[1]Beibei Hou
Zongxi Song[2*]
Baopeng Li[3]

Improved Adam: Incorporating Unified Conformable Fractional Derivative for fractional-order Momentum

Abstract: - Deep neural networks (DNNs) are closely tied to the training algorithm, and the performance of training process relies heavily on the choice of optimizer. The neural network optimizer is an algorithm used in deep learning (DL) to adjust the parameters of network to minimize the loss function. Currently, Adam is one of the most popular optimizers for its stability and efficiency. In recent years, there has been growing interest in exploring the use of fractional-order momentum, which offers greater flexibility compared to integer-order momentum and has shown promising performance in updating deep network parameters. One emerging method in fractional-order calculus is the Unified Conformable Fractional Derivative (UCFD), which has attracted extensive research attention. Motivated by this, this paper introduces an enhanced Adam optimizer incorporating unified conformable fractional-order momentum, referred to as the UCAdam. This method is trained and compared with the integer-order Adam using popular models and datasets for image classification tasks. The experiments indicate that the proposed optimizer achieves effective convergence and outperforms Adam in terms of performance.

Keywords: Fractional Derivative, Unified Conformable Fractional Derivative, Adam, Momentum

Chapter1 Introduction

As a pivotal subfield of artificial intelligence (AI), DL has made significant advancements and achieved remarkable success in various domains, including image recognition [1], large language models [2], and medical diagnosis [3]. etc. As core component of DNNs, optimizers are designed to tackle the challenge of finding the optimal set of parameters that minimize the difference between the predicted output of the network and the ground truth. This process, often referred to as optimization or training, involves iteratively updating the network's parameters based on the computed gradients of the loss function. The primary goals of optimizers are twofold: convergence and efficiency. Convergence ensures that the optimizer reaches a minimum of the loss function. Efficiency refers to the ability of the optimizer to achieve this convergence in a timely manner.

To train DNNs more effectively and efficiently, various optimizers have been proposed and extensively utilized. Stochastic gradient descent (SGD) [4] updates parameters by randomly selecting the gradient of one sample at each step, resulting in lower computational cost and the potential for finding better global optima due to its stochastic nature. However, the randomness of SGD can introduce training oscillations. To mitigate this issue, techniques such as momentum and adaptive learning rates have been introduced. Stochastic Gradient Descent with Momentum (SGDM) [5] introduces momentum in SGD to minimize oscillations. Its basic idea is to consider not only the gradient of the current step during parameter updates but also the accumulated gradient from previous steps, known as the momentum. The introduction of the momentum can be seen as adding inertia to the parameter updates, reducing fluctuations in the update direction of the parameters. However, a constant learning rate of SGDM may result in slow convergence or difficulties in convergence. On the other hand, Adagrad [6], Adadelta [7] and RMSprop [8] independently adjust the learning rate for each parameter. By combining momentum and adaptive learning rate techniques, Adam [9] addresses the limitations of other optimizers and offers several advantages. The momentum component in Adam helps accelerate the optimization process by introducing a moving average of past gradients which helps dampen oscillations and provides stability during training. Additionally, Adam incorporates adaptive learning rate mechanisms, leading to faster convergence and improved optimization performance. Adam is one of the most popular optimizers in DNNs due to its excellent performance

[1]Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, Shanxi, 710100, China; University of Chinese Academy of Sciences, Beijing, 101408, China

[2]Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, Shanxi, 710100, China;

[3]Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, Shanxi, 710100, China;

Corresponding Author: Zongxi Song, songxi@opt.ac.cn

and adaptability. However, the performance of Adam is contingent on the configuration of its hyper-parameter β_1 and β_2.

Fractional derivatives introduce a revolutionary concept in calculus that expands our understanding of derivatives. Since L'Hospital raised the question in 1695, if n is a fraction, then what is $\frac{d^n f}{dt^n}$, definitions of fractional differentials have emerged endlessly, such as Riemann–Liouville definition, Caputo definition [10-11], Hausdorff derivative [12] and Borges derivative [13] .etc. However, these definitions are complicated and cannot fully satisfy the properties of integer-order differentials. In 2014, Khalil et al. [14] proposed a new definition of fractional derivative and called it Conformable Fractional Derivative (CFD). The development of fractional calculus based on the CFD has provided a solid mathematical foundation for the application of fractional derivatives. This new definition of fractional derivatives possesses favorable mathematical properties and is relatively simple to work with. In subsequent research [15], scholars have further advanced the basic theory of fractional calculus, building upon the CFD. These advancements have enriched the understanding and application of fractional calculus in various fields, including deep learning optimization. Based on this, Wu et al. [16] extended the definition of the CFD to introduced the UCFD.

