Loss	Adagrad Loss	RMSprop Loss	SGD Loss
2	0.919596	0.640227	0.689822	0.218130	0.304278
4	0.688145	0.361751	0.524967	0.048839	0.052731
6	0.428438	0.165042	0.413728	0.015363	0.007620
8	0.211503	0.048304	0.332594	0.006501	0.001813
10	0.070354	0.003223	0.270984	0.002865	0.000712
12	0.007512	0.010078	0.223012	0.001171	0.000338
14	0.001704	0.039383	0.185010	0.000428	0.000168
16	0.000314	0.064289	0.154523	0.000138	0.000084
18	0.000040	0.071570	0.129823	0.000038	0.000042
20	0.001036	0.061574	0.109653	0.000009	0.000021
22	0.000542	0.042171	0.093073	0.000002	0.000011
24	0.001124	0.022718	0.079367	0.000000	0.000005
26	0.000768	0.009898	0.067981	0.000000	0.000003
28	0.001377	0.005689	0.058478	0.000000	0.000001
30	0.000959	0.007676	0.050514	0.000000	0.000001

4.1.4. Matyas Function

Further experiments were performed for the Matyas function, which is expressed as follows: $f(x,y) = 0.26(x^2 + y^2) - 0.48xy$. This function is less difficult to optimize than the three-hump camel function but is still of interest for testing algorithms, especially with respect to its accuracy and convergence rate near the minimum (Figure 4).

Some properties of the function are as follows:

Description:

1. The area of definition is usually considered on $(x, y \in [-10, 10])$.
2. The global minimum is $(f(0,0) = 0)$.

3. It is a unimodal function, which means it has only one minimum.
4. The function is symmetric about the origin.
5. Despite the simplicity of the formula, the Matyas function can be difficult for some optimization algorithms because of its flat region near the minimum.

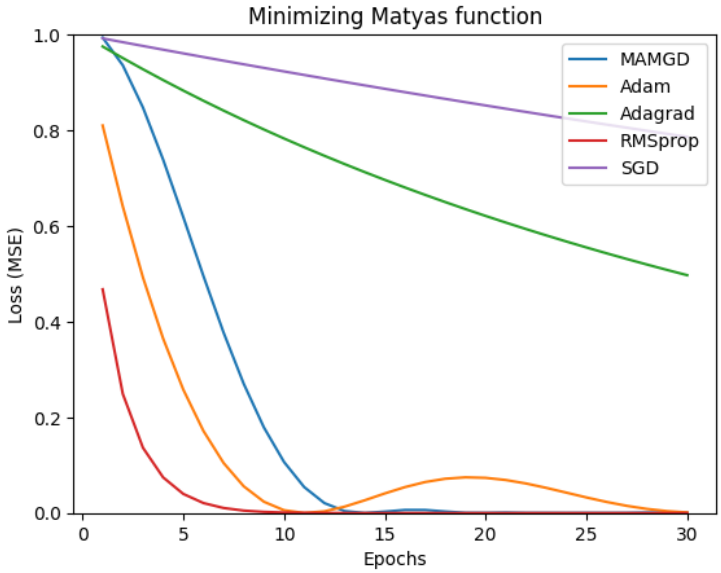

Figure 4. Matyas function minimization.

The results show that in this case, SGD did not show the best results, probably due to the flat region near the minimum of the function (Table 4). The best result was shown by the RMSprop method. Adam underwent correction as in the previous case. MAMGD achieved a good result at the 14th iteration, but after that it also underwent a small correction.

Table 4. Matyas function.

Epoch	MAMGD Loss	Adam Loss	Adagrad Loss	RMSprop Loss	SGD Loss
2	0.935771	0.640684	0.950924	0.248995	0.984096
4	0.738803	0.364773	0.904923	0.074502	0.968444
6	0.495462	0.171620	0.861725	0.020852	0.953042
8	0.270043	0.055840	0.821094	0.005086	0.937885
10	0.106671	0.005821	0.782821	0.001035	0.922968
12	0.020831	0.003469	0.746721	0.000169	0.908289
14	0.000015	0.026615	0.712628	0.000021	0.893844
16	0.006423	0.054194	0.680395	0.000002	0.879628
18	0.003157	0.071680	0.649889	0.000000	0.865638
20	0.000229	0.073522	0.620989	0.000000	0.851871
22	0.000168	0.061779	0.593588	0.000000	0.838322
24	0.000080	0.042687	0.567586	0.000000	0.824989
26	0.000106	0.023210	0.542893	0.000000	0.811868
28	0.000832	0.008589	0.519427	0.000000	0.798956
30	0.000208	0.001156	0.497113	0.000000	0.786249

4.1.5. Booth Function

Another function chosen for optimization was the Booth function, which is described as follows: $f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2$. The Booth function represents a good balance between simplicity and complexity, which makes it useful for testing optimization algorithms, especially when it is important to test the algorithm's ability to find a minimum that is not located in the center of the search area (Figure 5).

