axioms

MDPI

Article

Training Neural Networks by Time-Fractional Gradient Descent

Jingyi Xie and Sirui Li *

School of Mathematics and Statistics, Guizhou University, Guiyang 550025, China
* Correspondence: srli@gzu.edu.cn

Abstract: Motivated by the weighted averaging method for training neural networks, we study the time-fractional gradient descent (TFGD) method based on the time-fractional gradient flow and explore the influence of memory dependence on neural network training. The TFGD algorithm in this paper is studied via theoretical derivations and neural network training experiments. Compared with the common gradient descent (GD) algorithm, the optimization effect of the time-fractional gradient descent algorithm is significant when the value of fractional α is close to 1, under the condition of appropriate learning rate η. The comparison is extended to experiments on the MNIST dataset with various learning rates. It is verified that the TFGD has potential advantages when the fractional α nears 0.95∼0.99. This suggests that the memory dependence can improve training performance of neural networks.

Keywords: time-fractional gradient descent; training neural networks; weighted averaging; memory dependence

MSC: 26A33; 35K99; 35E15

Citation: Xie, J.; Li, S. Training Neural Networks by Time-Fractional Gradient Descent. *Axioms* **2022**, *11*, 507. https://doi.org/10.3390/axioms11100507

Academic Editor: Oscar Humberto Montiel Ross

Received: 1 September 2022
Accepted: 22 September 2022
Published: 26 September 2022

1. Introduction

Many problems arising in machine learning, and intelligent systems are usually reduced to optimization problems. Neural network training is essentially a process that minimizes model errors. The GD method, in particular the stochastic GD method has been extensively applied to solve such problems from different perspectives [1].

To improve the optimization error and the generalization error in the training of neural networks, the parameter-averaging technique that combines iterative solutions has been employed. Most works adopting simple averaging based on all iterative solutions generally cannot obtain satisfactory performance regardless of their advantages. Bottou [2] and Hardt et al. [3] uses the simple output of the last solution, which leads to faster convergence but has less stability in the optimization error. Some works [4,5] also analyze the stochastic GD method by the uniform averaging. To improve the convergence rate of strongly convex function optimization, the non-uniform averaging method has been proposed in [6,7]. However, the averaging method that combines all iterative solutions into a single solution remains to be discussed. Guo et al. [8] provide a beneficial attempt for this problem. For the non-strongly convex objectives, they analyze the optimization error and generalization error by synthesizing a polynomial increasing weighted average scheme. Again, based on a new primal averaging, Tao et al. [9] attain the optimal individual convergence using a simple modified gradient evaluation step. By a simple averaging of multiple points along the trajectory of stochastic GD, Izmailov et al. [10] also obtain a better generalization than conventional training and show that the stochastic weight averaging procedure finds much flatter solutions than stochastic GD.

The core formula in the GD algorithm is to calculate the derivative of the loss function. Fractional calculus is applied to the GD algorithm in the training of neural networks, which is called the fractional GD method since it gives better optimization performance, especially the stability of algorithm. Khan et al. [11] present a fractional gradient descent-based

learning algorithm for radial-basis-function neural networks, based on Riemann–Liouville derivative and a convex combination of improved fractional GD. Bao et al. [12] provide a Caputo fractional-order deep back-propagation neural network model armed with L^2 regularization. A fractional gradient descent method for the BP neural networks is proposed in [13]. In particular, the Caputo derivative is used to calculate the fractional-order gradient of the error expressed by the traditional quadratic energy function. An adaptive fractional-order BP neural network for handwritten digit recognition problems is presented in [14], combining a competitive evolutionary algorithm. The general convergence problem of the fractional-order gradient method has been tentatively investigated in [15].

There are a great deal of works on the fractional GD method, and we will not list them here.

We now turn to the time-fraction gradient flow. Two classical examples are the Allen–Cahn equation and the Cahn–Hilliard equation. The time-fractional Allen–Cahn equation has been investigated both theoretically and numerically [16–20]. Tang et al. [20] establish the energy dissipation law of the fractional time-field equation. Specifically, they prove that the fractional time-field model admits an integral-type energy dissipation law at a continuous level. Moreover, the discrete version of the energy-dissipative law is also inherited. Liao et al. [17] design a variable-step Crank–Nicolson-type scheme for the time fractional Allen–Cahn equation, which is unconditionally stable. The proposed scheme can preserve both the energy stability and the maximum principle. Liu et al. [18] investigate the fractional Allen–Cahn and Cahn–Hilliard phase field models and give an effect finite difference and Fourier spectral schemes. On the other hand, for the complex time-fractional Schrödinger equation and the space–time fractional differential equation, novel precise solitary wave solutions are obtained by the modified simple equation scheme [21]. A number of solitary envelope solutions of the quadratic nonlinear Klein–Gordon equation also are constructed by the generalized Kudryashov and extended Sinh–Gordon expansion schemes [22].