Unlike traditional integer-order derivatives, fractional-order derivatives can describe the changing characteristics of a function across a wider range. While integer-order derivatives primarily focus on the local rate of change of a function at a specific point, fractional-order derivatives offer a more detailed and comprehensive approach to measuring complex changes in functions at different scales. This novel derivative concept brings a fresh perspective to the design of DNNs [17-19], with fractional-order differences of momentum being one of the prevailing attempts in this direction. Peng et al. [20] presented the convergence proof of the backpropagation algorithm with momentum term. Yu Z. L. et al. [21] incorporated fractional-order differences of momentum and gradient into SGDM. Their experimental results indicate that the proposed optimizer surpasses other existing optimizers in terms of performance. Jing [22] respectively employed Hausdorff derivative and Borges derivative to calculate fractional-order momentum to enhance the RMSprop, resulting in a notable acceleration in the convergence speed of the algorithm.

Algorithm 1: UCAdam, our proposed extension of the Adam algorithm. See section 3 for details. g_k^2 indicates the elementwise square $g_k \square g_k$. Default settings are as same as Adam: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-6}$. All operation on vectors are element-wise. With β_1^k and β_2^k we denote β_1 and β_2 to the power k.

Require: η: Stepsize

Require: $\beta_1, \beta_1 \in [0,1)$: Component of decay rates for the moment estimates

Require: $f(\omega)$: Stochastic objective function with parameters ω

Require: ω_0: Initial parameter vector

$m_0 \leftarrow 0$: Initialize 1st moment vector

$v_0 \leftarrow 0$: Initialize 2nd moment vector

$k \leftarrow 0$: Initialize timestep

While ω_k not converged **do**

$k \leftarrow k+1$

$g_k = \nabla_\omega f_k(\omega_{k-1})$ (Get gradients w.r.t. stochastic objective at timestep *k*)

$m_k \leftarrow \left(1+\left(\beta_1 - 1\right)/k^{1-\alpha}\right) \cdot m_{k-1} + \left(\left(1-\beta_1\right)/k^{1-\alpha}\right) \cdot g_k$ (Update biased 1st moment)

$v_k \leftarrow \left(1+\left(\beta_2 - 1\right)/k^{1-\alpha}\right) \cdot v_{k-1} + \left(\left(1-\beta_2\right)/k^{1-\alpha}\right) \cdot g_k^2$ (Update biased 2nd moment)

$\hat{m}(k) \leftarrow m(k) \big/ \left(1-\beta_1^k\right)$ (Compute bias-corrected first moment estimate)

$\hat{v}(k) \leftarrow v(k) \big/ \left(1-\beta_2^k\right)$ (Compute bias-corrected second raw moment estimate)

$\omega(k) \leftarrow \omega(k-1) - \eta \cdot \hat{m}(k) \big/ \left(\sqrt{\hat{v}(k)} + \varepsilon\right)$ (Update parameters)

end while

return ω_k (Resulting parameters)

The incorporation of fractional-order momentum in optimizers allows for a more precise simulation and control of the parameter update process. This inclusion has the potential to enhance optimization and convergence properties, resulting in improved learning capabilities and generalization performance of the models. Considering the exceptional properties of the UCFD, this paper integrates fractional-order momentum computed by UCFD into the Adam optimization algorithm to improve the efficiency of the training process. This method is named unified conformable fractional-order Adam, abbreviated as UCAdam. The main innovations and contributions of this paper are as follows:

1. **Novel Optimization Algorithm:** Introducing UCAdam as a novel Adam optimization algorithm, where momentum is calculated using UCFD, providing a unique approach to training neural networks.
2. **Exploration of Fractional-Order Impact:** Investigating the effects of various fractional-order values on network training, shedding light on the influence of fractional derivatives on optimization dynamics and convergence behavior.
3. **Enhanced Training Performance:** Demonstrating that UCAdam surpasses the conventional integer-order Adam algorithm in terms of image classification accuracy and training speed, showcasing the efficacy of fractional-order momentum in optimization tasks.