Some properties include the following:

1. The area of definition is usually considered on $(x, y \in [-10, 10])$.
2. The global minimum is $(f(1, 3) = 0)$.
3. It is a unimodal function, which means that it has only one minimum.

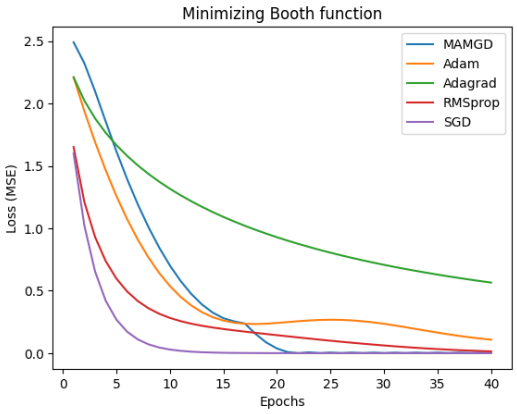

Figure 5. Booth function minimization.

As can be seen from the graph and measurement table, MAMGD accelerated sharply to the global minimum after 16 iterations; this result may be related to the choice of the exponential decay hyperparameter (Table 5). The SGD and RMSprop methods showed stable results, and Adam and Adagrad were rather slow to converge to the minimum in this function.

Table 5. Booth function.

Epoch	MAMGD Loss	Adam Loss	Adagrad Loss	RMSprop Loss	SGD Loss
2	2.323125	1.940616	2.023544	1.212334	1.024000
4	1.855067	1.464750	1.765136	0.737108	0.419431
6	1.394106	1.073439	1.580268	0.493918	0.171799
8	1.008421	0.765555	1.435507	0.359272	0.070369
10	0.701519	0.537251	1.316735	0.282088	0.028823
12	0.473483	0.381518	1.216434	0.235819	0.011806
14	0.324607	0.288052	1.130048	0.205457	0.004836
16	0.254285	0.243642	1.054571	0.182560	0.001981
18	0.153560	0.233271	0.987901	0.162734	0.000811
20	0.037101	0.241822	0.928501	0.144036	0.000332
22	0.000004	0.256036	0.875207	0.125871	0.000136
24	0.000409	0.266110	0.827116	0.108274	0.000056
26	0.000939	0.266470	0.783509	0.091500	0.000023
28	0.001095	0.255578	0.743802	0.075836	0.000009
30	0.001186	0.235005	0.707518	0.061524	0.000004
32	0.001243	0.208157	0.674258	0.048746	0.000002
34	0.001279	0.179021	0.643686	0.037620	0.000001
36	0.001303	0.151169	0.615514	0.028194	0.000000
38	0.001318	0.127124	0.589497	0.020447	0.000000
40	0.001329	0.108111	0.565423	0.014292	0.000000

4.1.6. Himmelblau's Function

The last function used for optimization was Himmelblau's function, which is described as follows: $f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$. Himmelblau's function is a popular choice for testing global optimization algorithms because it combines several complex

characteristics: multiple minima, symmetry, and nonlinearity. This makes it an excellent tool for evaluating the efficiency and robustness of optimization algorithms (Figure 6).

Properties:

1. The area of definition is usually considered on $(x, y \in [-5, 5])$.
2. For the global minima, the function has four identical local minima, all with the value 0: $[f(3.0, 2.0) = f(-2.805118, 3.131312) = f(-3.779310, -3.283186) = f(3.584428, -1.848126) = 0]$.
3. The function is multimodal, which means it has several local minima.
4. It has one local maximum at the point $(-0.270845, -0.923039)$ with a value of 181.617.

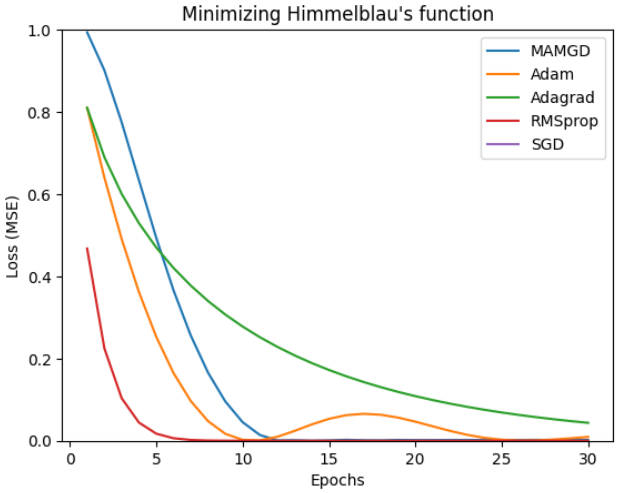

Figure 6. Himmelblau's function minimization.