In the training of neural networks, the parameter averaging technique can be regarded as memory-dependent, which just corresponds to an important feature of the time-fractional derivative. The weighted averaging cannot be easily used to analyze theoretically; however, the time-fractional derivative is well suited to this purpose. The main objective of this paper is to develop a TFGD algorithm based on the time-fractional gradient flow and theoretical derivations and then to explore the influence of memory dependence on training neural networks.

2. The Time-Fractional Allen–Cahn Equation

This section will be devoted to introducing the time-fractional Allen–Cahn equation and its basic properties. Consider the following time-fractional Allen–Cahn equation,

$$\partial_t^\alpha u = \varepsilon^2 \Delta u - f(u), \text{ for } x \in \Omega \text{ and } t > 0, \tag{1}$$

where the domain $\Omega = (0, L)^2 \subset \mathbb{R}^2$, and ε is an interface width parameter and u is a function of (x, t). The notation ∂_t^α represents the Caputo derivative of order α defined as

$$\partial_t^\alpha v = {}_0^C\mathcal{D}_t^\alpha \stackrel{\text{def}}{=} \mathcal{I}_t^{1-\alpha} v', \quad \alpha \in (0, 1), \tag{2}$$

where $\mathcal{I}_t^\alpha$ is the Riemann–Liouville fractional integration operator of order $\alpha > 0$,

$$(\mathcal{I}_t^\alpha v) \stackrel{\text{def}}{=} \int_0^t \omega_\alpha(t - s) v(s) ds, \quad \omega_\alpha(t) \stackrel{\text{def}}{=} \frac{t^{\alpha-1}}{\Gamma(\alpha)}.$$

The nonlinear function $f(u)$ is associated with the derivative of the bulk energy density function $F(u)$, which is usually chosen as

$$F(u) = \frac{1}{4}(1-u^2)^2.$$

The function $F(u)$ is of bistable type and admits two local minima.

When the fractional order parameter $\alpha = 1$, Equation (1) immediately becomes the classical Allen–Cahn equation,

$$\partial_t u = \varepsilon^2 \Delta u - f(u). \tag{3}$$

The Allen–Cahn Equation (3) can be viewed as an L^2 gradient flow,

$$\partial_t u = -\frac{\delta E}{\delta u}, \tag{4}$$

where $\frac{\delta E}{\delta u}$ represents the first-order variational derivative, and the corresponding free-energy functional $E[u]$ is defined as

$$E[u](t) = \int_\Omega \Big(\frac{\varepsilon^2}{2}|\nabla u|^2 + F(u)\Big)dx.$$

It is well-known that Equation (3) satisfies two important properties: one is the energy dissipation,

$$\frac{dE}{dt} = -\Big\|\frac{\delta E}{\delta u}\Big\|^2, \quad \text{or } E[u](t) \le E[u](s), \quad \forall t > s,$$

and the other is the maximum principle,

$$|u(x,t)| \le 1 \text{ if } |u(x,0)| \le 1.$$

These two properties play an important role in constructing stable numerical schemes.

In a similar way, the time-fractional Allen–Cahn Equation (1) can be also regarded as a fractional gradient flow,

$$\partial_t^\alpha u = -\frac{\delta E}{\delta u}. \tag{5}$$

It can be found in [20] that Equation (1) admits the maximum principle but only satisfies the special energy inequality,

$$E[u](t) \le E[u](0), \quad \forall t > 0.$$

TFGD Model and Numerical Schemes

Before giving numerical algorithms, we write down the TFGD model based on training neural networks.

In machine learning, the main task of learning a predictive model is typically formulated as the minimization problem of the empirical risk function,

$$\min \mathcal{J}(w), \quad \mathcal{J}(\omega) = \mathbb{E}_{x,y}\mathcal{L}(y, f(x;w)), \tag{6}$$

where w is a set of parameters in the neural networks and $\mathcal{L}(\cdot,\cdot)$ is the loss function for each sample, which quantifies the deviation between the predicted value f and the respective true one y. When the gradient descent is applied, the iteration of weights for (6) can be given by

$$w_{k+1} = w_k - \xi \nabla \mathcal{J}(w_k), \tag{7}$$

where ξ represents a positive stepsize. The continuous version is the gradient flow,

$$w_t = -\nabla \mathcal{J}(w).$$

In order to solve (7) associated with (6), existing works mostly consider the weighted averaging schemes over the training process instead of the newest weight, that is,

$$\widetilde{w}_k = \sum_{i=1}^{k} \gamma_i w_i. \tag{8}$$

The simplest case of (8) is the arithmetic mean, where γ_i are constants. In this paper, we will consider non-constant cases such as $\gamma_i \propto \beta^i$ for some $\beta \geq 1$ and propose a new type of weight that is connected to the fractional gradients.