The organization of this paper is as follows: Chapter 1 introduces the background of optimizers and fractional-order development. Chapter 2 presents the relevant fundamental knowledge. Chapter 3 derives the Adam optimization algorithm based on unified conformable fractional-order momentum. Chapter 4 describes the experimental design and the experimental results. Finally, Chapter 5 provides the conclusion.

Chapter2 related work

2.1 Adam algorithm

The Adam optimizer is a popular adaptive learning rate optimization algorithm, which combines the advantages of several successful optimization algorithms, particularly in handling non-convex optimization problems. Adam incorporates momentum to calculate both the first moment (mean) and second moment (variance) of the gradient simultaneously and utilizes them to update parameters. The state transition equation of Adam are as follows [9]:

$$\begin{aligned} m_k &= \beta_1 m_{k-1} + \left(1-\beta_1\right) g_k \\ v_k &= \beta_2 v_{k-1} + \left(1-\beta_2\right)\left(g_k\right)^2 \end{aligned} \quad (1)$$

In which, g_k is the current gradient of the model parameters. We collectively refer m_k and v_k as the momentum at step *k*. The 1st moment m_k is composed of the momentum m_{k-1} from the previous step and the current gradient, while the 2nd moment v_k is composed of the squared gradients. β_1 and β_2 are non-negative super parameters typically set to 0.9 and 0.999, respectively, with v_k moving the variance estimate slower than m_k moves the gradient estimate. If m_0 and v_0 is initialized as 0, it would result in a large initial bias ($0.1g_1$ and $0.001g_1^2$). To balance this bias, the normalized state variables are computed:

$$\hat{m}_k = \frac{m_k}{1-\beta_1^k}, \quad \hat{v}_k = \frac{v_k}{1-\beta_2^k} \quad (2)$$

In the above equation, $\hat{m}_k$ and $\hat{v}_k$ represent the bias-corrected estimates of m_k and v_k, respectively. The exponent *k* denotes the current step. The normalization helps in mitigating the initial bias and stabilize the Adam optimization algorithm.

According to the normalized state estimates, the parameter update equation of Adam is as follows:

$$\omega_k = \omega_{k-1} - \eta \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \varepsilon} \quad (3)$$

In the above Equation, ω_k and ω_{k-1} represent the parameters at step *k* and *k-1*, respectively. η is the learning rate, $\varepsilon = 10^{-6}$ is the scaling parameter to prevent the denominator from becoming zero.

2.2 Unified Conformable Fractional Derivative

The CFD is a newly emerging definition of fractional calculus. It has several notable features and advantages compared to traditional definitions of fractional derivatives. One of its main strengths is its relative simplicity and ease of understanding and application. The α order ($\alpha \in (n, n+1]$) CFD is defined as[23]:

$$\begin{aligned} \nabla^{\alpha} f(x) &= x^{1-\alpha}\left(f(x) - f(x-1)\right), \alpha \in (0,1] \\ \nabla^{\alpha} f(x) &= x^{[\alpha]-\alpha} \nabla^{n+1} f(x), \alpha \in (n, n+1] \end{aligned} \quad (4)$$

for all $n \in$ □. In which, $\nabla^{n+1} f(x)$ represents the (n+1)-order difference of $f(x)$, and [*] represents the ceiling function, denoted as $[\alpha] = n+1$.

Wu et al. [16] extended the definition of the CFD, representing it in a form accumulated from function values, referred to as UCFD:

$$\nabla^{\alpha} f(x) = x^{[\alpha]-\alpha} \sum_{i=x-[\alpha]}^{x} \frac{(-1)^{x-i}[\alpha]!}{([\alpha]-x+i)!(x-i)!} f(i), \alpha > 0 \quad (5)$$

Eq. (5) is the accumulation form of Eq. (4). Taking the examples of $\alpha \in (0, 2]$, the corresponding fractional derivatives are:

$$\begin{aligned} \nabla^{\alpha} f(x) &= x^{1-\alpha}\left(f(x) - f(x-1)\right), \alpha \in (0,1] \\ \nabla^{\alpha} f(x) &= x^{2-\alpha}\left(f(x) - 2f(x-1) + f(x-2)\right), \alpha \in (1,2] \end{aligned}$$