The measurement results show that the SGD method began to move away from the local and global minima and was therefore excluded from the measurement table (Table 6). The RMSprop and MAMGD methods moved steadily toward the minimum, Adagrad also showed weak results, and the Adam method also underwent correction.

Table 6. Himmelblau's function.

Epoch	MAMGD Loss	Adam Loss	Adagrad Loss	RMSprop Loss
2	0.901249	0.640186	0.689926	0.224643
4	0.633748	0.361631	0.529265	0.044212
6	0.366014	0.164804	0.420394	0.006078
8	0.165643	0.047671	0.339941	0.000583
10	0.045176	0.002165	0.277845	0.000066
12	0.000891	0.009526	0.228666	0.000017
14	0.000068	0.039635	0.189065	0.000005
16	0.002296	0.062283	0.156812	0.000001
18	0.000905	0.063224	0.130337	0.000000
20	0.001584	0.046405	0.108481	0.000000
22	0.001781	0.024033	0.090367	0.000000
24	0.001766	0.007052	0.075311	0.000000
26	0.001884	0.000463	0.062774	0.000000
28	0.001978	0.002933	0.052320	0.000000
30	0.002048	0.009349	0.043596	0.000000

4.2. Approximation Using Multilayer Neural Network

A multilayer neural network was chosen to approximate the function $f(x) = x^2 + x - 1$ (Figure 7).

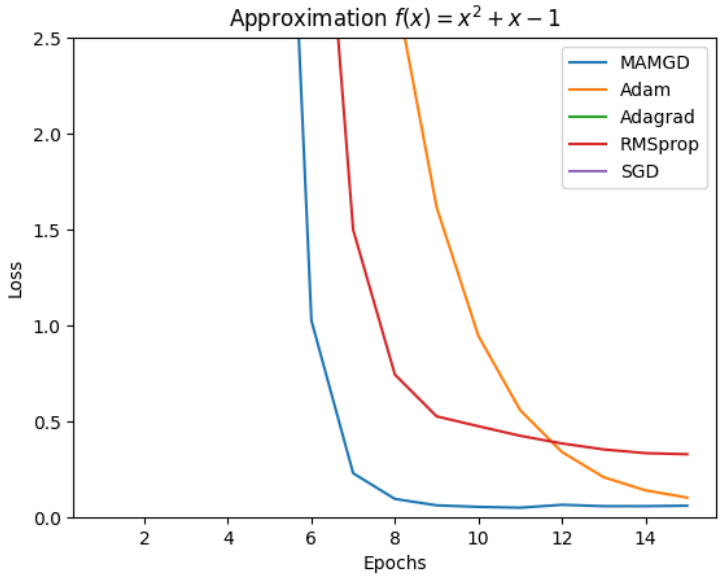

Figure 7. Neural network approximation.

As can be seen in the plot, MAMGD quickly converged to a minimum, which may have been influenced by the lucky exponential decay coefficient ($k = 0.0001$) and the discrete second derivative of the gradients (Table 7).

Table 7. Approximation using multilayer neural network.

Epoch	MAMGD Loss	Adam Loss	Adagrad Loss	RMSprop Loss	SGD Loss
1	821.636841	652.494873	1762.779175	655.232788	160.947144
2	111.159012	115.097504	1577.810669	112.074127	85.439232
3	49.195004	60.046024	1445.563843	55.858692	72.121544
4	18.571014	27.468365	1329.008545	24.525091	52.972687
5	5.915591	15.148467	1219.975220	11.512796	46.084648
6	1.025126	8.855510	1117.438354	4.281589	42.493931
7	0.227894	4.974255	1021.298096	1.498476	37.489067
8	0.093988	2.775730	931.561707	0.742871	49.106506
9	0.060457	1.615357	848.579773	0.524694	32.434353
10	0.052484	0.944320	772.376770	0.473356	32.483410
11	0.048284	0.557275	702.738586	0.423530	28.475237
12	0.063352	0.338988	639.490479	0.383350	31.728319
13	0.056214	0.207358	582.402405	0.352275	22.955385
14	0.056104	0.138861	531.055969	0.332885	32.714817
15	0.058539	0.100819	485.154205	0.327141	24.102997

4.3. MNIST Dataset

The MNIST dataset [42] is one of the most well-known and widely used datasets in the field of computer vision and neural network training. It consists of a set of images of handwritten digits from 0 to 9 that were created from a combination of handwritten samples from college students and staff at the National Institute of Standards and Technology (NIST).

The MNIST dataset contains a total of 60,000 training images and 10,000 test images; each image is 28 × 28 pixels. The images have been pre-processed to unify the size and center the digits in the center of each image.

