algorithms

MDPI

Article

An Integer-Fractional Gradient Algorithm for Back Propagation Neural Networks

Yiqun Zhang, Honglei Xu *, Yang Li, Gang Lin, Liyuan Zhang, Chaoyang Tao and Yonghong Wu

School of Electrical Engineering, Computation and Mathematical Sciences, Curtin University, Kent Street, Perth, WA 6102, Australia; yiqun.zhang@postgrad.curtin.edu.au (Y.Z.); yang.li32@student.curtin.edu.au (Y.L.); gang.lin@postgrad.curtin.edu.au (G.L.); liyuan.zhang2@student.curtin.edu.au (L.Z.); chaoyang.tao@student.curtin.edu.au (C.T.); y.wu@curtin.edu.au (Y.W.)

* Correspondence: h.xu@curtin.edu.au

Abstract: This paper proposes a new optimization algorithm for backpropagation (BP) neural networks by fusing integer-order differentiation and fractional-order differentiation, while fractional-order differentiation has significant advantages in describing complex phenomena with long-term memory effects and nonlocality, its application in neural networks is often limited by a lack of physical interpretability and inconsistencies with traditional models. To address these challenges, we propose a mixed integer-fractional (MIF) gradient descent algorithm for the training of neural networks. Furthermore, a detailed convergence analysis of the proposed algorithm is provided. Finally, numerical experiments illustrate that the new gradient descent algorithm not only speeds up the convergence of the BP neural networks but also increases their classification accuracy.

Keywords: BP neural networks; fractional-order differentiation; integer-order differentiation

Citation: Zhang, Y.; Xu, H.; Li, Y.; Lin, G.; Zhang, L.; Tao, C.; Wu, Y. An Integer-Fractional Gradient Algorithm for Back Propagation Neural Networks. *Algorithms* **2024**, *17*, 220. https://doi.org/10.3390/a17050220

Academic Editor: Francesc Pozo

Received: 23 April 2024
Revised: 13 May 2024
Accepted: 18 May 2024
Published: 20 May 2024

1. Introduction

Integer-order differentiation is a powerful tool for solving problems involving rates of change. Its roots are in Newton and Leibniz's research and it has evolved into a fundamental branch of calculus. In addition, it possesses several excellent characteristics including ease of understanding, reliable numerical stability, and physical solvability, which make it highly applicable in real-world situations [1].

Meanwhile, fractional-order differentiation is an extended form of integer-order differentiation that enables the description of more complex phenomena such as long-term memory effects and nonlocality [2]. In the 1960s, it was first used to describe how heat energy moves through materials in the study of heat conduction and diffusion [3]. Over time, fractional-order differentiation spread to numerous other fields. In recent years, many researchers have established various physical models via fractional-order differentiation, leading to high-quality approximate solutions. For example, Matlob et al. [4] delved into the application of fractional-order differential calculus in modeling viscoelastic systems. Dehestani et al. [5] introduced fractional-order Legendre–Laguerre functions and explored their applications in solving fractional partial differential equations. Yuxiao et al. [6] discussed the variable order fractional grey model. Kuang et al. [7] focused on the application of fractional-order variable-gain supertwisting control to stabilize and control wafer stages in photolithography systems. Liu et al. [8] presented a two-stage fractional dynamical system model for microbial batch processes. These researchers have established physical models that utilize fractional-order differentiation, leading to high-quality approximate solutions. In addition, many optimal control problems also involve fractional-order differentiation. For example, Wang, Li, and Liu et al. [9] focused on establishing necessary optimality conditions and exact penalization techniques for constrained fractional optimal control problems. Bhrawy et al. [10] solved fractional optimal control problems using a Chebyshev–Legendre operational technique. Saxena et al. [11] presented a control strategy for load frequency

using a fractional-order controller combined with reduced-order modeling. Mohammadi et al. [12] used Caputo-Fabrizio fractional modeling for analyzing hearing loss due to the Mumps virus.

The emergence of neural networks has greatly improved our ability to analyze and process huge volumes of data and deal with nonlinearity [13]. The concept of neural networks originated in the 1950s when people first used computers to simulate human thinking processes. The earliest neural network had a simple structure comprising neuron models and connection weights. With advancements in computer technology and data processing, neural network research experienced a breakthrough in the 1990s. During that period, the backpropagation algorithm and deep learning were introduced to enhance the performance of neural networks [14].

In recent years, fractional-order differentiation has found wide applications in neural networks with the rapid development of new technologies like big data, high-performance computing, and deep learning [15]. For example, Xu et al. [16] analyzed the characteristic equations of a neural network model using two different delays as bifurcation parameters. They developed a new fractional-order neural network model with multiple delays and demonstrated the impact of fractional order on stability. Pakdaman et al. [17] employed optimization methods to adjust the weights of artificial neural networks to solve different fractional-order equations, including linear and nonlinear terms. This approach ensures that the approximate solutions meet the requirements of the fractional-order equations. Similarly, Asgharinia et al. [18] employed a fractional-order controller in training radial basis function neural networks, resulting in improved performance and robustness. More evidence that fractional-order differentiation actively contributes to neural networks and produces desirable results is shown in the works of Fei et al. [19], Zhang et al. [20], and Cao et al. [21]. Comparing these models to integer-order models reveals that fractional-order neural networks exhibit high accuracy, fast convergence, and efficient memory usage. For BP neural networks, Bao et al. [22] and Han et al. [23] applied Caputo and Grünwald–Letnikov fractional-order differentiation to investigate their convergence performances, respectively, and achieved better results than those found with integer-order differentiation. However, only using fractional-order differentiation always lacks physical interpretation and produces inconsistency with many classical models. Thus, it necessitates the further study of the integration of the advantages of both integer-order differentiation and fractional-order differentiation for gradient descent learning of neural networks to improve the performance of neural networks in classification problems, which is what motivates the research.

This paper's main contribution is to incorporate fractional-order differentiation, which can describe the memory effect and complex dynamic behavior, to optimize the traditional integer-order gradient descent process and construct a novel BP neural network using the proposed MIF optimization algorithm. Experimental results demonstrate that the new neural network exhibits both integer-order and fractional-order differentiation advantages.

The remainder of this paper proceeds as follows: Section 2 proposes the novel MIF optimization algorithm for BP neural networks; this is followed by the corresponding convergence analysis in Section 3. Next, Section 4 illustrates the numerical experiments and the result analysis. Finally, Section 5 provides concluding remarks.

2. Method

2.1. Network Structure

We employ a three-layer BP neural network structure, as illustrated in Figure 1. In the neural network model, the input layer consists of n neurons, the hidden layer contains h neurons, and the output layer has m neurons. The neural network has both forward propagation and backward propagation. The forward propagation process consists of learning data features of the neural network. For given sample data, let $\mathbf{x} = (x_1,\ x_2, ...,\ x_n)^T$ be the variable vector of the input layer, $\mathbf{y} = (y_1,\ y_2, ...,\ y_h)^T$ the variable vector of the hidden layer, $\mathbf{z} = (z_1,\ z_2, ...,\ z_m)^T$ the variable vector of the output layer, and $\hat{\mathbf{z}} = (\hat{z}_1, \hat{z}_2, ..., \hat{z}_m)^T$

the desired output. Moreover, let $P = (p_{ji})$ be the weight matrix connected with the input layer and the hidden layer, and $Q = (q_{kj})$ the weight matrix connected to the hidden layer and the output layer, where $i = 1, \ldots, n$, $j = 1, \ldots, h$, and $k = 1, \ldots, m$. The input and output of neurons in the j^{th} hidden layer are described by

$$\text{neth}_j = \sum_{i=1}^{n} p_{ji} x_i = P_j \mathbf{x} \tag{1}$$

and

$$y_j = f(\text{neth}_j) \tag{2}$$

where $P_j = (p_{j1}, p_{j2}, \ldots, p_{jn})$, and f is an activation function. The k^{th} output layer's input and output are given by