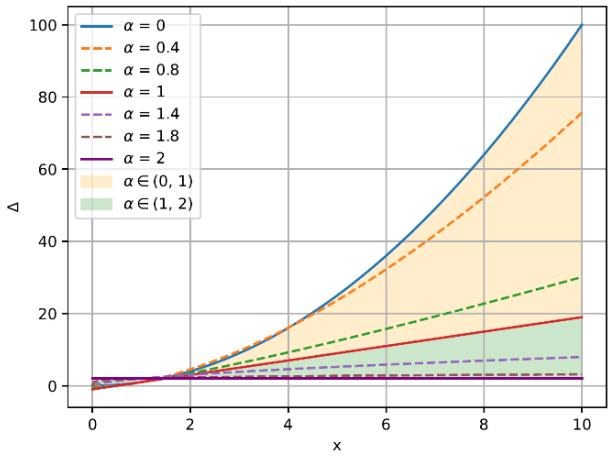

Figure 1 A comparison of UCFD in various fractional-order values and integer-order derivative and the original function.

By a simple function $y = x^2$, Fig.1 sequentially displays the original function (α=0), *0.4*-order differentiation, *0.8*-order differentiation, first-order differentiation, *1.4*-order differentiation, *1.8*-order differentiation, and second-order differentiation. It intuitively suggests that the UCFD facilitates a more extensive range of information representation. Regions that cannot be expressed by integer-order differentiation, exemplified by the yellow and green intervals in Fig. 1, can be effectively represented by adjusting fractional-order values. Therefore, introducing fractional-order difference into optimizers of DNNs helps to provide a more diverse update direction of parameters.

Chapter 3 Improved Algorithm Modeling

Adam updates the momentum at step *k* using the momentum from step (*k*-1). Referring to Eq. (1), the update value is:

$$\begin{aligned}\Delta m &= m_k - m_{k-1} = \left(\beta_1 - 1\right) m_{k-1} + \left(1-\beta_1\right) g_k \\ \Delta v &= v_k - v_{k-1} = \left(\beta_2 - 1\right) v_{k-1} + \left(1-\beta_2\right) {g_k}^2\end{aligned} \tag{6}$$

In this paper, by employing the fractional-order momentum computed by UCFD as the update value, we have:

$$\begin{aligned}\Delta m &= \nabla^{\alpha} m_k \\ \Delta v &= \nabla^{\alpha} v_k\end{aligned} \tag{7}$$

Then we derive:

$$\begin{aligned}\left(\beta_1 - 1\right) m_{k-1} + \left(1-\beta_1\right) g_k &= k^{[\alpha]-\alpha} \sum_{i=k-[\alpha]}^{k} \frac{\left(-1\right)^{k-i} [\alpha]!}{\left([\alpha]-k+i\right)!\left(k-i\right)!} m_i \\ \left(\beta_2 - 1\right) v_{k-1} + \left(1-\beta_2\right) {g_k}^2 &= k^{[\alpha]-\alpha} \sum_{i=k-[\alpha]}^{k} \frac{\left(-1\right)^{k-i} [\alpha]!}{\left([\alpha]-k+i\right)!\left(k-i\right)!} v_i\end{aligned} \quad , \quad \alpha \in \left(n, n+1\right] \tag{8}$$

To simplify the expression, consider the cases where $n=1$ and $n=0$ separately.

***n* = 1** When $n=1$, $\alpha \in (1,2]$, organizing Eq. (8) we have:

$$\begin{aligned}m_k &= \left(2 + \frac{\beta_1 - 1}{k^{2-\alpha}}\right) m_{k-1} - m_{k-2} + \frac{1-\beta_1}{k^{2-\alpha}} g_k \\ v_k &= \left(2 + \frac{\beta_2 - 1}{k^{2-\alpha}}\right) v_{k-1} - v_{k-2} + \frac{1-\beta_2}{k^{2-\alpha}} {g_k}^2\end{aligned} \tag{9}$$

Within the UCAdam framework, with a fractional-order range of (1, 2], the momentum update incorporates insights from the preceding two steps, offering a more comprehensive dataset information and theoretically resulting in more smooth parameter updates. Nonetheless, this strategy demands more memory consumption. Hence, in this study, we leverage fractional-order values between 0 and 1 to enhance the Adam algorithm.