Each image in the dataset has a label associated with it, which indicates which digit is depicted in the image (from 0 to 9). These labels are used to train the neural network and to evaluate its performance. MNIST is a popular choice for training neural networks because it is relatively small and easily accessible. It is also a convenient dataset for testing

and comparing different neural network algorithms and architectures in handwritten digit recognition (Figure 8).

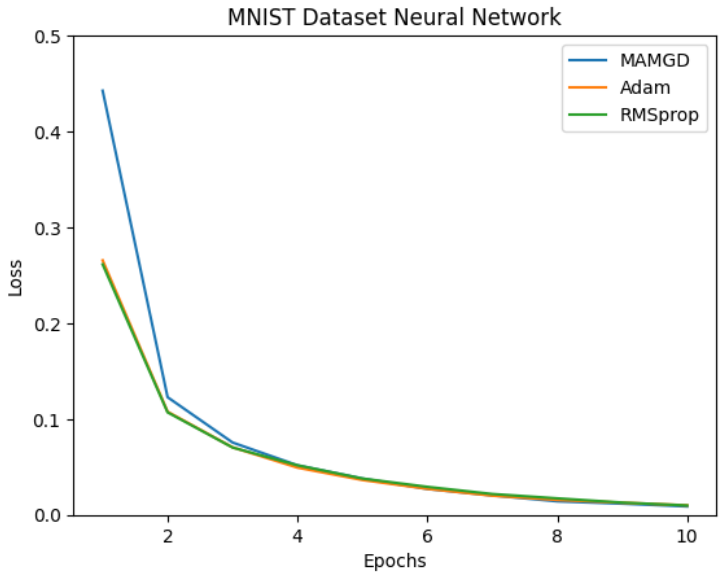

Figure 8. Training a neural network for digit classification on the MNIST dataset.

As can be seen, the MAMGD optimizer initially lags behind the Adam and RMSProp optimizer but soon catches up and converges to a minimum. This shows the problem of initial approximations and the choice of hyperparameters, such as the exponential decay factor (Table 8).

Table 8. MNIST dataset.

Epoch	MAMGD Loss	Adam Loss	RMSprop Loss
1	0.448672	0.266809	0.263463
2	0.124826	0.107094	0.105576
3	0.077700	0.069585	0.070089
4	0.052628	0.049328	0.050346
5	0.037809	0.037673	0.037978
6	0.026434	0.028731	0.028972
7	0.020605	0.020112	0.021667
8	0.015816	0.015615	0.016610
9	0.011611	0.012404	0.012298
10	0.008266	0.010504	0.008964

4.4. IMDB Dataset

The IMDB (Internet Movie Database) dataset is one of the most widely used and popular datasets for training neural networks in the tasks of text tone analysis and classification of movie reviews.

The IMDB dataset consists of 50,000 movie reviews, which have been divided into two parts: 25,000 reviews are for training and the remaining 25,000 are for testing the classifier. Each review has its own class label indicating whether the review is positive or negative. Positive reviews are labeled 1, while negative reviews are labeled 0.

The IMDB dataset is a good choice for training neural networks in text classification and tone analysis tasks, as it contains balanced data with a large number of examples for each class and also displays real user reviews of movies (Figure 9).

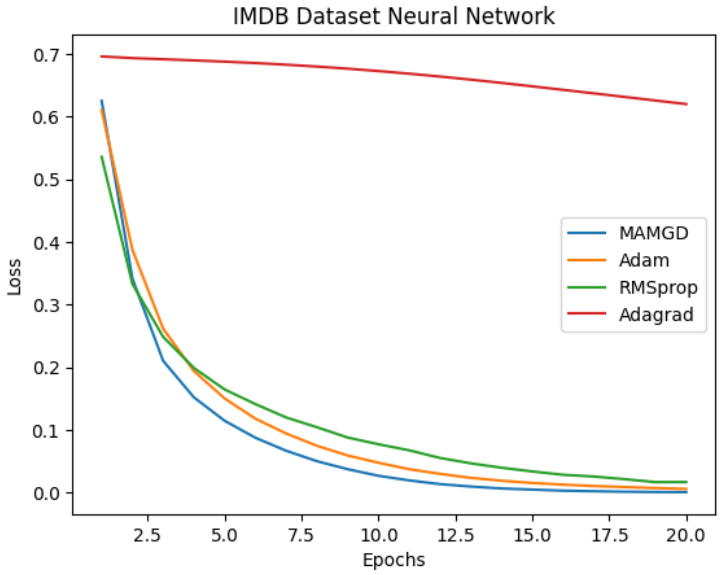

Figure 9. Training a neural network to classify reviews on the IMDB dataset.

The MAMGD optimization method shows a good and high convergence rate. A successful choice of hyperparameters allowed for fast convergence to the minimum while avoiding oscillations. This could also be influenced by the discrete derivative of the gradients, which accelerated the descent to the minimum of the target function (Table 9).