$$\text{neto}_k = \sum_{j=1}^{h} q_{kj} y_j = Q_k \mathbf{y} \tag{3}$$

and

$$z_k = f(\text{neto}_k) \tag{4}$$

where $Q_k = (q_{k1}, q_{k2}, \ldots, q_{kh})$. Then, in every iteration, we obtain the output result of the neural network. Now, we consider a total of t sample inputs. For the d-th sample input ($d \in \{1, \ldots, t\}$), the neural network model automatically corrects the weight p_{ji} and the weight q_{kj}. The total error equation of the neural network's performance evaluation is formulated by

$$E = \frac{1}{2} \sum_{d=1}^{t} \sum_{k=1}^{m} \left(\hat{z}_k - z_k^d \right)^2 = \sum_{d=1}^{t} g_d \left(\text{neto}_k^d \right) = \sum_{d=1}^{t} g_d \left(Q_k \mathbf{y}^d \right) \tag{5}$$

where

$$g_d \left(\text{neto}_k^d \right) = \frac{1}{2} \sum_{k=1}^{m} \left(\hat{z_k} - f \left(\text{neto}_k^d \right) \right)^2 \tag{6}$$

and

$$\mathbf{y}^d = \left(f \left(\text{neth}_1^d \right), f \left(\text{neth}_2^d \right), \ldots, f \left(\text{neth}_h^d \right) \right)^T \tag{7}$$

The error calculation process above is used as the forward propagation process of the neural network model. Its weight update by gradient descent through the error equation can be regarded as the backpropagation of the neural network. For the k-th output layer's error $E_k = \frac{1}{2} \sum_{d=1}^{t} \left(\hat{z}_k - z_k^d \right)^2$, according to the chain rule, the partial derivative of E_k with respect to q_{kj} is given by

$$\frac{\partial E_k}{\partial q_{kj}} = \sum_{d=1}^{t} \frac{\partial E_k}{\partial \text{neto}_k^d} \frac{\partial \text{neto}_k^d}{\partial q_{kj}} = \sum_{d=1}^{t} g_d' \left(\text{neto}_k^d \right) y_j^d \tag{8}$$

Similarly, the partial derivative of E_k with respect to p_{ji} is expressed by

$$\frac{\partial E_k}{\partial p_{ji}} = \sum_{d=1}^{t} g_d' \left(\text{neto}_k^d \right) q_{kj} f'(\text{neth}_j^d) x_i^d \tag{9}$$

Then, the update functions of the weights q_{kj} and p_{ji} are calculated by

$$q_{kj}^l = q_{kj}^{l-1} - \eta \frac{\partial E_k}{\partial q_{kj}^{l-1}} \tag{10}$$

and

$$p_{ji}^{l} = p_{ji}^{l-1} - \eta \frac{\partial E_k}{\partial p_{ji}^{l-1}} \tag{11}$$

where l is the iteration number, and η is the learning rate. The neural network's backward propagation process is implemented by using the updated weights, which will be applied to the next iteration.

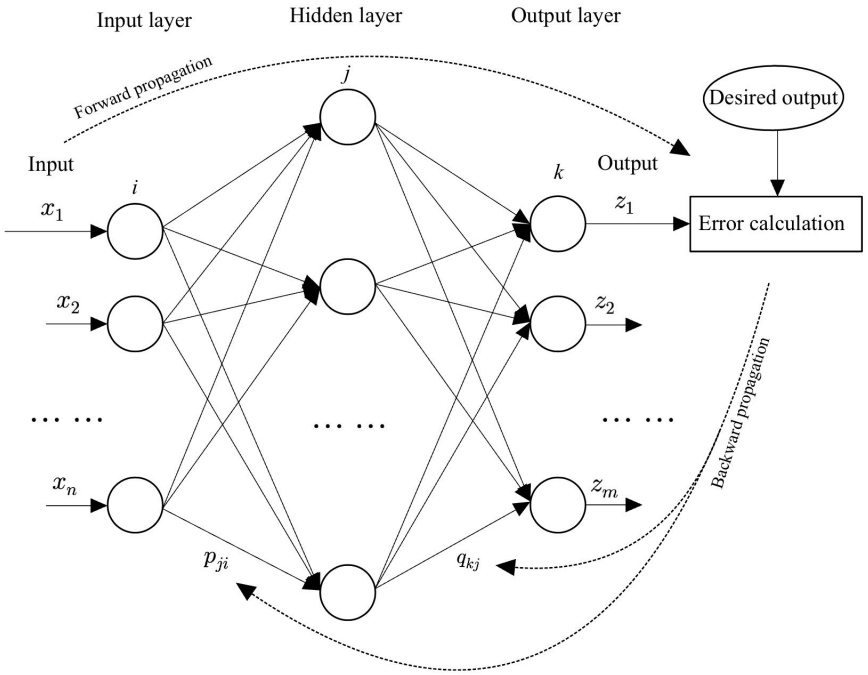

Figure 1. A three-layer BP neural network architecture.

2.2. Fractional Parameter Update

We incorporate the fractional derivative parameter update method into the neural network to prevent the neural network from becoming trapped in local optima during the training process [24]. In the existing literature, the Riemann–Liouville fractional derivative, the Caputo fractional derivative, and the Grunwald–Letnikov fractional derivative are the three most commonly used types of fractional-order derivatives.

Definition 1. *The Riemann–Liouville fractional-order derivative of $f(x)$ is defined as*

$${}_{a}^{RL}D_{x}^{\alpha} f(x) = D^{n}({}_{a}D_{x}^{\alpha-n} f(x)) = \frac{1}{\Gamma(n-\alpha)} \frac{d^{n}}{dx^{n}} \int_{a}^{x} \frac{f(t)}{(x-t)^{\alpha-n+1}} dt \tag{12}$$

where α is a positive real number and n is the smallest integer greater than α, and $\Gamma = \int_0^\infty t^{z-1} e^{-t} dt$.

Definition 2. *The Caputo fractional-order derivative of $f(x)$ is defined as*

$${}_{a}^{C}D_{x}^{\alpha} f(x) = \frac{1}{\Gamma(n-\alpha)} \int_{a}^{x} \frac{f^{(n)}(t)}{(x-t)^{\alpha-n+1}} dt = \frac{1}{\Gamma(n-\alpha)} \int_{a}^{x} (x-t)^{\alpha-n+1} f(t) dt \tag{13}$$

where α is a positive real number and $n-1 < \alpha < n$.

Definition 3. *The Grunwald–Letnikov fractional-order derivative $f(x)$ is defined as*

$${}_{a}^{GL}D_{x}^{\alpha} = \lim_{h \to 0} \frac{1}{h^{\alpha}} \sum_{k=0}^{\left[\frac{x-a}{h}\right]} \frac{(-1)^{k} \Gamma(\alpha+1)}{\Gamma(k+1)\Gamma(\alpha-k+1)} f(x-kh) \tag{14}$$

where α is a positive real number.

We employ Caputo fractional-order differentiation in this paper for the following reasons: (1) Caputo fractional-order differentiation has a more unambiguous physical significance. It can be regarded as performing integer-order differentiation on function f first, followed by integration with a weight. This weight can be viewed as a "memory effect", in which the function's value at time x is influenced by its value at time t, with the degree of influence decreasing with the increase in time interval $x - t$ [25]. (2) The Caputo fractional order's initial point is the same as that for the integer order. We know that the Caputo fractional-order derivative's initial value problem can be written as $f(0)$, $f'(0), \ldots f^{(n-1)}(0)$. These are all integer-order derivatives in those forms. Since the initial values are known and the calculation of complicated fractional-order derivatives is not necessary, this greatly simplifies the problem's solution [26]. (3) In comparison to the other two definitions of fractional order, the Caputo fractional-order derivative in the application of neural networks greatly reduces the amount of calculation. Meanwhile, it is closer to the integer order, making it more appropriate for neural networks' error calculation process.