Now, we consider the following model equation:

$$\partial_t^\alpha w(t) = -\nabla \mathcal{J}(w), \quad 0 < \alpha < 1, \tag{9}$$

$$w(0) = w_0, \tag{10}$$

where ∂_t^α denotes the Caputo fractional derivative operator defined by

$$\begin{aligned} \partial_t^\alpha w(t) =& \frac{1}{\Gamma(1-\alpha)} \int_0^t (t-s)^{-\alpha} \frac{d}{ds} w(s) ds \\ =& \frac{d}{dt} \int_0^t \frac{(t-s)^{-\alpha}}{\Gamma(1-\alpha)} (w(s) - w(0)) ds \\ =& P.V. \int_0^t \frac{(t-s)^{-\alpha-1}}{\Gamma(-\alpha)} (w(s) - w(0)) ds \end{aligned} \tag{11}$$

with $P.V.$ being the principal value.

Remark 1. *For the gradient flow $w_t = -\nabla \mathcal{J}(w)$, the loss is descent since $\partial_t \mathcal{J}(w) \leq 0$. For the case of the fractional flow, however, it can be seen from [20] that there is a regularized loss that is descending. In addition, when $\alpha \to 0$, the Equation (9) recovers the classical Allen–Cahn equation.*

We know that Equation (9) corresponds to a fractional gradient flow (5). Similar to the work [20], to maintain the energy dissipative law, we define the modified variational energy as

$$E_\alpha(w) = \mathcal{J}(w) + \frac{1}{2} \mathcal{I}_t^\alpha \|\nabla \mathcal{J}(w)\|^2 \stackrel{\text{def}}{=} \mathcal{J}(w) + \mathcal{R}(w), \tag{12}$$

where the non-negative term $\mathcal{R}(w)$ is a regularization term, which is given by

$$\mathcal{R}(w) = \frac{1}{2} \int_0^t \frac{1}{(t-s)^{1-\alpha} \Gamma(\alpha)} \|\nabla \mathcal{J}(w(s))\|^2 ds. \tag{13}$$

Then the modified fractional gradient flow of (12) is given by

$$\partial_t^\alpha w = -\frac{\delta E_\alpha}{\delta w}. \tag{14}$$

Following the work [20], we have the following dissipation properties.

Theorem 1. *The modified variational energy E_α is dissipative along the time-fractional gradient flow (14).*

Proof. To begin with, the time-fractional Allen–Cahn Equation (9) can be equivalently reformulated as

$$\partial_t w(t) = -{}^R\partial_t^{1-\alpha} \nabla \mathcal{J}(w), \tag{15}$$

where ${}^R\partial_t^\alpha = {}_0^{RL}\mathcal{D}_t^\alpha$ is the Riemann–Liouville derivative defined by

$${}^R\partial_t^\alpha v \overset{\text{def}}{=} \partial_t \mathcal{I}_t^{1-\alpha} v, \quad \alpha \in (0,1). \tag{16}$$

Along the time fractional gradient flow $\partial_t^\alpha w(t) = -\nabla \mathcal{J}(w)$, we have

$$\frac{d\mathcal{J}(u)}{dt} = \Big\langle \nabla \mathcal{J}(w), \partial_t w \Big\rangle = -\Big\langle \nabla \mathcal{J}(w), {}^R\partial_t^{1-\alpha} \nabla \mathcal{J}(w) \Big\rangle. \tag{17}$$

Using the inequality from [23],

$$v(t)\Big({}^R\partial_t^{1-\alpha} v\Big)(t) \geq \frac{1}{2}\Big({}^R\partial_t^{1-\alpha} v^2\Big)(t) + \frac{1}{2}\omega_\alpha(t) v^2(t), \quad \forall vs. \in C[0,T],$$

and letting $v(t) = \nabla \mathcal{J}(w)$, we derive from (15) and (17) that

$$\begin{aligned} \frac{d\mathcal{J}(w)}{dt} \leq & -\frac{1}{2}{}^R\partial_t^{1-\alpha} \|\nabla \mathcal{J}(w)\|^2 - \frac{1}{2}\omega_\alpha(t)\|\nabla \mathcal{J}(w)\|^2 \\ = & -\partial_t \mathcal{I}_t^\alpha \|\nabla \mathcal{J}(w)\|^2 - \frac{1}{2}\omega_\alpha(t)\|\nabla \mathcal{J}(w)\|^2. \end{aligned}$$