***n* = 0** When $n=0$, $\alpha \in (0,1]$, organizing Eq. (8) we have the state transition equation of UCAdam:

$$\begin{aligned}m_k^{uc} &= \left(1 + \frac{\beta_1 - 1}{k^{1-\alpha}}\right) m_{k-1}^{uc} + \frac{1-\beta_1}{k^{1-\alpha}} g_k \\ v_k^{uc} &= \left(1 + \frac{\beta_2 - 1}{k^{1-\alpha}}\right) v_{k-1}^{uc} + \frac{1-\beta_2}{k^{1-\alpha}} {g_k}^2\end{aligned} \tag{10}$$

Denote $\gamma_{\beta_*,k}=1+\frac{\beta_*-1}{k^{1-\alpha}},\lambda_{\beta_*,k}=\frac{1-\beta_*}{k^{1-\alpha}}$, where $\beta_*\in[\beta_1,\beta_2]$, $\gamma_{\beta_*,k}$ represents the influence of the previous step's momentum, while $\lambda_{\beta_*,k}$ reflects the weight of the current gradient and can be seen as the overall learning rate of momentum. It is apparent that $\gamma_{\beta_*,k}=1-\lambda_{\beta_*,k}$, which means that the momentum is the weighted means of previous momentum and gradient of parameters. It is evident that when α = 1, the UCAdam reduces to the Adam, strongly affirming the coherence of the proposed UCAdam. This observation further exemplifies that the proposed UCAdam extends from Adam, enhancing the refinement and adjustability of the CNN parameter update direction through the introduction of the hyperparameter 'α'.

With $\beta_1=0.9$, select $\alpha=0.4,0.6,0.8,0.9,1$, record the change of $\gamma_{\beta_1,k}$ and $\lambda_{\beta_1,k}$ with the number of iterations k respectively, and the curves are as shown in Fig.2. It can be observed that when $\alpha\in(0,1]$, the momentum weight $\gamma_{\beta_1,k}$ increases with the number of iterations k, at mean time the unified learning rate $\lambda_{\beta_1,k}$ behaves in the opposite way. Within the first 10 iterations, $\gamma_{\beta_1,k}$ and $\lambda_{\beta_1,k}$ change rapidly, but after around 10 iterations, they stabilize. The introduction of fractional-order momentum directly transforms Adam's fixed hyper-parameters to adaptive ones. In scenarios where optimization stalls at saddle points with tiny gradients, momentum becomes pivotal in steering the process out of these points. UCAdam's dynamic weighting mechanism between momentum and gradient proves beneficial in navigating such situations, because as the number of iteration increases, momentum's impact grows and the gradient's influence diminishes. Moreover, the evolving weights offer momentum a broader range of update patterns through various fractional-order values, potentially resulting in improved training performance.

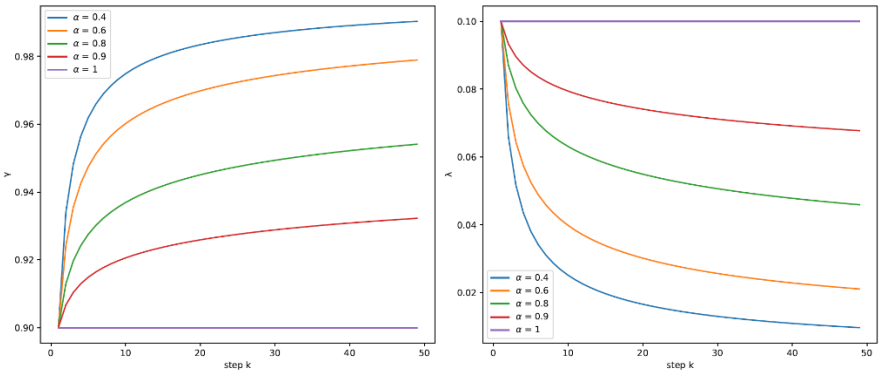

Figure 2 Change of the momentum weight $\gamma_{\beta_1,k}$ and the unified learning rate $\lambda_{\beta_1,k}$ with the number of iterations k when $\alpha\in(0,1]$.