In order to simplify the computational complexity of neural networks, this paper chooses the power function and calculates its fractional derivative. For the power function $f(x) = (x-a)^p$, its n-th derivative is

$$f^{(n)}(x) = \frac{\Gamma(p+1)}{\Gamma(p-n+1)}(x-a)^{p-n} \tag{15}$$

Substituting $f^{(n)}(x)$ into the Caputo derivative definition 2 yields

$${}_a^C D_x^\alpha = \frac{1}{\Gamma(n-\alpha)} \int_a^x \frac{\frac{\Gamma(p+1)}{\Gamma(p-n+1)}(t-a)^{p-n}}{(x-t)^{\alpha-n+1}} dt \tag{16}$$

This integral can be further simplified by the properties of the Gamma function. When $n = p$ (in which case p is an integer and $p \geq \alpha$), the integration becomes relatively simple. Finally, we obtain a specific form of fractional derivative for the power function used in this paper

$$\frac{d^\alpha}{dx^\alpha}(x-a)^p = \frac{\Gamma(p+1)}{\Gamma(p-\alpha+1)}(x-a)^{p-\alpha}$$

Therefore, the gradient is calculated by

$$D_{q_{kj}}^\alpha E_k = \sum_{d=1}^{t} \frac{\left(q_{kj}\right)^{1-\alpha}}{\Gamma(2-\alpha)} g_d'\left(\text{neto}_k^d\right) y_j^d \tag{17}$$

$$D_{p_{ji}}^\alpha E_k = \sum_{d=1}^{t} \frac{\left(p_{ji}\right)^{1-\alpha}}{\Gamma(2-\alpha)} g_d'\left(\text{neto}_k^d\right) q_{kj} f'(\text{neth}_j^d) x_i^d \tag{18}$$

Finally, according to Formulas (1)–(18), combining fractional-order gradients and the integer-order gradient, the updated weights of the neural network model are

$$q_{kj}^l = q_{kj}^{l-1} - \eta \left(\frac{\partial E_k}{\partial q_{kj}^{l-1}} + \tau D_{q_{kj}^{l-1}}^\alpha E_k \right) \tag{19}$$

$$p_{ji}^l = p_{ji}^{l-1} - \eta \left(\frac{\partial E_k}{\partial p_{ji}^{l-1}} + \tau D_{p_{ji}^{l-1}}^\alpha E_k \right) \tag{20}$$

where τ is a parameter greater than zero.

2.3. Algorithms

This subsection will provide the proposed MIFBP neural network (MIFBPNN) algorithms. To further demonstrate the algorithm's superiority, this paper compares this algorithm to the BP neural network (BPNN) and fractional-order BP neural network (FBPNN) algorithms. These three algorithms are presented in Algorithms 1–3.

Algorithm 1 BPNN

1. Initialize the weight matrices p_{ji}, q_{kj}, and the learning parameter η
2. Compute the l^{th} error function $E_k = \sum_{d=1}^{t} g_d\left(Q_k \mathbf{y}^d\right)$
3. Update the weight matrices by $q_{kj}^{l} = q_{kj}^{l-1} - \eta \frac{\partial E_k}{\partial q_{kj}^{l-1}}$ and $p_{ji}^{l} = p_{ji}^{l-1} - \eta \frac{\partial E_k}{\partial p_{ji}^{l-1}}$
4. Check if the total error function E meets the accuracy requirements or $l > l_{\max}$, exit the algorithm; If not, go to step 2.

Algorithm 2 FBPNN

1. Initialize the weight matrices p_{ji}, q_{kj}, the learning parameter η, and the parameter α
2. Compute the l^{th} error function $E_k = \sum_{d=1}^{t} g_d\left(Q_k \mathbf{y}^d\right)$
3. Update the weight matrices by $q_{kj}^{l} = q_{kj}^{l-1} - \eta D^{\alpha}_{q_{kj}^{l-1}} E_k$ and $p_{ji}^{l} = p_{ji}^{l-1} - \eta D^{\alpha}_{p_{ji}^{l-1}} E_k$
4. Check if the total error function E meets the accuracy requirements or $l > l_{\max}$, exit the algorithm; If not, go to step 2.

Algorithm 3 MIFBPNN

1. Initialize the weight matrices p_{ji}, q_{kj}, the learning parameter η, and the parameters α and τ
2. Compute the l^{th} error function $E_k = \sum_{d=1}^{t} g_d\left(Q_k \mathbf{y}^d\right)$
3. Update the weight matrices by

$$q_{kj}^{l} = q_{kj}^{l-1} - \eta \left(\frac{\partial E_k}{\partial q_{kj}^{l-1}} + \tau D^{\alpha}_{q_{kj}^{l-1}} E_k \right)$$

and

$$p_{ji}^{l} = p_{ji}^{l-1} - \eta \left(\frac{\partial E_k}{\partial p_{ji}^{l-1}} + \tau D^{\alpha}_{p_{ji}^{l-1}} E_k \right)$$

4. Check if the total error function E meets the accuracy requirements or $l > l_{\max}$, exit the algorithm; If not, go to step 2.

3. Convergence Analysis

In this section, we will prove the convergence properties of the MIFBPNN. To proceed, we need the following assumptions.

Assumption 1. *The activation function f is a Sigmoid function.*

Assumption 2. *The weights q_{kj} and p_{ji} are bounded during the MIFBPNN training process.*

Assumption 3. *The learning rate η and the parameter τ are positive numbers and have upper bounds.*

We have the following main results.

Theorem 1. *Suppose that the Assumptions (1)–(3) are satisfied, then we conclude that the MIF-BPNN is convergent.*

Proof of Theorem 1. This proof consists of two parts. First, we prove that the error function sequence E^l is monotonic. Then we demonstrate that the sequence E^l is bounded. To explore the monotonicity of the sequence E^l, we compute

$$E^* = E^l - E^{l-1} = \sum_{d=1}^{t}\sum_{k=1}^{m}\left(g_d\left(Q_k^l \mathbf{y}^{d,l}\right) - g_d\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)\right) \tag{21}$$

According to the Taylor expansion formula and (21), we obtain

$$\begin{aligned} &g_d\left(Q_k^l \mathbf{y}^{d,l}\right) - g_d\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right) \\ &= g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)\left(Q_k^l \mathbf{y}^{d,l} - Q_k^{l-1}\mathbf{y}^{d,l-1}\right) \\ &+ \frac{1}{2}g_d''(\varepsilon_1)\left(Q_k^l \mathbf{y}^{d,l} - Q_k^{l-1}\mathbf{y}^{d,l-1}\right)^2 \end{aligned}$$

where ε_1 is the Lagrange constant between $Q_k^l \mathbf{y}^{d,l}$ and $Q_k^{l-1}\mathbf{y}^{d,l-1}$.