Table 9. IMDB dataset.

Epoch	MAMGD Loss	Adam Loss	Adagrad Loss	RMSprop Loss
1	0.637524	0.587573	0.691539	0.571567
2	0.373726	0.363699	0.689856	0.359916
3	0.232306	0.250259	0.688095	0.265192
4	0.167434	0.193303	0.686177	0.214027
5	0.127229	0.152671	0.684051	0.178158
6	0.098317	0.123702	0.681721	0.156204
7	0.075212	0.100980	0.679159	0.133551
8	0.057454	0.082415	0.676337	0.116825
9	0.043390	0.066716	0.673241	0.104766
10	0.032139	0.054332	0.669879	0.087954
11	0.023305	0.044377	0.666331	0.078029
12	0.016889	0.034820	0.662608	0.067212
13	0.012188	0.027816	0.658768	0.058585
14	0.008631	0.022097	0.654841	0.048537
15	0.005894	0.016978	0.650807	0.044989
16	0.004280	0.013270	0.646712	0.034727
17	0.002633	0.010459	0.642588	0.030603
18	0.001816	0.008271	0.638408	0.025305
19	0.001069	0.006590	0.634208	0.023882
20	0.000751	0.005273	0.630040	0.018804

4.5. Reuters Dataset

The Reuters dataset is one of the most popular datasets for training neural networks in natural language processing. It consists of news articles that were collected by Reuters between 1987 and 1996.

The dataset contains a variety of news categories, such as politics, economics, sports, science, and others.

The overall structure of the Reuters dataset has several key features: 1. Size: The Reuters Dataset consists of more than 10,000 news articles. It includes enough variation in neural network training data to allow researchers and developers to conduct different experiments and testing. 2. Multiple classes: There are 46 different classes of news in the Reuters dataset. Each article belongs to one of these classes, allowing for a multi-class classification task (Figure 10).

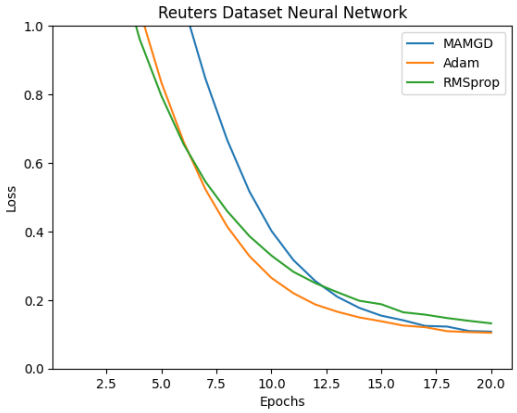

Figure 10. Training a neural network to classify Reuters texts.

As in the MNIST dataset, the optimizer initially lagged behind the other algorithms but later overtook RMSProp, probably due to the choice of hyperparameters (Table 10).

Table 10. Reuters dataset.

Epoch	MAMGD Loss	Adam Loss	RMSprop Loss
1	3.785023	3.380734	2.784578
2	3.470449	2.057770	1.560803
3	2.726303	1.371544	1.194215
4	1.900168	1.046967	0.962105
5	1.369266	0.833554	0.794977
6	1.062885	0.662638	0.655288
7	0.845602	0.522731	0.545252
8	0.665268	0.413226	0.458509
9	0.516712	0.328357	0.386447
10	0.402084	0.264667	0.329770
11	0.316678	0.219969	0.282453
12	0.255255	0.186874	0.249277
13	0.209545	0.165941	0.222823
14	0.177048	0.149283	0.197969
15	0.154396	0.138011	0.187647
16	0.140863	0.125612	0.164271
17	0.124534	0.120747	0.157349
18	0.122545	0.109029	0.147332
19	0.109503	0.106501	0.139231
20	0.107755	0.104526	0.132018

4.6. Boston Housing Price Dataset

Boston Housing Price is a dataset that contains information about housing prices in different neighborhoods of Boston, USA. It consists of various attributes, such as the average number of rooms in a house, percentage of the population with low social status, crime rate, etc.

This dataset is often used to train neural networks as it provides a good set of diverse attributes that can be used to predict real estate prices. Neural networks can be trained on this dataset to learn identification tasks.

It is important to understand the complex relationships between attributes and house prices. Using the Boston Housing Price dataset to train neural networks allows for the creation of a model that will be able to make real estate price predictions based on various factors. This can be useful for real estate agents, real estate market analysts, and other interested parties who want information about expected real estate prices in different Boston neighborhoods (Figure 11).