Therefore, we obtain

$$\frac{dE_\alpha}{dt} = \frac{d\mathcal{J}(w)}{dt} + \partial_t \mathcal{I}_t^\alpha \|\nabla \mathcal{J}(w)\|^2 \leq -\tfrac{1}{2}\omega_\alpha(t)\|\nabla \mathcal{J}(w)\|^2 \leq 0.$$

□

Remark 2. *Armed with (2) and (16), we see that the Caputo derivative ${}_0^C\mathcal{D}_t^\alpha$ and the Riemann–Liouville derivative ${}_0^{RL}\mathcal{D}_t^\alpha$ can be defined by the Riemann–Liouville fractional integration operator,*

$$\begin{aligned} {}_0^{RL}\mathcal{D}_t^\alpha v(t) = & \partial_t \mathcal{I}_t^{1-\alpha} v = \frac{1}{\Gamma(1-\alpha)} \frac{d}{dt} \int_0^t (t-s)^{-\alpha} v(s) ds, \\ {}_0^C\mathcal{D}_t^\alpha v(t) = & \mathcal{I}_t^{1-\alpha} v' = \frac{1}{\Gamma(1-\alpha)} \int_0^t (t-s)^{-\alpha} v'(s) ds. \end{aligned}$$

For an absolutely continuous (derivable) function $v : \mathbb{R}^+ \to \mathbb{R}$, the Caputo fractional derivative and the Riemann–Liouville fractional derivative can be connected with each other by the following relations:

$${}_0^C\mathcal{D}_t^\alpha v(t) = {}_0^{RL}\mathcal{D}_t^\alpha (v(t) - v(0)), \text{ or } {}_0^C\mathcal{D}_t^\alpha v(t) = {}_0^{RL}\mathcal{D}_t^\alpha v(t) - \frac{v(0)}{\Gamma(1-\alpha)} t^{-\alpha},$$

where $0 < \alpha < 1$. Compared with the Riemann–Liouville derivative, the Caputo derivative is more flexible for handling initial and boundary value problems.

The remaining part of this section is to construct the numerical scheme for the TFDG model by weighted averaging with the newest weight. The first-order accurate Grünwald–Letnikov formula for

$$P.V. \int_0^t \frac{(t-s)^{-\alpha-1}}{\Gamma(-\alpha)} w(s) ds$$

is given by

$$P.V.\int_0^{t_n}\frac{(t_n-s)^{-\alpha-1}}{\Gamma(-\alpha)}w(s)ds\approx\frac{1}{\eta}^{\alpha}\sum_{k=0}^{n}\phi_{n-k}^{(\alpha)}w(t_k), \tag{18}$$

where $t_k=k\eta$, η is time step size, and $\phi_n^{(\alpha)}=(-1)^n\binom{\alpha}{n}=\frac{\Gamma(n-\alpha)}{\Gamma(-\alpha)\Gamma(n+1)}$ satisfies

$$\phi^{(\alpha)}(z)=(1-z)^{\alpha}=\sum_{n=0}^{\infty}\omega_n^{(\alpha)}z^n,\quad |z|\le 1,$$

which can be calculated by the recurrence formula

$$\phi_n^{(\alpha)}=\frac{n-1-\alpha}{n}\phi_{n-1}^{(\alpha)},\quad \phi_0^{(\alpha)}=1. \tag{19}$$

Then, from (11) and (18), we obtain

$$\partial_t^{\alpha}w(t_n)\approx\frac{1}{\eta}^{\alpha}\sum_{k=0}^{n}\phi_{n-k}^{(\alpha)}(w(t_k)-w(0)). \tag{20}$$

This provides a training scheme

$$\frac{1}{\eta^{\alpha}}\sum_{i=0}^{k+1}\omega_{k+1-i}^{(\alpha)}(u_i-u_0)=-\nabla\mathcal{J}(w_n),$$

which is reformulated as

$$w_{k+1}=w_0-\eta^{\alpha}\Big(\nabla\mathcal{J}(w_k)+\sum_{i=0}^{k}\phi_{k+1-i}^{(\alpha)}(w_i-w_0)\Big). \tag{21}$$

It should be emphasized that scheme (21) is new and is a key step in the time-fractional gradient descent algorithm, while other steps are similar to those in usual gradient descent.

3. Numerical Simulation and Empirical Analysis

In this section, we will present numerical experiments of neural networks to verify our theoretical findings by combining the actual data and the classical MNIST data set. The experiment involves two important parameters: the learning rate (lr) denoted by η and the fractional-order parameter α. We will explore the influence of the two parameters on the TFGD algorithm and compare it with the general GD algorithm.

We now present the algorithm description for TFGD, which is a simple but effective modification for the general GD algorithm in the training neural networks. The TFGD procedure is summarized in Algorithm 1.