Moreover, we see that

$$\begin{aligned} Q_k^l \mathbf{y}^{d,l} - Q_k^{l-1}\mathbf{y}^{d,l-1} &= Q_k^l\left(\mathbf{y}^{d,l} - \mathbf{y}^{d,l-1}\right) + \left(Q_k^l - Q_k^{l-1}\right)\mathbf{y}^{d,l-1} \\ &= Q_k^{l-1}\xi_d^l + \Delta Q_k^l \xi_d^l + \Delta Q_k^l \mathbf{y}^{d,l-1} \end{aligned} \tag{22}$$

where $\Delta Q_k^l = Q_k^l - Q_k^{l-1}$ and $\xi_d^l = \mathbf{y}^{d,l} - \mathbf{y}^{d,l-1}$

Then, substituting (22) to (21) yields

$$\begin{aligned} E^* &= \sum_{d=1}^{t}\sum_{k=1}^{m} g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)Q_k^{l-1}\xi_d^l + \sum_{d=1}^{t}\sum_{k=1}^{m} g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)\Delta Q_k^l \xi_d^l \\ &+ \sum_{d=1}^{t}\sum_{k=1}^{m} g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)\Delta Q_k^l \mathbf{y}^{d,l-1} + \frac{1}{2}\sum_{d=1}^{t}\sum_{k=1}^{m} g_d''(\varepsilon_2)\left(Q_k^l \mathbf{y}^{d,l} - Q_k^{l-1}\mathbf{y}^{d,l-1}\right)^2 \end{aligned} \tag{23}$$

We process $Q_k^{l-1}\xi_d^l$ according to the Taylor extension formula to obtain

$$\begin{aligned} Q_k^{l-1}\xi_d^l &= Q_k^{l-1}\left(\mathbf{y}^{d,l} - \mathbf{y}^{d,l-1}\right) \\ &= \sum_{j=1}^{h} q_{kj}^{l-1}\left(f\left(P_j^l \mathbf{x}^d\right) - f\left(P_j^{l-1}\mathbf{x}^d\right)\right) \\ &= \sum_{j=1}^{h} q_{kj}^{l-1}\left(f'\left(P_j^{l-1}\mathbf{x}^d\right)\Delta P_j^l \mathbf{x}^d + \frac{1}{2}f''(\varepsilon_2)\left(\Delta P_j^l \mathbf{x}^d\right)^2\right) \end{aligned} \tag{24}$$

where $\Delta P_j^l = P_j^l - P_j^{l-1}$ and ε_2 is the Lagrange constant between $P_j^l \mathbf{x}^d$ and $P_j^{l-1}\mathbf{x}^d$. According to (20), (23) and (24), we have

$$\begin{aligned}
&\sum_{d=1}^{t}\sum_{k=1}^{m} g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)Q_k^{l-1}\xi_d^l \\
&= \sum_{d=1}^{t}\sum_{k=1}^{m}\sum_{j=1}^{h}\sum_{i=1}^{n} g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)q_{kj}^{l-1}f'\left(P_j^{l-1}\mathbf{x}^d\right)\Delta p_{ji}^l x_i^d \\
&+ \frac{1}{2}\sum_{d=1}^{t}\sum_{k=1}^{m}\sum_{j=1}^{h} g_d'\left(Q^{l-1}\mathbf{y}^{d,l-1}\right)q_{kj}^{l-1}f''(\varepsilon_2)\left(\Delta P_j^l\mathbf{x}^d\right)^2
\end{aligned} \tag{25}$$

where

$$\Delta p_{ji}^l = p_{ji}^l - p_{ji}^{l-1} = -\eta\left(\frac{\partial E_k}{\partial p_{ji}^{l-1}} + \tau D^{\alpha}_{p_{ji}^{l-1}}E_k\right) = -\eta\left(\left(p_{ji}^{l-1}\right)^{\alpha-1}\Gamma(2-\alpha)+\tau\right)D^{\alpha}_{p_{ji}^{l-1}}E_k \tag{26}$$

For simplicity, we define

$$\varphi_1 = \sum_{d=1}^{t}\sum_{k=1}^{m}\sum_{j=1}^{h}\sum_{i=1}^{n} g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)q_{kj}^{l-1}f'\left(P_j^{l-1}\mathbf{x}^d\right)\Delta p_{ji}^l x_i^d \tag{27}$$

$$\varphi_2 = \frac{1}{2}\sum_{d=1}^{t}\sum_{k=1}^{m}\sum_{j=1}^{h} g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)q_{kj}^{l-1}f''(\varepsilon_2)\left(\Delta P_j^l\mathbf{x}^d\right)^2 \tag{28}$$

$$\varphi_3 = \sum_{d=1}^{t}\sum_{k=1}^{m} g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)\Delta Q_k^l\xi_d^l \tag{29}$$

$$\varphi_4 = \sum_{d=1}^{t}\sum_{k=1}^{m} g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)\Delta Q_k^l\mathbf{y}^{d,l-1} \tag{30}$$

$$\varphi_5 = \frac{1}{2}\sum_{d=1}^{t}\sum_{k=1}^{m} g_d''(\varepsilon_2)\left(Q_k^l\mathbf{y}^{d,l} - Q_k^{l-1}\mathbf{y}^{d,l-1}\right)^2 \tag{31}$$

From (17), we obtain

$$\sum_{d=1}^{t} g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)q_{kj}^{l-1}f'\left(P_j^{l-1}\mathbf{x}^d\right)x_i^d = \left(p_{ji}^{l-1}\right)^{\alpha-1}\Gamma(2-\alpha)D^{\alpha}_{p_{ji}^{l-1}}E_k \tag{32}$$

Furthermore, it follows from (26), (27) and (32) that

$$\begin{aligned}
\varphi_1 &= \Gamma(2-\alpha)\sum_{k=1}^{m}\sum_{j=1}^{h}\sum_{i=1}^{n}\left(p_{ji}^{l-1}\right)^{\alpha-1}\left(D^{\alpha}_{p_{ji}^{l-1}}E_k\right)\Delta p_{ji}^l \\
&= -\eta\Gamma(2-\alpha)\sum_{k=1}^{m}\sum_{j=1}^{h}\sum_{i=1}^{n}\left(p_{ji}^{l-1}\right)^{\alpha-1}\left(D^{\alpha}_{p_{ji}^{l-1}}E_k\right)^2\left(\left(p_{ji}^{l-1}\right)^{\alpha-1}\Gamma(2-\alpha)+\tau\right) \\
&\leqslant -\tau\eta\Gamma(2-\alpha)\sum_{k=1}^{m}\sum_{j=1}^{h}\sum_{i=1}^{n}\left(p_{ji}^{l-1}\right)^{\alpha-1}\left(D^{\alpha}_{p_{ji}^{l-1}}E_k\right)^2 \\
&= -\tau\eta\Gamma(2-\alpha)\sum_{k=1}^{m}\sum_{j=1}^{h}\left(p_{ji}^{l-1}\right)^{\alpha-1}\|D^{\alpha}_{P_j^{l-1}}E_k\|^2
\end{aligned} \tag{33}$$

where $D^{\alpha}_{P_j^{l-1}}E_k = (D^{\alpha}_{p_{ji}^{l-1}}E_k)_{1\times h}$ and

$$\varphi_3 \leqslant \frac{1}{2}\sum_{d=1}^{t}\sum_{k=1}^{m} g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)\left(\|\Delta Q_k^l\|^2 + \|\xi_d^l\|^2\right) \tag{34}$$

By using (2), (3), (18) and (19), we have

$$\sum_{d=1}^{t} g_d'\left(\text{neto}_k^d\right) y_j^d = \sum_{d=1}^{t} g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right) f\left(P_j^l \mathbf{x}^d\right) = \left(q_{kj}^{l-1}\right)^{\alpha-1}\Gamma(2-\alpha) D^{\alpha}_{q_{kj}^{l-1}} E_k \tag{35}$$

and

$$\varphi_4 = \sum_{d=1}^{t}\sum_{k=1}^{m}\sum_{j=1}^{h} g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right) y_j^{d,l-1}\Delta q_{kj}^l = \Gamma(2-\alpha)\sum_{k=1}^{m}\sum_{j=1}^{h}\left(q_{kj}^{l-1}\right)^{\alpha-1}\Delta q_{kj}^l\left(D^{\alpha}_{q_{kj}^{l-1}} E_k\right) \tag{36}$$

where

$$\Delta q_{kj}^l = q_{kj}^l - q_{kj}^{l-1} = -\eta\left(\frac{\partial E}{\partial q_{kj}^{l-1}} + \tau D^{\alpha}_{q_{kj}^{l-1}} E\right) = -\eta\left(\left(q_{kj}^{l-1}\right)^{\alpha-1}\Gamma(2-\alpha) + \tau\right) D^{\alpha}_{q_{kj}^{l-1}} E_k \tag{37}$$