The update values for CNN parameters in UCAdam comprehensively integrate both the first and second moments. With the same bias correction as Adam, this update in UCAdam is executed according to the following equation:

$$\hat{m}_k^{uc}=\frac{m_k^{uc}}{1-\beta_1^k}$$

$$\hat{v}_k^{uc}=\frac{v_k^{uc}}{1-\beta_2^k} \qquad (11)$$

$$\omega_k=\omega_{k-1}-\eta\frac{\hat{m}_k^{uc}}{\sqrt{\hat{v}_k^{uc}}+\varepsilon}$$

This paper introduces a novel optimizer as an extension of Adam, where fractional-order momentum, calculated using UCFD, substitutes integer-order momentum. From the aforementioned analysis, it can be inferred that the proposed UCAdam offers a broader spectrum of momentum update patterns, granting a more extensive selection

and finer tuning for parameter updates within deep neural networks. Refer to Algorithm 1 for the pseudo-code of our proposed UCAdam algorithm.

Chapter4 Experiments and result analysis

4.1 Experiments

In this paper, UCAdam is applied to LeNet-5 network and AlexNet for image classification task respectively. The datasets used are MNIST and cifar10.

Model Architectures

LeNet-5, a pioneering CNN architecture, was originally proposed by Yann LeCun in 1998 [24] for handwritten digit recognition. It comprises an Input layer, Convolutional Layer, Subsampling or Pooling Layer, Fully Connected Layer, and Output layer.

AlexNet, a seminal CNN in deep learning, was introduced by Alex Krizhevsky in collaboration with the University of Toronto [25]. It features 8 layers, including 5 convolutional layers, 3 fully connected layers, Local Response Normalization (LRN), and pooling layers. The architecture employs the ReLU activation function and integrates Dropout in fully connected layers to enhance generalization. Furthermore, AlexNet enhances feature learning through response normalization within localized regions.

Datesets

The **MNIST** dataset consists of 70,000 grayscale images of handwritten numerals, each sized at 28x28 pixels, representing digits 0 to 9 with equal sample distribution [26].

The **CIFAR10** dataset offers 60,000 colorful images, each 32x32 pixels in size, across 10 classes with 6,000 images per class [27]. It is divided into a training set of 50,000 images and a testing set of 10,000 images, covering categories like airplanes, cars, birds, cats, deer, dogs, frogs, horses, boats, and trucks.

Table 1 Experiment Setup

	LeNet-MNIST	**AlexNet-Cifar10**
Model	LeNet-5	AlexNet
Dataset	MNIST	Cifar10
Stepsize	0.001	0.0001
Epochs	12	40
Tasks	image classification	image classification

Implementation Details

Two sets of experiments are conducted, with the configurations detailed in Tab. 1. Both of them executed multiple training runs using UCAdam, a new optimizer derived from Adam with fractional-order momentum computed using UCFD. Each fractional-order value underwent independent training five times to mitigate the impact of stochasticity in gradient calculation. The experiments systematically evaluated the performance of UCAdam across various fractional-order values in contrast to the conventional integer-order Adam algorithm. These comparisons were conducted on an NVIDIA GeForce RTX 3060 system equipped with CUDA version 12.0, providing insights into the efficacy of fractional-order momentum in optimization algorithms and influence of various fractional-order values.

4.2 Experiment results

The experimental results are depicted in Fig. 3. Upon analysis, it is evident that:

Firstly, the experimental results validate the convergence of the proposed UCAdam. This convergence implies that with this optimization strategy, the loss function can converge towards its minimum value.

Secondly, in both sets of experiments, the training performance notably subpar when α=0.4. This decline can be linked to a significant imbalance in the weighting mechanism between momentum and gradient at α=0.4, as depicted in Figure 2. At this threshold, momentum gradients dwindle, halting updates and causing momentum to

stagnate. Consequently, network parameters are updated with fixed values, losing their learning potential, resulting in decreased performance.