Algorithm 1 Time-fractional gradient descent (TFGD)

Input: fractional-order parameter $\alpha\in(0,1)$, weight w_0, LR bounds η_1,η_2, cycle length c (for constant learning rate $c=1$), number of iterations n
Ensure: weight w
  $w\leftarrow w_0$ {Initialize weights with w_0}
  for $k\leftarrow 1,2,\cdots,n$ **do**
    $\eta\leftarrow\eta(k)$ {Calculate LR for the iteration}
    Calculate the gradient $g_k\leftarrow\nabla\mathcal{J}_k(w)$
    $w\leftarrow w-\eta g_k$ {for GD update} $(*)$
    or execute
    $w\leftarrow w-\eta^{\alpha}\Big(g_k+\sum_{i=0}^{k}\phi_{k+1-i}^{(\alpha)}(w_i-w_0)\Big)$ {for TFGD update} $(**)$
    Store w as w_k {for TFGD}
  end for

Some explanations for the TFGD algorithm are indispensable. We linearly decrease the learning rate from η_1 to η_2 in each cycle. The formula for the learning rate at iteration k is given by

$$\eta(k) = (1 - t(k))\eta_1 + t(k)\eta_2, \quad t(k) = \frac{1}{c}\big(\mathrm{mod}(k-1, c) + 1\big),$$

where the base learning rates $\eta_1 \geq \eta_2$ and the cycle length c are hyper-parameters of the method. On the other hand, when choosing $(*)$ to update w, Algorithm 1 corresponds to the general GD algorithm. When $(**)$ is chosen to update w, Algorithm 1 becomes the TFGD algorithm. Again, when updating the step $(**)$, the calculation of $\phi^{(\alpha)}$ is executed by the recurrence formula (19).

3.1. Numerical Simulation

The experimental data are randomly set. The date x is randomly taken as 20,000 points on the interval of $(-1, 1)$, and the value y is obtained by the test function

$$f(x) = \sin(x) + \sin(\pi x) + \sin(ex).$$

The neural network model with two hidden layers is applied in this experiment. The input is one and the output is one, and two hidden layers contain 25 neurons.

We next choose the two parameters. The learning rate η are taken as two different value $[0.1, 0.01]$, and the fractional order parameter α are chosen as $[0.7, 0.8, 0.9, 0.95, 0.99]$ which belong to $(0.5, 1)$.

In the neural network training experiment based on the TFGD algorithm, each set of data on the basis of 500 iterations will be repeatedly run 20 times for training. Then, we take the average loss of 20 times of repeated experiments when the loss of each iteration is the lowest. We obtain the obvious loss effects of the neural network associated with different α, when the learning rate η is taken as 0.01 or 0.1, respectively. The experiment result is shown in Figure 1.

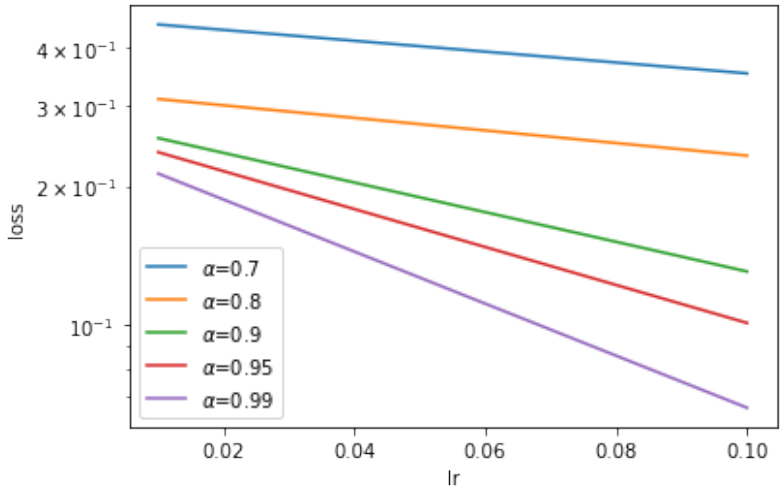

Figure 1. Effects of fractional order parameter α, where lr $= \eta$ denotes the learning rate.

Next, to make the effect of the neural network training experiment more clear and intuitive, the learning rate is fixed at 0.1 and 0.01. We compare the loss effects of the full batch gradient descent algorithm and the TFGD algorithm. The results are shown in Figure 2.