Thus, we obtain

$$\begin{aligned}
\varphi_4 &= -\eta\Gamma(2-\alpha)\sum_{k=1}^{m}\sum_{j=1}^{h}\left(q_{kj}^{l-1}\right)^{\alpha-1}\left(\left(q_{kj}^{l-1}\right)^{\alpha-1}\Gamma(2-\alpha)+\tau\right)\left(D^{\alpha}_{q_{kj}^{l-1}} E_k\right)^2 \\
&\leqslant -\tau\eta\Gamma(2-\alpha)\sum_{k=1}^{m}\sum_{j=1}^{h}\left(q_{kj}^{l-1}\right)^{\alpha-1}\left(D^{\alpha}_{q_{kj}^{l-1}} E_k\right)^2 \\
&= -\tau\eta\Gamma(2-\alpha)\sum_{k=1}^{m}\left(q_{kj}^{l-1}\right)^{\alpha-1}\|D^{\alpha}_{Q_k^{l-1}} E_k\|^2
\end{aligned} \tag{38}$$

where $D^{\alpha}_{Q_k^{l-1}} E_k = (D^{\alpha}_{q_{kj}^{l-1}} E_k)_{1\times m}$.

$$\begin{aligned}
\varphi_5 &= \frac{1}{2} g_d''(\varepsilon_2)\sum_{d=1}^{t}\sum_{k=1}^{m}\left(Q_k^l\mathbf{y}^{d,l} - Q_k^{l-1}\mathbf{y}^{d,l} + Q_k^{l-1}\mathbf{y}^{d,l} - Q_k^{l-1}\mathbf{y}^{d,l-1}\right)^2 \\
&= \frac{1}{2} g_d''(\varepsilon_2)\sum_{d=1}^{t}\sum_{k=1}^{m}\left(\Delta Q_k^l\mathbf{y}^{d,l} + Q_k^{l-1}\xi_d^l\right)^2 \\
&\leqslant g_d''(\varepsilon_2)\sum_{d=1}^{t}\sum_{k=1}^{m}\left(\|\Delta Q_k^l\|^2\|\mathbf{y}^{d,l}\|^2 + \|Q_k^{l-1}\|^2\|\xi_d^l\|^2\right)
\end{aligned} \tag{39}$$

Next, let

$$A_1 = \max\{\sup|g_d(t)|, \sup|g_d'(t)|, \sup|g_d''(t)|, \sup|f'(x)|, \sup|f''(x)|\}, \tag{40}$$

$$A_2 = \max\|\mathbf{x}^d\|,\ 1 \leqslant d \leqslant t \tag{41}$$

$$A_3 = \max\|\Phi\|, \tag{42}$$

where $\Phi = \left\{q_{kj}, p_{ji}\right\}$ represents the vector consisting of all stretched weighted parameters q_{kj} and p_{ji}. Then, we obtain through (33)–(42) that

$$\|\Delta Q_k^l\|^2 \leqslant 2\eta^2\left(A_3^{\alpha-1}\Gamma(2-\alpha)\right)^2 + \tau^2)\|D^{\alpha}_{Q_k^{l-1}} E_k\|^2 \tag{43}$$

$$\|\Delta P_j^l\|^2 \leqslant 2\eta^2\left(\left(A_3^{\alpha-1}\Gamma(2-\alpha)\right)^2 + \tau^2\right)\|D^{\alpha}_{P_j^{l-1}} E_k\|^2 \tag{44}$$

$$\begin{aligned}
&\|\Delta\Phi^l\|^2 \leqslant \|\Delta Q_k^l\|^2 + \|\Delta P_j^l\|^2 \\
&\leqslant 2\eta^2\left(\left(A_3^{\alpha-1}\Gamma(2-\alpha)\right)^2 + \tau^2\right)\left(\|D^{\alpha}_{Q_k^{l-1}} E_k\|^2 + \|D^{\alpha}_{P_j^{l-1}} E_k\|^2\right)
\end{aligned} \tag{45}$$

Moreover, (24), (43) and (44) imply

$$\|\xi_d^l\|^2 = \|\mathbf{y}^{d,l} - \mathbf{y}^{d,l-1}\|^2 = \|P_j^l\mathbf{x}^d - P_j^{l-1}\mathbf{x}^d\|^2 = \|f'(\varepsilon_2)\Delta P_j^l\mathbf{x}^d\|^2 \leqslant A_1^2A_2^2\|\Delta P_j^l\|^2 \tag{46}$$

Hence, from (27)–(46), we obtain

$$\varphi_1 \leqslant -\tau\eta m\Gamma(2-\alpha)A_3^{\alpha-1}\|D^{\alpha}_{P_j^{l-1}}E_k\|^2 \tag{47}$$

$$\varphi_2 \leqslant \frac{1}{2}D\sum_{k=1}^{m}\sum_{j=1}^{h}A_1A_3g_d'\left(Q_k^{l-1}\mathbf{y}^{d,l-1}\right)\left(\Delta P_j^{l-1}\mathbf{x}^d\right)^2 \leqslant \frac{1}{2}mDA_1^2A_2^2A_3 \parallel \Delta\Phi \parallel^2 \tag{48}$$

$$\varphi_3 \leqslant \frac{1}{2}A_1\sum_{d=1}^{t}\sum_{k=1}^{m}\left(\left(\Delta Q_k^l\right)^2 + A_1^2A_2^2\left(\Delta P_j^l\right)^2\right) \leqslant \frac{mD}{2}A_1\left(1+A_1^2A_2^2\right)\|\Delta\Phi\|^2 \tag{49}$$

$$\varphi_4 \leqslant -\tau\eta m\Gamma(2-\alpha)A_3^{\alpha-1}\|D^{\alpha}_{Q_k^{l-1}}E_k\|^2 \tag{50}$$

$$\varphi_5 \leqslant A_1\sum_{d=1}^{t}\sum_{k=1}^{m}\left(hA_1^2 \parallel \Delta Q_k^l \parallel^2 + A_1^2A_2^2A_3^2 \parallel Q_k^{l-1} \parallel^2\right) \leqslant mDA_1\left(hA_1^2 + A_1^2A_2^2A_3^2\right) \parallel \Delta\Phi \parallel^2 \tag{51}$$

We note from (45) that

$$\|D^{\alpha}_{Q_k^{l-1}}E_k\|^2 + \|D^{\alpha}_{P_j^{l-1}}E_k\|^2 \geqslant \frac{\|\Delta\Phi^l\|^2}{2\eta^2\left(\left(A_3^{\alpha-1}\Gamma(2-\alpha)\right)^2 + \tau^2\right)} \tag{52}$$

So, using (46)–(52), we convert E^* into the following:

$$\begin{aligned}
E^* &\leqslant -\tau\eta m\Gamma(2-\alpha)A_3^{\alpha-1}\left(\|D^{\alpha}_{Q_k^{l-1}}E_k\|^2 + \|D^{\alpha}_{P_j^{l-1}}E_k\|^2\right)\\
&+ m\left(\frac{1}{2}DA_1^2A_2^2A_3 + \frac{1}{2}DA_1\left(1+A_1^2A_2^2\right) + DA_1\left(A_1^2 + A_1^2A_2^2A_3^2\right)\right) \parallel \Delta\Phi \parallel^2\\
&\leqslant -\frac{1}{2\eta}\frac{\tau m\Gamma(2-\alpha)A_3^{\alpha-1}}{\left(\left(A_3^{\alpha-1}\Gamma(2-\alpha)\right)^2+\tau^2\right)} \parallel \Delta\Phi \parallel^2\\
&+ m\left(\frac{1}{2}DA_1^2A_2^2A_3 + \frac{1}{2}DA_1\left(1+A_1^2A_2^2\right) + DA_1\left(hA_1^2 + A_1^2A_2^2A_3^2\right)\right) \parallel \Delta\Phi \parallel^2\\
&\leqslant -\frac{1}{2\eta}\frac{\tau m\Gamma(2-\alpha)A_3^{\alpha-1}}{2\tau\Gamma(2-\alpha)A_3^{\alpha-1}} \parallel \Delta\Phi \parallel^2\\
&+ m\left(\frac{1}{2}DA_1^2A_2^2A_3 + \frac{1}{2}DA_1\left(1+A_1^2A_2^2\right) + DA_1\left(hA_1^2 + A_1^2A_2^2A_3^2\right)\right) \parallel \Delta\Phi \parallel^2\\
&\leqslant -\frac{1}{4\eta} \parallel \Delta\Phi \parallel^2 + \left(\frac{1}{2}DA_1^2A_2^2A_3 + \frac{1}{2}DA_1\left(1+A_1^2A_2^2\right) + DA_1\left(hA_1^2 + A_1^2A_2^2A_3^2\right)\right) \parallel \Delta\Phi \parallel^2
\end{aligned} \tag{53}$$

