Research Article

Solving Fractional Differential Equations by Using Triangle Neural Network

Feng Gao ¹, Yumin Dong ², and Chunmei Chi ³

¹School of Science, Qingdao University of Technology, Qingdao 266033, China

²College of Computer and Information Science, Chongqing Normal University, Chongqing 401331, China

³School of Information and Control Engineering, Qingdao University of Technology, Qingdao 266033, China

Correspondence should be addressed to Yumin Dong; dym@cqu.edu.cn

Received 9 February 2021; Revised 23 March 2021; Accepted 29 March 2021; Published 14 April 2021

Academic Editor: Xinguang Zhang

Copyright © 2021 Feng Gao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this paper, numerical methods for solving fractional differential equations by using a triangle neural network are proposed. The fractional derivative is considered Caputo type. The fractional derivative of the triangle neural network is analyzed first. Then, based on the technique of minimizing the loss function of the neural network, the proposed numerical methods reduce the fractional differential equation into a gradient descent problem or the quadratic optimization problem. By using the gradient descent process or the quadratic optimization process, the numerical solution to the FDEs can be obtained. The efficiency and accuracy of the presented methods are shown by some numerical examples. Numerical tests show that this approach is easy to implement and accurate when applied to many types of FDEs.

1. Introduction

Fractional differential equations (FDEs) have been a hot topic in many scientific fields, such as dynamical system control theory, fluid flow, modelling in rheology, dynamic process of self-similar porous structure, diffusion transport similar to diffusion, electric network, and probability statistics [1–9]. These problems in science and engineering sometimes require us to get the solutions of various fractional differential equations. But as we know, it is difficult to find the exact solutions in most cases. So, we have to use numerical methods to solve fractional differential equations.

In the literature, some numerical methods for solving FDEs have been proposed, such as nonlinear functional analysis methods, including monotone iterative technique [10], topological degree theory [11], and fixed point theorem [12]. In addition, someone proposed the following numerical methods: random walk [13], Adomian decomposition method and variational iteration method [14], homotopy perturbation method [15–17], etc.

In recent years, some scholars try to use the neural network to solve differential equations [18–20]. Lagaris et al. [21] proposed an artificial neural network method for solving initial and boundary value problems. In their work, a trial solution is adopted and written as the sum of two parts. The first part satisfies the initial or boundary conditions and does not contain adjustable parameters while the construction of the second part does not affect the initial and boundary conditions. Then, the neural network is trained to satisfy the differential equation at many selected points. The question for this method is that it is difficult to construct the first part of the trial solution and this method cannot be applied to fractional partial differential equations.

Piscopo et al. [22] also introduced a method to find the numerical solutions of many types of differential equations. The proposed method does not depend on the trial solution and therefore has more flexibility in many cases. It can be used for solving many types of ODE and PDE. The two mentioned neural network techniques motivate us to develop more neural network methods to solve FDEs, but how to

get the fractional order of the neural network is a difficult problem.

To overcome this difficulty, in this work, we use a triangle base neural network as basis function to propose an alternative method called triangle neural network methods. This paper is organized as follows. In Section 2, we study the fractional derivative of the triangle base neural network and present the numerical method for solving many types of FDEs. In Section 3, we show the efficiency of the proposed method by some numerical examples. Section 4 is the conclusion.

2. Fractional Derivative of Triangle Neural Network and Numerical Algorithm

2.1. Ordinary Fractional Differential Equation. To solve the following fractional initial value problem (1) and boundary value problem (2),

$$\begin{cases} D^{(\alpha)}y = f(x, y), \\ y^{(k)}(0) = y_k, \quad k = 0, 1, \dots, m-1, \end{cases} \quad (1)$$

where $m = [\alpha]$:

$$\begin{cases} a(x)D^{(\alpha)}y + b(x)D^{(\beta)}y = f(x), \\ y(0) = y_0, y(l) = y_1, \end{cases} \quad (2)$$

where $a(x)$ and $b(x)$ are real functions, $1 < \alpha \leq 2$, $0 < \beta \leq 1$. $D^{(\alpha)}$ and $D^{(\beta)}$ are Caputo fractional derivative operators.