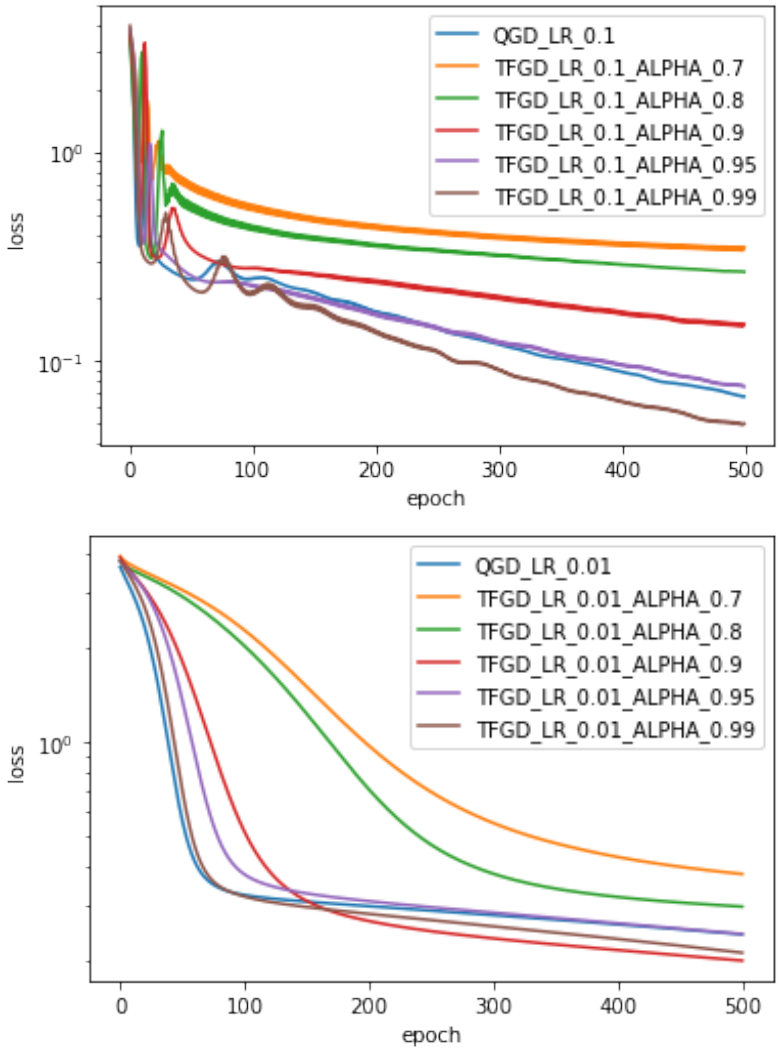

Figure 2. Comparison of loss effects for the learning rates $\eta = 0.1, 0.01$, and different fractional order parameter α, where QGD and TFGD stand for the general gradient descent and the time-fractional gradient descent, respectively.

From Figure 2, we can verify the following two facts:

- For different learning rates ($\eta = 0.01, 0.1$) and the fractional order parameter $\alpha = 0.99$, the effect of the neural network optimization based on the TFGD algorithm is significantly better than that of the general CD algorithm. Again, the fractional order parameter α is larger, and the optimization effect of the neural network is better.
- When $\eta = 0.1, \alpha = 0.99$ or $\eta = 0.01, \alpha = 0.9, 0.99$, the loss effect of the TFGD algorithm is better than that of the general GD algorithm. However, the loss effect of TFGD is the same as that of the general GD if $\eta = 0.01, \alpha = 0.95$.

In a word, under the condition of selecting the appropriate learning rate, if the value of α is larger, the loss effect of the TFGD is better. This verifies that the memory dependence has an impact on the training of neural networks.

3.2. Empirical Analysis

To further determine the optimization effect of the TFGD algorithm in the neural network and to verify the accuracy of the previous numerical experiments, I apply the TFDG algorithm to the neural network optimization of real data. In the empirical analysis, I mainly use the classical handwritten digital data set, i.e., the MINST data set. To improve

the efficiency of the training, the parameter α is taken as $[0.95, 0.99]$, and the different learning rates $\eta = 0.01, 0.1$ are also selected.

The input data of this experimental model are 784-dimensional (28×28), the output data are one-dimensional, and the cross entropy is used as the loss function. To improve the efficiency of neural network training, the first 1000 data of handwritten digital data sets are selected for the experiment. We run the TFGD algorithm with 100 iterations. Under different learning rates ($\eta = 0.01$ or $\eta = 0.1$), the optimization effect of the TFGD algorithm is better than that of the general GD algorithm. Again, for the TFGD algorithm, the loss effect with $\alpha = 0.99$ is better than that with $\alpha = 0.95$. The corresponding results are shown in Figure 3.