Therefore, the error function sequence E^l is monotonically decreasing. Next, since

$$\frac{1}{4\eta} \geqslant \frac{1}{2}DA_1^2A_2^2A_3 + \frac{1}{2}DA_1\left(1+A_1^2A_2^2\right) + DA_1\left(hA_1^2 + A_1^2A_2^2A_3^2\right) \tag{54}$$

then, we obtain the learning rate

$$\eta \leqslant \frac{1}{2DA_1^2A_2^2A_3 + 2DA_1\left(1+A_1^2A_2^2\right) + 4DA_1\left(hA_1^2 + A_1^2A_2^2A_3^2\right)} \tag{55}$$

Now, we start to prove the above sequence is also bounded. Let

$$\delta^{l-1} = \|D^{\alpha}_{Q_k^{l-1}} E_k\|^2 + \|D^{\alpha}_{P_j^{l-1}} E_k\|^2 \tag{56}$$

According to (53)–(56) we obtain

$$\begin{aligned} E^* &\leqslant -\Big(\tau\eta m\Gamma(2-\alpha)A_3^{\alpha-1} \\ &-2\eta^2 m\Big(\Big(A_3^{\alpha-1}\Gamma(2-\alpha)\Big)+\tau^2\Big)\Big(\frac{1}{2}DA_1^2A_2^2A_3+\frac{1}{2}DA_1\Big(1+A_1^2A_2^2\Big)+DA_1\Big(hA_1^2+A_1^2A_2^2A_3^2\Big)\Big)\Big)\delta^{l-1} \\ &\leqslant -\Big(\tau\eta\Gamma(2-\alpha)A_3^{\alpha-1} \\ &-4\tau\eta^2\Gamma(2-\alpha)A_3^{\alpha-1}\Big(\frac{1}{2}DA_1^2A_2^2A_3+\frac{1}{2}DA_1\Big(1+A_1^2A_2^2\Big)+DA_1\Big(hA_1^2+A_1^2A_2^2A_3^2\Big)\Big)\Big)\delta^{l-1} \\ &\leqslant -\rho\delta^{l-1} \end{aligned} \tag{57}$$

where

$$\begin{aligned} \rho &= \tau\eta\Gamma(2-\alpha)A_3^{\alpha-1} \\ &-4\tau\eta^2\Gamma(2-\alpha)A_3^{\alpha-1}\Big(\frac{1}{2}DA_1^2A_2^2A_3+\frac{1}{2}DA_1\Big(1+A_1^2A_2^2\Big)+DA_1\Big(hA_1^2+A_1^2A_2^2A_3^2\Big)\Big) \end{aligned} \tag{58}$$

From (57), we infer that

$$0 \leqslant E^l \leqslant E^{l-1}-\rho\delta^{l-1} \leqslant E^{l-2}-\rho\delta^{l-2}-\rho\delta^{l-1} \leqslant E^{l-3}-\rho\delta^{l-3}-\rho\delta^{l-2}-\rho\delta^{l-1} \leqslant \ldots \leqslant E^0-\rho\sum_{w=0}^{l-1}\delta^w \tag{59}$$

Then,

$$\sum_{w=0}^{l-1}\delta^w \leqslant \frac{1}{\rho}E^0 \tag{60}$$

When $l \to \infty$, we have $\sum_{w=0}^{\infty}\delta^w \leqslant \frac{1}{\rho}E^0 < \infty$ and $\lim_{l\to\infty}\delta^l = 0$. Therefore, according to (57)–(59) the sequence E^l is bounded. Using the monotone convergence theorem, we conclude that the sequence E^l converges. □

4. Numerical Experiments

4.1. Experiment Preparation

In this section, we use the MNIST handwritten dataset to evaluate the efficiency of the MIFBPNN. We also use BPNN and FBPNN to perform comparison experiments. The MNIST handwritten digit dataset contains 70,000 handwritten digit images ranging from 0 to 9, each with a size of 28 × 28 pixels, represented as a 784 × 1 vector. Each element in the vector takes values ranging from 0 to 255. We first build a 784-50-10 neural network model ($n = 784, h = 30, m = 10$) based on the characteristics of the MNIST data, and in our experiments, we found that when the number of hidden layers is 50, we could achieve high accuracy in the test set, which greatly reduces the computational amount of the neural network model. The activation function uses the Sigmoid function, which is expressed as $f(x) = \frac{1}{1+e^{-x}}$. Then, we divided the dataset into 60,000 handwritten images to use as the training set, and the remaining 10,000 sets of data to use as the test set. As the convergence of the neural network does not mean that the network has completely learned all the features and patterns of the dataset, but rather that the network has reached a relatively stable state, in order to study this stable state, in the experiments of this paper, we set the condition: When the accuracy of the test set of every 10 epochs grows no more than 0.001, then exit the training. In this way, it is possible to determine whether the neural network has reached a stable state based on the number of epochs, and thus better evaluate

the neural network. (The experimental platform for these experiments was a PC running Windows OS, with an i7-11657G7 CPU and M40 GPU).

4.2. Optimal Parameters Tuning

In the experiment, we set the same parameters for each model to more effectively compare the three models used in this paper. There are three parameters in the proposed MIFBPNN. We first determined $\eta = 0.7$ and $\alpha = 0.6$ based on the BP neural network training result in Table 1 and the training result of the FBPNN in Table 2. Based on the training outcomes of the MIFBPNN in Table 3, we discovered $\tau = 0.2$.

Table 1. Test set accuracy of the BP neural network with various parameter values of η.

	$\eta = 0.1$	$\eta = 0.2$	$\eta = 0.3$	$\eta = 0.4$	$\eta = 0.5$	$\eta = 0.6$	$\eta = 0.7$	$\eta = 0.8$	$\eta = 0.9$
Accuracy	93.79%	94.15%	94.50%	95.20%	95.22%	95.30%	95.65%	95.60%	95.45%
Training epoch	246	297	245	257	217	191	163	197	203

Table 2. Test set accuracy of the FBPNN with $\eta = 0.7$ and various parameter values of α.

$\eta = 0.7$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$
Accuracy	96.28%	95.86%	95.87%	95.76%	96.01%	96.65%	96.13%	95.93%	95.70%
Training epoch	79	75	55	63	70	68	67	73	57

Table 3. Test set accuracy of the MIFBPNN with $\eta = 0.7$, $\alpha = 0.6$ and various parameter values of τ.

$\eta = 0.7, \alpha = 0.6$	$\tau = 0.1$	$\tau = 0.2$	$\tau = 0.3$	$\tau = 0.4$	$\tau = 0.5$	$\tau = 0.6$	$\tau = 0.7$	$\tau = 0.8$	$\tau = 0.9$
Accuracy	97.52%	97.64%	97.34%	97.30%	97.27%	97.41%	97.46%	97.49%	97.36%
Training epoch	63	71	54	55	50	51	47	56	55

Tables 1–3 show that the MIFBPNN achieves a stable state faster and has a higher accuracy rate than the FBPNN and BPNN, which may suggest that the MIFBPNN is able to learn better under the same circumstances.