We consider the following triangle base neural network 1 (see Figure 1) to approximate the solution of problems (1) and (2), where w_j are weights for the neural networks and $C_j(x)$ are triangle base functions as the following:

$$C_j(x) = \begin{cases} \cos(jx), & j = 0, 1, 2, \dots, N, \\ \sin[(j-N)x], & j = N+1, N+2, \dots, 2N, \end{cases} \quad (3)$$

where $C_j(x)$ are activation function of neurons in the hidden layer of the above neural network and N is an integer and $x \in [0, \pi]$.

Let the weight matrix be $w = [w_0, w_1, \dots, w_{2N}]^T$ and the activation matrix be $C(x) = [c_0(x), c_1(x), \dots, c_{2N}(x)]^T$. The triangle base neural network can be written as

$$y = \sum_{j=0}^{2N} w_j C_j(x). \quad (4)$$

When this neural network is used to be the numerical solution of problem (1), the loss function is

$$J = \frac{1}{2} \sum_{k=1}^m e^2(t) = \frac{1}{2} \sum_{k=1}^m [D^{(\alpha)}y(x_t) - f(x_t, y(x_t))]^2. \quad (5)$$

FIGURE 1: Triangle neural network 1.

For problem (2), The loss function is

$$J = \frac{1}{2} \sum_{k=1}^m e^2(t) = \frac{1}{2} \sum_{k=1}^m [a(x_t)D^{(\alpha)}y(x_t) + b(x_t)D^{(\beta)}y(x_t) - f(x_t)]^2, \quad (6)$$

where $x_t, t = 1, 2, \dots, m$ are training points. We have two methods to minimize the loss function to get the corresponding numerical solution. One is the gradient descent algorithm, and another one is the optimization process. For both methods, we need to compute the α derivative of the triangle neural network. For this purpose, we have the following theorems.

Theorem 1. For given $\alpha \in R^+$, $f(x) \in C^1(R)$, then

$$D^{(\alpha)}f(\lambda x) = \lambda^\alpha f(u)|_{u=\lambda x}. \quad (7)$$

Proof. Since $D^{(\alpha)}f(x) = 1/\Gamma(1-\alpha) \int_0^x f'(\tau)/(x-\tau)^\alpha d\tau$.

We have

$$D^{(\alpha)}f(\lambda x) = \frac{1}{\Gamma(1-\alpha)} \int_0^x \frac{[f(\lambda\tau)]'}{(x-\tau)^\alpha} d\tau. \quad (8)$$

Let $\lambda\tau = u$, we have

$$\begin{aligned} D^{(\alpha)}f(\lambda x) &= \frac{1}{\Gamma(1-\alpha)} \int_0^{\lambda x} \frac{\lambda f'(u)}{(x-u/\lambda)^\alpha} du \\ &= \frac{1}{\Gamma(1-\alpha)} \int_0^{\lambda x} \frac{f'(u)}{(\lambda x-u)^\alpha \lambda^{-\alpha}} du \\ &= \frac{1}{\Gamma(1-\alpha)} \int_0^{\lambda x} \frac{\lambda^\alpha f'(u)}{(\lambda x-u)^\alpha} du \\ &= \frac{\lambda^\alpha}{\Gamma(1-\alpha)} \int_0^{\lambda x} \frac{f'(u)}{(\lambda x-u)^\alpha} du = \lambda^\alpha D^{(\alpha)}f(\lambda x). \end{aligned} \quad (9)$$

We also have the following.

Theorem 2. Given $\alpha \in R^+$, $\lambda, b \in R$, then

$$\begin{aligned} D^{(\alpha)}[\sin(\lambda x + b)] &= \lambda^\alpha \sin\left(\lambda x + b + \frac{\alpha\pi}{2}\right), \\ D^{(\alpha)}[\cos(\lambda x + b)] &= \lambda^\alpha \cos\left(\lambda x + b + \frac{\alpha\pi}{2}\right). \end{aligned} \quad (10)$$

Proof.

$$D^{(\alpha)}[\sin(\lambda x + b)] = D^{(\alpha)}[\sin(\lambda x) \cos b + \cos(\lambda x) \sin b]. \quad (11)$$

From Theorem 1, we have

$$\begin{aligned} &\cos b D^{(\alpha)} \sin(\lambda x) + \sin b D^{(\alpha)} \cos(\lambda x) \\ &= \cos b \lambda^\alpha \sin\left(\lambda x + \frac{\alpha\pi}{2}\right) + \sin b \lambda^\alpha \cos\left(\lambda x + \frac{\alpha\pi}{2}\right) \\ &= \lambda^\alpha \sin\left(\lambda x + b + \frac{\alpha\pi}{2}\right). \end{aligned} \quad (12)$$

The second part of this theorem can be verified in the same way. Based on Theorems 1 and 2, we can get the α derivative of the triangle base neural network.