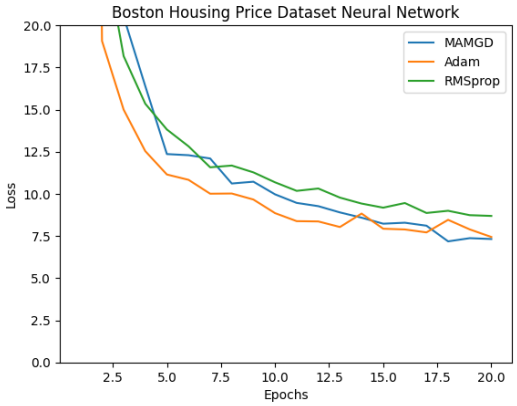

Figure 11. Training of the neural network for real estate price regression of the Boston Housing Price database.

In this case, the MAMGD optimization method was very similar to the Adam algorithm, which was influenced by the small value of the exponential decay coefficient (Table 11).

Table 11. Boston Housing Price dataset.

Epoch	MAMGD Loss	Adam Loss	RMSprop Loss
1	350.594757	140.978012	157.523834
2	33.531769	19.073381	25.266592
3	20.512051	14.994828	18.173952
4	16.396505	12.534935	15.347956
5	12.359325	11.147352	13.817107
6	12.293142	10.832203	12.820888
7	12.098958	10.008436	11.574941
8	10.613210	10.020133	11.678750
9	10.721079	9.662991	11.273701
10	9.970358	8.856193	10.683825
11	9.463201	8.380616	10.177731
12	9.269162	8.364103	10.316528
13	8.898938	8.032142	9.776771
14	8.586391	8.830625	9.423545
15	8.228765	7.930937	9.180243
16	8.287912	7.887546	9.455106
17	8.110354	7.714012	8.865941
18	7.180910	8.458524	8.996699
19	7.370471	7.891291	8.734899
20	7.320936	7.440734	8.690137

5. Discussion

The proposed MAMGD algorithm is an interesting optimization approach that combines several new ideas in the field of gradient descent. However, its performance and applicability deserve further discussion.

First, the use of exponential decay for the coefficients β_1 and β_2 is a novel approach. It potentially allows for the algorithm to better adapt to changing optimization landscapes, especially in problems with non-stationary objective functions. However, reducing these coefficients may lead to unnecessary slowdown in speed at the end of training. Consequently, the choice of parameter k to control the decay rate becomes critical and may require careful tuning.

The inclusion of an estimate of the second derivative in the form of a term $(\frac{g_t - g_{t-1}}{\theta_t - \theta_{t-1} + \epsilon})^2$ is another novel approach to adapting the learning step. It is reminiscent of the idea behind Newton's method but implemented in a more computationally efficient form. However, such second-order approximations can be unstable in high-dimensional parameter spaces typical of deep neural networks. This approach, too, requires additional research and experimentation.

In conclusion, it is worth noting that the proposed methods require further empirical verification and discussion of the results.

6. Conclusions

Despite significant progress in neural network optimization methods, the existing algorithms still face challenges, such as slow convergence and instability when training complex architectures. Our research aimed to overcome these limitations, potentially leading to more efficient and robust methods for training neural networks.

In this paper, we have analyzed existing neural network optimization methods, discussed their strengths and weaknesses, and presented a new optimization method based on exponential fading and the adaptive learning rate using a discrete derivative of second-order gradients.

Our experiments demonstrate that the proposed method has the potential to improve the learning process of neural networks, especially in problems requiring fast convergence and robustness to oscillations. However, the results also indicate that in some scenarios, the benefits of the new method may be limited compared to state-of-the-art optimizers.

This study raises the important question of whether current optimizers have reached the limit in improving the idea of gradient descent without increasing computational and memory costs. We hypothesize that future research could focus on exploring methods based on evolutionary algorithms or other approaches that can significantly increase the convergence rate and improve the accuracy of neural network training results.

7. Extensions

AdaExp

We apply exponential fading to Adam's gradient-based optimization method and then we obtain the following algorithm: (Algorithm 7).

Algorithm 7 AdaExp

$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$
$\hat{\beta_1} \leftarrow \beta_1 exp(-kt)$
$\hat{\beta_2} \leftarrow \beta_2 exp(-kt)$
$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
$n_t \leftarrow \beta_2 n_{t-1} + (1 - \beta_2) g_t^2$
$\hat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t}$
$\hat{n}_t \leftarrow \frac{n_t}{1 - \beta_2^t}$
$\theta_t \leftarrow \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{n}_t} + \epsilon}$