4.3. Training for Different Training Sets' Sizes

A larger training set can provide more sample data for the model to learn from, which can help to improve the accuracy and generalizability of the model. Typically, the size of the training set can have an impact on the training results. Here, we investigate the performance of the three neural networks by adjusting various training set sizes, and the experimental results are displayed in Table 4, which shows that a larger training set can aid the model in better-capturing patterns and features from the data.

Table 4. Performance of different neural networks under optimal hyperparameters on various sizes of training sets.

Training Dataset Size	BP		FBP		MIFBP	
	Accuracy	Training Epoch	Accuracy	Training Epoch	Accuracy	Training Epoch
10,000	91.69%	124	92.99%	86	94.55%	79
20,000	93.71%	176	93.29%	76	96.06%	64
30,000	93.81%	140	93.85%	68	96.94%	69
40,000	94.20%	109	94.23%	58	97.17%	55
50,000	95.23%	157	95.69%	71	97.28%	61
60,000	95.65%	163	96.95%	68	97.64%	71

Table 4 shows that, in terms of training accuracy and the number of layers to stop training, the MIFBPNN performs best on training datasets of all sizes, followed by the FBPNN. It is worth noting that change in the MIFBPNN's accuracy from the training dataset of 10,000 to 60,000 was the smallest, followed by that of the FBPNN. This can also be viewed as evidence that the MIFBPNN is more general and has a stronger capacity for convergence and learning while maintaining high accuracy.

4.4. Training Performances of Different Neural Networks

In this subsection, we demonstrate the performance of the three neural networks using the optimal parameters and a training dataset of 60,000. Figures 2 and 3 depict the three neural networks' test accuracy and test set loss performance, respectively, and Figure 4 depicts the three neural networks' training set loss performance.

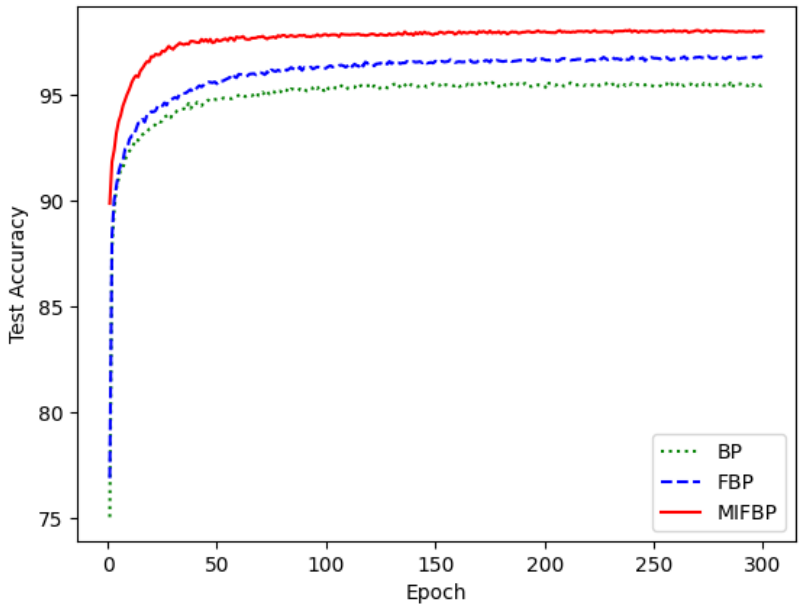

Figure 2. Test accuracy performance.

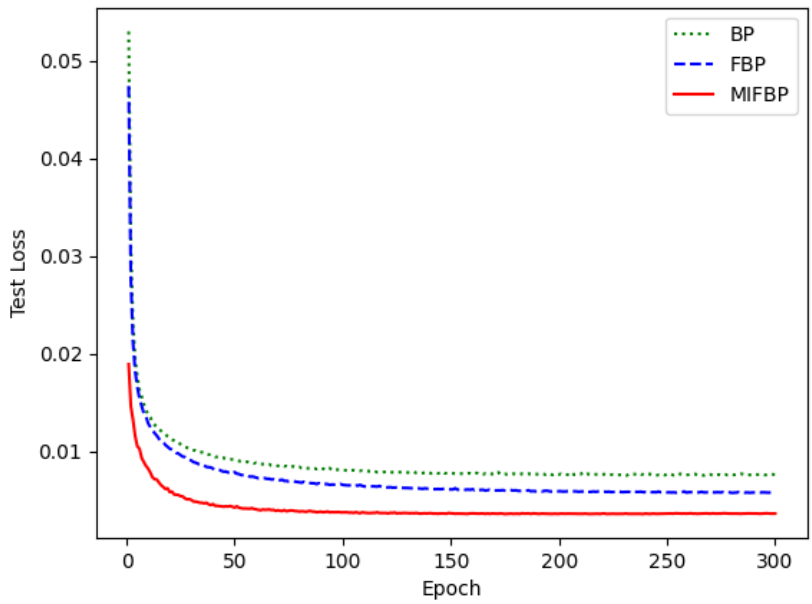

Figure 3. Test loss performance.

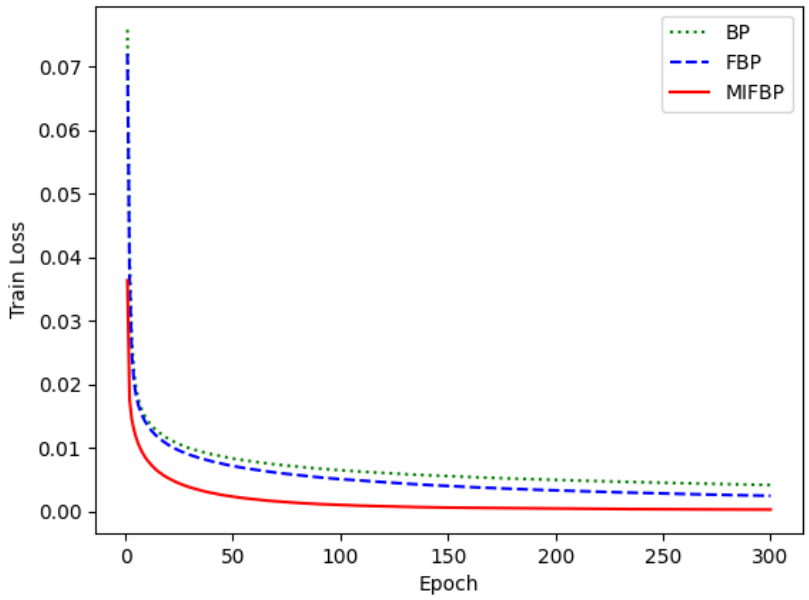

Figure 4. Training loss performance.

The MIFBPNN has the highest accuracy, followed by the FBPNN, as can be seen in Figure 3. In terms of convergence speed, the accuracy of the MIFBPNN and FBPNN reaches a relatively stable state in about 50 and 80 epochs, while the BP neural network experiences this process in about 120 epochs. The MIFBPNN has the smallest loss and can reach the minimum loss fastest, both in the training set loss and the test set loss, as shown by Figures 3 and 4.