In fact, let the solution to problems (1) and (2) be

$$\begin{aligned} y &= \sum_{j=0}^N w_j \cos jx + \sum_{j=0}^N w_{N+j} \sin jx \\ &= w_0 + \sum_{j=1}^N w_j \cos jx + \sum_{j=1}^N w_{N+j} \sin jx. \end{aligned} \quad (13)$$

We can get

$$D^{(\alpha)}y = \sum_{j=1}^N j^\alpha w_j \cos\left(jx + \frac{\alpha}{2}\pi\right) + \sum_{j=1}^N j^\alpha w_{N+j} \sin\left(jx + \frac{\alpha}{2}\pi\right). \quad (14)$$

Thus, we get the loss function for problem (1):

$$\begin{aligned} J &= \frac{1}{2} \sum_{k=1}^m e^2(t) = \frac{1}{2} \sum_{k=1}^m \left[D^{(\alpha)}y(x_t) - f(x_t, y(x_t)) \right]^2 \\ &= \frac{1}{2} \sum_{t=1}^m \sum_{j=1}^N \left[\left(j^\alpha w_j \cos\left(jx_t + \frac{\alpha}{2}\pi\right) \right. \right. \\ &\quad \left. \left. + j^\alpha w_{N+j} \sin\left(jx_t + \frac{\alpha}{2}\pi\right) \right) - f(x_t, y(x_t)) \right]^2. \end{aligned} \quad (15)$$

To carry out the gradient descent process, we have

$$\begin{aligned} \frac{\partial J}{\partial w_k} &= \left[\sum_{t=1}^m \sum_{j=0}^N j^\alpha \left(w_j \cos\left(jx_t + \frac{\alpha}{2}\pi\right) \right. \right. \\ &\quad \left. \left. + w_{N+j} \sin\left(jx_t + \frac{\alpha}{2}\pi\right) \right) - f(x_t, y) \right] \\ &\quad \cdot \left(k^\alpha \cos\left(kx_t + \frac{\alpha}{2}\pi\right) - f_y(x_t, y) \cos(kx_t) \right) \end{aligned} \quad (16)$$

for $j = 0, 1, 2, \dots, N$, and we also have

$$\begin{aligned} \frac{\partial J}{\partial w_k} &= \left[\sum_{t=1}^m \sum_{j=0}^N j^\alpha w_j \cos\left(jx_t + \frac{\alpha}{2}\pi\right) \right. \\ &\quad \left. + w_{N+j} \sin\left(jx_t + \frac{\alpha}{2}\pi\right) \right) - f(x_t, y) \right] \\ &\quad \cdot \left(k^\alpha \sin\left(kx_t + \frac{\alpha}{2}\pi\right) - f_y(x_t, y) \sin(k - N)x_t \right) \end{aligned} \quad (17)$$

for $j = N + 1, N + 2, \dots, 2N$.

So, we can see that getting the numerical solution of (1) is equivalent to finding w_j s by minimizing the loss function J . Usually, we have two methods to do this work. One is the gradient descent method, and another one is adopting the optimization process.

The gradient descent method is as below:

$$w_k^{(n+1)} = w_k^{(n)} - \eta \frac{\partial J}{\partial w_k}, \quad k = 1, 2, \dots, 2N, \quad (18)$$

where η is the step size for the gradient descent. If the function $f(x, y)$ is a linear function of y , the initial value problem can also be reduced to

$$\begin{aligned} \min \quad & J = \frac{1}{2} \sum_{k=1}^m \left[D^{(\alpha)}y_k - f(x_k, y_k) \right]^2 \\ \text{s.t.} \quad & y(0) = y_0. \end{aligned} \quad (19)$$

That is,

$$\begin{aligned} \min \quad & J = \frac{1}{2} \sum_{k=1}^m \left[D^{(\alpha)}y_k - f(x_k, y_k) \right]^2 \\ \text{s.t.} \quad & \sum_{i=0}^N w_i = y_0, \end{aligned} \quad (20)$$

which is a quadratic optimization problem. For fractional

FIGURE 2: Neural network 2.