Author Contributions: Data curation, Investigation, Software—N.S.; Conceptualization, Methodology, Validation—D.A., N.S.; Formal analysis, Project administration, Writing—original draft—E.P.; Supervision, Writing—review editing—S.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data that support the findings of this study are openly 17 July 2024 https://github.com/NekkittAY/MAMGD_Optimizer (accessed on 27 August 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Seo, S.; Kim, J. Efficient weights quantization of convolutional neural networks using kernel density estimation based non-uniform quantizer. *Appl. Sci.* **2019**, *9*, 2559. [CrossRef]
2. Pan, H.; Pang, Z.; Wang, Y.; Wang, Y.; Chen, L. A new image recognition and classification method combining transfer learning algorithm and mobilenet model for welding defects. *IEEE Access* **2020**, *8*, 119951–119960. [CrossRef]
3. Wang, P.; Fan, E.; Wang, P. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognit. Lett.* **2021**, *141*, 61–67. [CrossRef]
4. Paramonov, A.A.; Nguyen, V.M.; Nguyen, M.T. Multi-task neural network for solving the problem of recognizing the type of QAM and PSK modulation under parametric a priori uncertainty. *Russ. Technol. J.* **2023**, *11*, 49–58. [CrossRef]
5. Hou, F.; Lei, W.; Li, S.; Xi, J. Deep learning-based subsurface target detection from GPR scans. *IEEE Sens. J.* **2021**, *21*, 8161–8171. [CrossRef]
6. Liu, Y.; Sun, P.; Wergeles, N.; Shang, Y. A survey and performance evaluation of deep learning methods for small object detection. *Expert Syst. Appl.* **2021**, *172*, 114602. [CrossRef]
7. Ghasemi, Y.; Jeong, H.; Choi, S.H.; Park, K.B.; Lee, J.Y. Deep learning-based object detection in augmented reality: A systematic review. *Comput. Ind.* **2022**, *139*, 103661. [CrossRef]
8. Khalid, S.; Oqaibi, H.M.; Aqib, M.; Hafeez, Y. Small pests detection in field crops using deep learning object detection. *Sustainability* **2023**, *15*, 6815. [CrossRef]
9. Yang, M.; Wu, C.; Guo, Y.; Jiang, R.; Zhou, F.; Zhang, J.; Yang, Z. Transformer-based deep learning model and video dataset for unsafe action identification in construction projects. *Autom. Constr.* **2023**, *146*, 104703. [CrossRef]
10. Priyadarshini, I.; Sharma, R.; Bhatt, D.; Al-Numay, M. Human activity recognition in cyber-physical systems using optimized machine learning techniques. *Clust. Comput.* **2023**, *26*, 2199–2215. [CrossRef]
11. Boutros, F.; Struc, V.; Fierrez, J.; Damer, N. Synthetic data for face recognition: Current state and future prospects. *Image Vis. Comput.* **2023**, *135*, 104688. [CrossRef]
12. Hwang, R.H.; Lin, J.Y.; Hsieh, S.Y.; Lin, H.Y.; Lin, C.L. Adversarial patch attacks on deep-learning-based face recognition systems using generative adversarial networks. *Sensors* **2023**, *23*, 853. [CrossRef]
13. Mercha, E.M.; Benbrahim, H. Machine learning and deep learning for sentiment analysis across languages: A survey. *Neurocomputing* **2023**, *531*, 195–216. [CrossRef]
14. Khan, W.; Daud, A.; Khan, K.; Muhammad, S.; Haq, R. Exploring the frontiers of deep learning and natural language processing: A comprehensive overview of key challenges and emerging trends. *Nat. Lang. Process. J.* **2023**, *4*, 100026. [CrossRef]
15. Mehrish, A.; Majumder, N.; Bharadwaj, R.; Mihalcea, R.; Poria, S. A review of deep learning techniques for speech processing. *Inf. Fusion* **2023**, *99*, 101869. [CrossRef]
16. Andriyanov, N.; Khasanshin, I.; Utkin, D.; Gataullin, T.; Ignar, S.; Shumaev, V.; Soloviev, V. Intelligent System for Estimation of the Spatial Position of Apples Based on YOLOv3 and Real Sense Depth Camera D415. *Symmetry* **2022**, *14*, 148. [CrossRef]
17. Osipov, A.V.; Pleshakova, E.S.; Gataullin, S.T. Production processes optimization through machine learning methods based on geophysical monitoring data. *Comput. Opt.* **2024**, *48*, 633–642. [CrossRef]
18. Ivanyuk, V. Forecasting of digital financial crimes in Russia based on machine learning methods. *J. Comput. Virol. Hacking Tech.* **2023**, 1–14. [CrossRef]
19. Boltachev, E. Potential cyber threats of adversarial attacks on autonomous driving models. *J. Comput. Virol. Hacking Tech.* **2023**, 1–11. [CrossRef]
20. Efanov, D.; Aleksandrov, P.; Mironov, I. Comparison of the effectiveness of cepstral coefficients for Russian speech synthesis detection. *J. Comput. Virol. Hacking Tech.* **2023**, 1–8. [CrossRef]
21. Pleshakova, E.; Osipov, A.; Gataullin, S.; Gataullin, T.; Vasilakos, A. Next gen cybersecurity paradigm towards artificial general intelligence: Russian market challenges and future global technological trends. *J. Comput. Virol. Hacking Tech.* **2024**, 1–12. [CrossRef]
22. Dozat, T. Incorporating nesterov momentum into adam. In Proceedings of the 4th International Conference on Learning Representations, Workshop Track, San Juan, Puerto Rico, 2–4 May 2016; pp. 1–4.
23. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.