Based on the above experimental results, it can be concluded that MIFBPNN exhibits high accuracy and fast convergence speed in various training set sizes under different experimental conditions. Especially on the maximum size training set (60,000 samples), the MIFBPNN not only achieved the highest accuracy (97.64%) but also had lower training epoch (71 epochs) than the other two networks, demonstrating its excellent learning ability and generalization ability. Furthermore, from the chart analysis, it can be seen that the MIFBPNN outperforms BPNN and FBPNN in terms of testing accuracy, testing loss, and training loss. Especially in terms of minimizing losses, MIFBP networks can achieve lower loss values faster, which is of great significance for optimizing and improving efficiency in practical applications. Secondly, the MIFBPNN also demonstrated its superiority in parameter adjustment experiments. After fixing $\eta = 0.7$ and $\alpha = 0.6$, and adjusting the τ parameters, it is found that the network performed best at $\tau = 0.2$, with an accuracy of 97.64%. This result not only demonstrates the advantage of MIFBPNN in parameter sensitivity but also highlights its adjustment flexibility and efficiency in practical applications. However, the limitation of this article is that it only applies the MIF algorithm to BPNN, and manual adjustments are used during the parameter adjustment process. In future work, attempts can be made to apply the MIF algorithm to other neural networks, and heuristic algorithms can be considered when adjusting parameters α, η, and τ. Furthermore, the computational effort of the MIFBPNN is not explained in this article. These issues are worth exploring in future work.

5. Conclusions

In this paper, we develop a new MIF gradient descent algorithm to construct the MIFBP neural network. Our research demonstrates the MIFBP neural network's superior performance on the typical MNIST handwritten dataset. For various training set sizes, the MIFBP network outperforms conventional BP and FBPNN in terms of accuracy, convergence speed, and generalization. Additionally, in both training and testing scenarios, the MIFBPNN achieves the highest accuracy and the lowest loss under optimal parame-

ter conditions. The MIFBPNN has shown good classification performance in the MNIST handwritten data and may perform challenging pattern recognition and prediction tasks in various fields in the future.

Author Contributions: Y.Z.: conceptualization, methodology, software, validation, formal analysis, investigation, and writing—original draft preparation. H.X.: conceptualization, methodology, writing—review and editing, supervision, and funding acquisition. Y.L.: methodology, project administration, and funding acquisition. G.L.: writing—review and editing, and validation. L.Z.: methodology, formal analysis, and investigation. C.T.: methodology, software, and validation. Y.W.: conceptualization and supervision. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded an Australian Research Council linkage grant (LP160100528), an innovative connection grant (ICG002517), and an industry grant (SE67504).

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

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

References

1. Edwards, C.J. *The Historical Development of the Calculus*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
2. Oldham, K.; Spanier, J. *The Fractional Calculus Theory and Applications of Differentiation and Integration to Arbitrary Order*; Academic Press: Cambridge, MA, USA, 1974.
3. Hahn, D.W.; Özisik, M.N. *Heat Conduction*; John Wiley & Sons: Hoboken, NJ, USA, 2012.
4. Matlob, M.A.; Jamali, Y. The concepts and applications of fractional order differential calculus in modeling of viscoelastic systems: A primer. *Crit. Rev. Biomed. Eng.* **2019**, *47*, 249–276. [CrossRef] [PubMed]
5. Dehestani, H.; Ordokhani, Y.; Razzaghi, M. Fractional-order Legendre–Laguerre functions and their applications in fractional partial differential equations. *Appl. Math. Comput.* **2018**, *336*, 433–453. [CrossRef]
6. Yuxiao, K.; Shuhua, M.; Yonghong, Z. Variable order fractional grey model and its application. *Appl. Math. Model.* **2021**, *97*, 619–635. [CrossRef]
7. Kuang, Z.; Sun, L.; Gao, H.; Tomizuka, M. Practical fractional-order variable-gain supertwisting control with application to wafer stages of photolithography systems. *IEEE/ASME Trans. Mechatronics* **2021**, *27*, 214–224. [CrossRef]
8. Liu, C.; Yi, X.; Feng, Y. Modelling and parameter identification for a two-stage fractional dynamical system in microbial batch process. *Nonlinear Anal. Model. Control* **2022**, *27*, 350–367. [CrossRef]
9. Wang, S.; Li, W.; Liu, C. On necessary optimality conditions and exact penalization for a constrained fractional optimal control problem. *Optim. Control Appl. Methods* **2022**, *43*, 1096–1108. [CrossRef]
10. Bhrawy, A.H.; Ezz-Eldien, S.S.; Doha, E.H.; Abdelkawy, M.A.; Baleanu, D. Solving fractional optimal control problems within a Chebyshev–Legendre operational technique. *Int. J. Control* **2017**, *90*, 1230–1244. [CrossRef]
11. Saxena, S. Load frequency control strategy via fractional-order controller and reduced-order modeling. *Int. J. Electr. Power Energy Syst.* **2019**, *104*, 603–614. [CrossRef]
12. Mohammadi, H.; Kumar, S.; Rezapour, S.; Etemad, S. A theoretical study of the Caputo–Fabrizio fractional modeling for hearing loss due to Mumps virus with optimal control. *Chaos Solitons Fractals* **2021**, *144*, 110668. [CrossRef]
13. Aldrich, C. *Exploratory Analysis of Metallurgical Process Data with Neural Networks and Related Methods*; Elsevier: Amsterdam, The Netherlands, 2002.
14. Bhattacharya, U.; Parui, S.K. Self-adaptive learning rates in backpropagation algorithm improve its function approximation performance. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; IEEE: Piscataway, NJ, USA, 1995; Volume 5, pp. 2784–2788.
15. Niu, H.; Chen, Y.; West, B.J. Why do big data and machine learning entail the fractional dynamics? *Entropy* **2021**, *23*, 297. [CrossRef]
16. Xu, C.; Liao, M.; Li, P.; Guo, Y.; Xiao, Q.; Yuan, S. Influence of multiple time delays on bifurcation of fractional-order neural networks. *Appl. Math. Comput.* **2019**, *361*, 565–582. [CrossRef]
17. Pakdaman, M.; Ahmadian, A.; Effati, S.; Salahshour, S.; Baleanu, D. Solving differential equations of fractional order using an optimization technique based on training artificial neural network. *Appl. Math. Comput.* **2017**, *293*, 81–95. [CrossRef]
18. Asgharnia, A.; Jamali, A.; Shahnazi, R.; Maheri, A. Load mitigation of a class of 5-MW wind turbine with RBF neural network based fractional-order PID controller. *ISA Trans.* **2020**, *96*, 272–286. [CrossRef] [PubMed]
19. Fei, J.; Wang, H.; Fang, Y. Novel neural network fractional-order sliding-mode control with application to active power filter. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *52*, 3508–3518. [CrossRef]
20. Zhang, Y.; Xiao, M.; Cao, J.; Zheng, W.X. Dynamical bifurcation of large-scale-delayed fractional-order neural networks with hub structure and multiple rings. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *52*, 1731–1743. [CrossRef]

21. Cao, J.; Stamov, G.; Stamova, I.; Simeonov, S. Almost periodicity in impulsive fractional-order reaction–diffusion neural networks with time-varying delays. *IEEE Trans. Cybern.* **2020**, *51*, 151–161. [CrossRef] [PubMed]
22. Bao, C.; Pu, Y.; Zhang, Y. Fractional-order deep backpropagation neural network. *Comput. Intell. Neurosci.* **2018**, *2018*, 7361628. [CrossRef] [PubMed]
23. Han, X.; Dong, J. Applications of fractional gradient descent method with adaptive momentum in BP neural networks. *Appl. Math. Comput.* **2023**, *448*, 127944. [CrossRef]
24. Prasad, K.; Krushna, B. Positive solutions to iterative systems of fractional order three-point boundary value problems with Riemann–Liouville derivative. *Fract. Differ. Calc.* **2015**, *5*, 137–150. [CrossRef]
25. Hymavathi, M.; Ibrahim, T.F.; Ali, M.S.; Stamov, G.; Stamova, I.; Younis, B.; Osman, K.I. Synchronization of fractional-order neural networks with time delays and reaction-diffusion terms via pinning control. *Mathematics* **2022**, *10*, 3916. [CrossRef]
26. Wu, Z.; Zhang, X.; Wang, J.; Zeng, X. Applications of fractional differentiation matrices in solving Caputo fractional differential equations. *Fractal Fract.* **2023**, *7*, 374. [CrossRef]