boundary value problem (2), the numerical solution can be reduced to the following optimization process:

$$\begin{aligned} \min \quad & J = \frac{1}{2} \sum_{k=1}^m \left[a(x_k) D^{(\alpha)} y_k + b(x_k) D^{(\beta)} y_k - f(x_k) \right]^2 \\ \text{s.t.} \quad & \begin{cases} \sum_{i=1}^N w_i = y_0, \\ \sum_{i=0}^N \cos(il) w_i + \sin(N+i) l w_{N+i} = y_1. \end{cases} \end{aligned} \quad (21)$$

So, there are two methods to solve this problem. One method is to get the solution through the gradient descent method. Another method is using the optimization technique.

2.2. Fractional Partial Differential Equation. For fractional partial differential equation problem

$$\begin{cases} \frac{\partial^\alpha u(x, t)}{\partial t^\alpha} = a \frac{\partial^2 u(x, t)}{\partial x^2} + f(t, x), \\ u(0, t) = u_0(t), u(1, t) = u_1(t), u(x, 0) = v(x), \end{cases} \quad (22)$$

where $0 < \alpha \leq 1$, and problem

$$\begin{cases} \frac{\partial u(x, t)}{\partial t} = a \frac{\partial^\beta u(x, t)}{\partial x^\beta} + f(t, x), \\ u(0, t) = u_0(t), u(1, t) = u_1(t), u(x, 0) = v(x), \end{cases} \quad (23)$$

where $1 < \beta \leq 2$. We use the triangle base neural network (see Figure 2) to approximate the solution of problems (22) and (23). The triangle base neural network can be written as

$$u(x, y) = \sum_{i=1}^N w_i \sin(a_{i1}x + a_{i2}y - b_i), \quad (24)$$

where a_{i1}, a_{i2} are weights for the import layer in the neural network, b_i are bias parameters for the hidden layer in the

neural network, w_i are weights for the export layer in the neural network, and $\sin(x)$ is the activation function of neurons in the hidden layer.

Based on Theorems 1 and 2, we can get the fractional derivative of the neural network as below:

$$\begin{aligned} \frac{\partial^\alpha u(x, y)}{\partial x^\alpha} &= \sum_{i=1}^N w_i a_{i1}^\alpha \sin\left(a_{i1}x + a_{i2}y + b_i + \frac{\alpha}{2}\pi\right), \\ \frac{\partial^\beta u(x, y)}{\partial y^\beta} &= \sum_{i=1}^N w_i a_{i2}^\beta \sin\left(a_{i1}x + a_{i2}y + b_i + \frac{\alpha}{2}\pi\right). \end{aligned} \quad (25)$$

The loss function for problem (22) is

$$\min J = \frac{1}{2} \sum_{k=1}^m \left[\frac{\partial^\alpha u}{\partial t^\alpha} - \frac{\partial^2 u}{\partial x^2} - f(t_k, x_k) \right]^2 + \frac{1}{2} \sum_{(t_k, x_k) \in B} [u(t_k, x_k) - B(t_k, x_k)]^2, \quad (26)$$

where (t_k, x_k) are training points and B is the boundary of problem (22) and

$$\begin{aligned} B(0, t) &= u_0(t), \\ B(l, t) &= u_1(t), \\ B(x, 0) &= v(x). \end{aligned} \quad (27)$$

The loss function for problem (23) can be given in the same way.

We use the gradient descent algorithm to train the neural network. In fact, we can train the neural network by layers. First, we train the export layer to get w_i s, then the bias parameters to get b_i s in the hidden layer, and finally, we train the import layer to get a_{i1} s and a_{i2} s.

3. Numerical Experiment

3.1. Numerical Test 1. Consider the following example 1:

$$\begin{cases} D^{(\alpha)} y = \frac{1}{x^2 + 1} y + \frac{\Gamma(3)}{\Gamma(3 - \alpha)} x^{2-\alpha}, \\ y(0) = 1, \end{cases} \quad (28)$$

where $0 < \alpha \leq 1$. The exact solution to this problem is $y = x^2 + 1$. We let α be 0.5, 0.9 and use the optimization method when $N = 5, N = 15$ and gradient descent method when $N = 5, N = 15$, respectively. The computational results are listed in Tables 1 and 2.