24. Hinton, G.; Srivastava, N.; Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. **2012**, *14*, 2. Available online: https://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf (accessed on 9 August 2024).
25. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
26. Shazeer, N.; Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 4596–4604.
27. Massé, P.Y.; Ollivier, Y. Speed learning on the fly. *arXiv* **2015**, arXiv:1511.02540.
28. Smith, L.N. Cyclical learning rates for training neural networks. In Proceedings of the 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, USA, 24–31 March 2017; pp. 464–472.
29. Zhu, Z.; Zhu, X.; Tan, Z. An accelerated conjugate gradient method with adaptive two-parameter with applications in image restoration. *Comput. Appl. Math.* **2024**, *43*, 116. [CrossRef]
30. Okamoto, K.; Hayashi, N.; Takai, S. Distributed Online Adaptive Gradient Descent With Event-Triggered Communication. *IEEE Trans. Control Netw. Syst.* **2024**, *11*, 610–622. [CrossRef]
31. Foret, P.; Kleiner, A.; Mobahi, H.; Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In Proceedings of the International Conference on Learning Representations, Virtual, 3–7 May 2021.
32. Sun, H.; Shen, L.; Zhong, Q.; Ding, L.; Chen, S.; Sun, J.; Li, J.; Sun, G.; Tao, D. Adasam: Boosting sharpness-aware minimization with adaptive learning rate and momentum for training deep neural networks. *Neural Netw.* **2024**, *169*, 506–519. [CrossRef]
33. Ganesha, T.; Prakash, S.B.; Rani, S.S.; Ajith, B.S.; Patel, G.M.; Samuel, O.D. Biodiesel yield optimization from ternary (animal fat-cotton seed and rice bran) oils using response surface methodology and grey wolf optimizer. *Ind. Crops Prod.* **2023**, *206*, 117569. [CrossRef]
34. Kim, S.; Jang, M.G.; Kim, J.K. Process design and optimization of single mixed-refrigerant processes with the application of deep reinforcement learning. *Appl. Therm. Eng.* **2023**, *223*, 120038. [CrossRef]
35. Sigue, S.; Abderafi, S.; Vaudreuil, S.; Bounahmidi, T. Design and steady-state simulation of a CSP-ORC power plant using an open-source co-simulation framework combining SAM and DWSIM. *Therm. Sci. Eng. Prog.* **2023**, *37*, 101580. [CrossRef]
36. Sheng, Y.; Liu, Y.; Zhang, J.; Yin, W.; Oztireli, A.C.; Zhang, H.; Lin, Z.; Shechtman, E.; Benes, B. Controllable shadow generation using pixel height maps. In *European Conference on Computer Vision*; Springer Nature: Cham, Switzerland, 2022; pp. 240–256.
37. Izuchukwu, C.; Shehu, Y. A new inertial projected reflected gradient method with application to optimal control problems. *Optim. Methods Softw.* **2024**, *39*, 197–226. [CrossRef]
38. Kubentayeva, M.; Yarmoshik, D.; Persiianov, M.; Kroshnin, A.; Kotliarova, E.; Tupitsa, N.; Pasechnyuk, D.; Gasnikov, A.; Shvetsov, V.; Baryshev, L.; et al. Primal-dual gradient methods for searching network equilibria in combined models with nested choice structure and capacity constraints. *Comput. Manag. Sci.* **2024**, *21*, 15. [CrossRef]
39. Zhou, Z.; Cai, C.; Tan, B.; Dong, Q. A modified generalized version of projected reflected gradient method in Hilbert spaces. *Numer. Algorithms* **2024**, *95*, 117–147. [CrossRef]
40. Yu, Y.; Liu, F. Effective Neural Network Training with a New Weighting Mechanism-Based Optimization Algorithm. *IEEE Access* **2019**, *7*, 72403–72410. [CrossRef]
41. Valjarević, A.; Djekić, T.; Stevanović, V.; Ivanović, R.; Jandziković, B. GIS numerical and remote sensing analyses of forest changes in the Toplica region for the period of 1953–2013. *Appl. Geogr.* **2018**, *92*, 131–139. [CrossRef]
42. Cohen, G.; Afshar, S.; Tapson, J.; Schaik, A.V. Emnist: Extending mnist to handwritten letters. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 2921–2926.