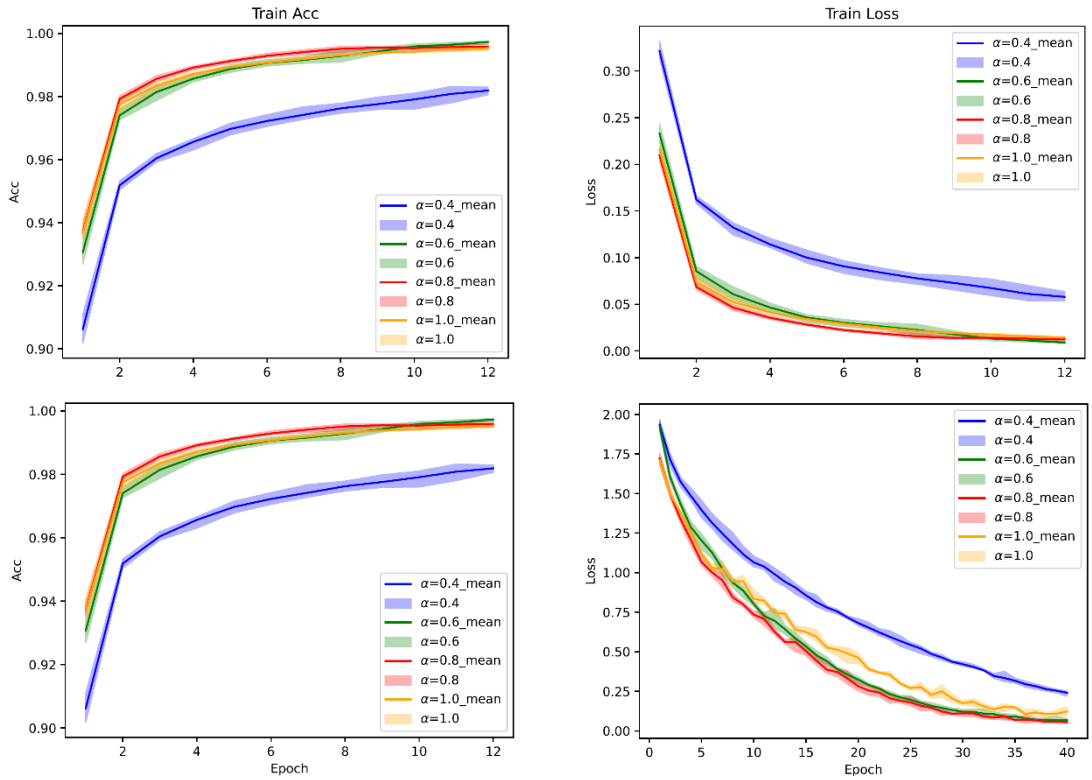

Figure 3 Comparison of training performance for various fractional-order values in UCAdam. The 1st row showcases the performance of LeNet-5 on MNIST; the 2nd row showcases the performance of AlexNet on Cifar10. Note: when α=1.0, UCAdam reduces to Adam.

Finally, UCAdam with α=0.8 exhibits superior performance with LeNet-5 on MNIST, while both UCAdam with α=0.8 and α=0.6 significantly outperform Adam in terms of both convergence speed and accuracy on AlexNet-Cifar10. Particularly notable is that with α=0.8, after 5 epochs, UCAdam's training efficiency begins to outstrip Adam and eventually converges to a higher level of accuracy. In experiment group with AlexNet on Cifar10, upon comprehensive comparison of the best results from 5 training runs for each fractional order value, as presented in Tab. 2, it is evident that the training accuracy of UCAdam optimizer with alpha=0.8 surpasses that of Adam. This indicates that our proposed UCAdam effectively enhances the performance of Adam.

Table 2 Comparison of training results on AlexNet-Cifar10

Fractional-order value	**Loss**	**Accuracy**
0.4	0.252	0.910
0.6	0.077	0.974
0.8	**0.070**	**0.978**
1.0	0.154	0.946

In conclusion, UCAdam exhibits superior performance when the fractional-order value is not less than 0.6. This is because an excessively small fractional-order value can result in stagnation in momentum updates. When the fractional order surpasses 0.6, the selection of the fractional order value should be tailored to the characteristics of the network architecture and dataset. By introducing a hyper-parameter (fractional order value α), UCAdam enables dynamic changes in the weighting of momentum and gradient during momentum updates, thus requiring the identification of an appropriate α, corresponding to a particular pattern of momentum and gradient weighting changes. Such a pattern will aid in achieving faster and more stable convergence of the loss function towards its minimum value. This pattern is evidently beyond the capabilities of the basic Adam algorithm. UCAdam, as proposed by us, presents a potential avenue for discovering and leveraging this pattern.