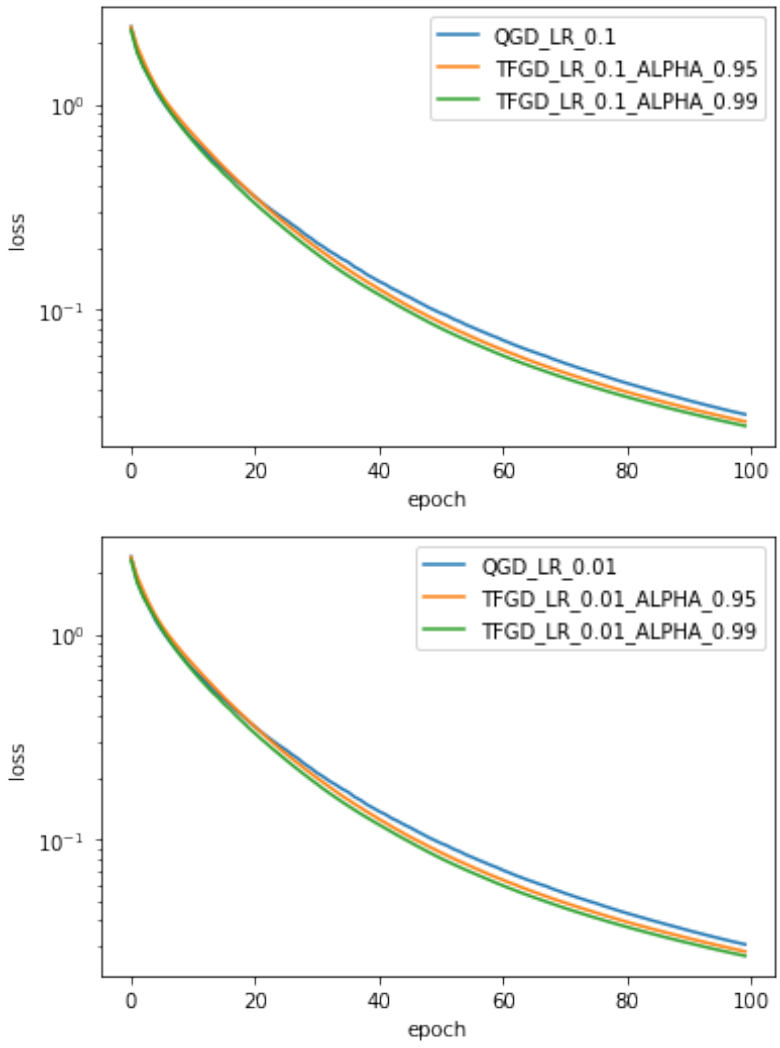

Figure 3. Comparison of loss effects for the learning rates $\eta = 0.1$ and $\eta = 0.01$, and the fractional order parameters $\alpha = 0.95, 0.99$, where QGD and TFGD stand for the general gradient descent and the time-fractional gradient descent, respectively.

In the above, armed with the classical handwritten digital data sets, the neural network training experiments based on the TFGD algorithm and the general GD algorithm are carried out, respectively. At different learning rates, we verify that the memory dependence affects neural network training.

4. Conclusions

This paper mainly studies the TFGD algorithm based on a weighted average to optimize the neural network training, which finally leads to the better generalization perfor-

mance and the lower loss. Specifically, the parameter averaging method in neural network training can be regarded as memory-dependent, which just corresponds to an important feature of the time-fractional derivative. For theoretical analysis, the parameter averaging method is not convenient; however, the time-fractional derivative is more suitable. Again, the energy dissipation law and numerical stability of the time-fractional derivative are all inherited. Based on the above advantages, the new TFGD algorithm is proposed. We verify that the TFGD algorithm has potential advantages when the fractional order parameter α nears 0.95∼0.99. This implies that the memory dependence could improve the training performance of neural networks.

There are many exciting directions for future works. The TFGD algorithm is only applied to the function fitting and the image classification in one dimension. In order to verify the applicability and effectiveness of this algorithm, we consider more general numerical examples, such as fittings of the multi-dimensional function, CIFAR10, CIFAR100, and the other problems of image classification. The convergence analysis of this algorithm could be developed for future research. The TFGD algorithm is a new attempt that combines the averaging method in neural networks with the time-fractional gradient descent. We hope that the algorithm will inspire further work in this area.

Author Contributions: The contributions of J.X. and S.L. are equal. All authors have read and agreed to the published version of the manuscript.

Funding: Sirui Li is supported by the Growth Foundation for Youth Science and Technology Talent of Educational Commission of Guizhou Province of China under grant No. [2021]087.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to thank the editors and the referees for their valuable comments and suggestions to improve our paper. We also would like to thank Yongqiang Cai from Beijing Normal University and Fanhai Zeng from Shandong University for their helpful discussions and recommendations regarding this problem.