Chapter5 Conclusion

This paper introduces an effective optimizer, UCAdam, by incorporating fractional-order momentum computed by UCFD into Adam. Theoretical analysis demonstrates that UCAdam complements and extends the capabilities of the Adam optimizer. When the fractional-order value is appropriately chosen, UCAdam provides a more refined and adaptive solution for updating momentum within Adam, offering a broader range of options. Experimental results also confirm the potential of UCAdam to achieve higher convergence efficiency and improved accuracy compared to Adam. However, UCAdam introduces a new hyperparameter, α, which, when too small, can lead to momentum updates stagnating. Further research and discussion are necessary to explore how this additional parameter affects the performance of UCAdam.

Reference：

[1] Matei A , Glavan A , Talavera E .Deep learning for scene recognition from visual data: a survey[J]. 2020.DOI:10.1007/978-3-030-61705-9_64.

[2] Brown T B , Mann B , Ryder N ,et al. Language Models are Few-Shot Learners[J]. 2020.DOI:10.48550/arXiv.2005.14165.

[3] Qiao-Li Z , Di Z , Xue-Bin C .Review for Deep Learning Based on Medical Imaging Diagnosis[J].Computer Science, 2017.

[4] Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010. Springer, pp 177–186

[5] Qian N (1999) On the momentum term in gradient descent learning algorithms. Neural Netw 12:145–151

[6] Duchi,John,Hazan, et.al. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.[J].Journal of Machine Learning Research, 2011.

[7] Zeiler MD (2012) Adadelta: an adaptive learning rate method. arXiv:1212.5701

[8] Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning, 4(2), 26-31.

[9] Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980

[10] K. Nishimoto, Fractional Calculus: Integrations and Differentiations of Arbitrary Order, University of New Haven Press, New Haven, Conn, USA, 1989.

[11] I. Podlubny, Fractional differential Eq.s: an introduction to fractional derivatives, fractional differential Eq.s, to methods of their solution and some of their applications, vol. 198, Academic press, 1998.

[12] Chen W, Wang F J,Zheng B,et al. Non-Euclidean distance fundamental solution of Hausdorff derivative partial differential Eq.s [J]. Engineering Analysis With Boundary Elements ,2017,84:213 -219.

[13] Chen W. Time-space fabric underlying anomalous diffusion [J]. Chaos, Solitons & Fractals,2006 ,28 (4):923-929.

[14] Khalil R , Horani M A , Yousef A ,et al. A new definition of fractional derivative[J].Journal of Computational and Applied Mathematics, 2014(1).DOI:10.1016/j.cam.2014.01.002.

[15] Abdeljawad T .On conformable fractional calculus[J].Journal of Computational and Applied Mathematics, 2015, 279(Null).DOI:10.1016/j.cam.2014.10.016.

[16] Wu, Wenqing, et al. "A novel conformable fractional non-homogeneous grey model for forecasting carbon dioxide emissions of BRICS countries." Science of the Total Environment 707 (2020): 135447.

[17] Bao C , Pu Y , Zhang Y .Fractional-Order Deep Backpropagation Neural Network[J].Computational intelligence and neuroscience, 2018, 2018:7361628.DOI:10.1155/2018/7361628.

[18] Fractional ordering of activation functions for neural networks: A case study on Texas wind turbine

[19] Hybrid LSTM-Based Fractional-Order Neural Network for Jeju Island'sWind Farm Power Forecasting

[20] 彭先伦. "带动量项的梯度下降算法的收敛性." Journal of East China University of Science & Technology 47.6 (2021).

[21] Yu Z L , Sun G , Lv J .A fractional-order momentum optimization approach of deep neural networks[J].Neural Computing and Applications, 2022, 34(9):7091-7111.DOI:10.1007/s00521-021-06765-2.

[22] 基于分数阶动量法的卷积神经网络优化及其在图像识别中的应用；辽宁大学，剪静

[23] Ma, Xin, et al. "The conformable fractional grey system model." ISA transactions 96 (2020): 255-271.

[24] Lecun Y , Bottou L .Gradient-based learning applied to document recognition [J].Proceedings of the IEEE, 1998, 86(11):2278-2324.DOI:10.1109/5.726791.

[25] Krizhevsky A , Sutskever I , Hinton G .ImageNet Classification with Deep Convolutional Neural Networks[J].Advances in neural information processing systems, 2012, 25(2).DOI:10.1145/3065386.

[26] MNIST: http://yann.lecun.com/exdb/mnist/

[27] Cifar10: http://www.cs.toronto.edu/~kriz/cifar.html