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

References

1. Bottou, L.; Curtis, F.; Nocedal, J. Optimization methods for large-scale machine learning. *SIAM Rev.* **2018**, *60*, 223–311. [CrossRef]
2. Bottou, L. Large-scale machine learning with stochastic gradient descent. In Proceedings of the International Conference on Computational Statistics, Paris, France, 22–27 August 2010; pp. 177–186.
3. Hardt, M.; Recht, B.; Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016.
4. Polyak, B.T.; Juditsky, A.B. Acceleration of stochastic approximation by averaging. *SIAM J. Control. Optim.* **1992**, *30*, 838–855. [CrossRef]
5. Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In Proceedings of the 20th International Conference on Machine Learning (ICML-03), Washington, DC, USA, 21–24 August 2003; pp. 928–936.
6. Rakhlin, A.; Shamir, O.; Sridharan, K. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv* **2011**, arXiv:1109.5647.
7. Shamir, O.; Zhang, T. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; pp. 71–79.
8. Guo, Z.; Yan, Y.; Yang, T. Revisiting SGD with increasingly weighted averaging: Optimization and generalization perspectives. *arXiv* **2020**, arXiv:2003.04339v3.
9. Tao, W.; Pan, Z.; Wu, G.; Tao, Q. Primal averaging: A new gradient evaluation step to attain the optimal individual convergence. *IEEE Trans. Cybern.* **2020**, *50*, 835–845. [CrossRef] [PubMed]
10. Izmailov, P.; Podoprikhin, D.; Garipov, T.; Vetrov, D.; Wilson, A.G. Averaging weights leads to wider optima and better generalization. In Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence (UAI-2018), Monterey, CA, USA, 6–10 August 2018; pp. 876–885.
11. Khan, S.; Malik, M.A.; Togneri, R.; Bennamoun, M. A fractional gradient descent-based RBF neural network. *Circuits Syst. Signal Process.* **2018**, *37*, 5311–5332. [CrossRef]

12. Bao, C.; Pu, Y. ; Zhang, Y. Fractional-order deep back propagation neural Network. *Comput. Intell. Neurosci.* **2018**, *2018*, 7361628. [CrossRef] [PubMed]
13. Wang, J.; Wen, Y.; Gou, Y.; Ye, Z.; Chen, H. Fractional-order gradient descent learning of BP neural networks with Caputo derivative. *Neural Netw.* **2017**, *89*, 19–30. [CrossRef] [PubMed]
14. Chen, M.; Chen, B.; Zeng, G.; Lu, K.; Chu, P. An adaptive fractional-order BP neural network based on extremal optimization for handwritten digits recognition. *Neurocomputing* **2020**, *391*, 260–272. [CrossRef]
15. Wei, Y.; Kang, Y.; Yin, W.; Wang, Y. Generalization of the gradient method with fractional order gradient direction. *J. Frankl. Inst.* **2020**, *357*, 2514–2532. [CrossRef]
16. Du, Q.; Yang, J.; Zhou, Z. Time-fractional Allen-Cahn equations: Analysis and numerical methods. *J. Sci. Comput.* **2020**, *42*, 85. [CrossRef]
17. Liao, H.L.; Tang, T.; Zhou, T. An energy stable and maximum bound preserving scheme with variable time steps for time fractional Allen-Cahn equation. *SIAM J. Sci. Comput.* **2021**, *43*, A3503–A3526. [CrossRef]
18. Liu, H.; Cheng, A. ; Wang, H.; Zhao, J. Time-fractional Allen-Cahn and Cahn-Hilliard phase-field models and their numerical investigation. *Comput. Math. Appl.* **2018**, *76*, 1876–1892. [CrossRef]
19. Quan, C.; Tang, T.; Yang, J. How to define dissipation-preserving energy for timefractional phase-field equations. *CSIAM Trans. Appl. Math.* **2020**, *1*, 478–490. [CrossRef]
20. Tang, T.; Yu, H.; Zhou, T. On energy dissipation theory and numerical stability for time-fractional phase-field equations. *SIAM J. Sci. Comput.* **2019**, *41*, A3757–A3778. [CrossRef]
21. Rahman, Z.; Abdeljabbar, A.; Roshid, H.; Ali, M.Z. Novel precise solitary wave solutions of two time fractional nonlinear evolution models via the MSE scheme. *Fractal Fract.* **2022**, *6*, 444. [CrossRef]
22. Abdeljabbar, A.; Roshid, H.; Aldurayhim, A. Bright, dark, and rogue wave soliton solutions of the quadratic nonlinear Klein-Gordon equation. *Symmetry* **2022**, *14*, 1223. [CrossRef]
23. Alsaedi, A.; Ahmad, B.; Kirane, M. Maximum principle for certain generalized time and space-fractional diffusion equations. *Quart. App. Math.* **2015**, *73*, 163–175. [CrossRef]