3.2. Numerical Test 2. Consider the following example 2 for boundary value problem:

$$\begin{cases} D^{(1+\alpha)} y + D^{(\alpha)} y = \sqrt{2} \sin\left(x + \frac{\pi}{4}\right) \cos \frac{\alpha}{2} + \sqrt{2} \cos\left(x + \frac{\pi}{4}\right) \sin \frac{\alpha}{2}, \\ y(0) = 0, y\left(\frac{\pi}{2}\right) = 1, \end{cases} \quad (29)$$

TABLE 1: Computational errors for example 1 where $\alpha = 0.5, h = 0.1$.

x	1 h	2 h	3 h	4 h	5 h	6 h	7 h	8 h	9 h	10 h
optimi ($N = 5$)	0.010	0.013	0.015	0.016	0.018	0.019	0.021	0.022	0.024	0.026
optimi ($N = 15$)	0.003	0.003	0.008	0.009	0.012	0.014	0.013	0.014	0.015	0.016
grad, $N = 5$	0.011	0.013	0.017	0.019	0.025	0.021	0.020	0.019	0.027	0.030
grad, $N = 15$	0.008	0.008	0.007	0.004	0.005	0.009	0.012	0.018	0.015	0.013

TABLE 2: Computational errors for example 1 where $\alpha = 0.9, h = 0.1$.

x	1 h	2 h	3 h	4 h	5 h	6 h	7 h	8 h	9 h	10 h
optimi ($N = 5$)	0.000	0.010	0.010	0.011	0.012	0.014	0.018	0.017	0.018	0.020
optimi ($N = 15$)	0.000	0.002	0.006	0.008	0.010	0.010	0.011	0.010	0.012	0.012
grad, $N = 5$	0.009	0.013	0.012	0.014	0.021	0.020	0.020	0.016	0.021	0.021
grad, $N = 15$	0.005	0.006	0.003	0.002	0.007	0.012	0.013	0.010	0.010	0.010

FIGURE 3: Solution of Test 2 when $\alpha = 0.5$. FIGURE 4: Solution of Test 2 when $\alpha = 0.8$.

where $0 < \alpha \leq 1$. The exact solution to this problem is $y = \sin x$.

We let α be 0.5 and $N = 30, N = 45$, respectively; the computational error is listed in Figure 3.

We let α be 0.8 and $N = 20, N = 30$. The computational error is listed in Figure 4.

As we see in example 1, the solution becomes more accurate when N is increased. And for the boundary value problem, we use two constraints when we use the optimization process.

3.3. *Numerical Test 3.* Consider the following example 3:

$$\begin{cases} \frac{\partial^\alpha u(x, t)}{\partial t^\alpha} = -\frac{\Gamma(4)}{\Gamma(4-\alpha)} t^{-\alpha} \frac{\partial^2 u(x, t)}{\partial x^2}, \\ u(0, t) = u(\pi, t) = 0, u(x, 0) = 0, \end{cases} \quad (30)$$

where $0 < \alpha \leq 1$. The exact solution to this problem is $u = t^3 \sin x$.

We use the gradient descent method to solve this problem, and the computational error is listed in Figure 5. In the training process of the neural network, we set a stopping criteria $J < 10^{-5}$ for the computing process to stop. If this stopping criteria cannot be achieved, the computing will be stopped when 10^5 times of training is completed.

3.4. *Numerical Test 4.* Consider the following example 4:

$$\begin{cases} \frac{\partial u(x, t)}{\partial t} = \frac{\partial^\beta u(x, t)}{\partial x^\beta} - t^3 \sin \left(x + \frac{\beta}{2} \pi \right) + 3t^2 \sin x, \\ u(0, t) = u(\pi, t) = 0, u(x, 0) = 0, \end{cases} \quad (31)$$

where $1 < \beta \leq 2$. The exact solution to this problem is $u = t^3 \sin x$.

We use the gradient descent method to solve this problem, and the computational error is listed in Figure 6.

FIGURE 5: Solution of Test 3 when $\alpha = 0.8$. FIGURE 6: Solution of Test 4 when $\alpha = 0.8$.

4. Conclusion

The neural network method is a promising approach for solving fractional differential equations. The difficulty for this method is how to calculate the fractional derivatives of the involved neural network. In this paper, we propose numerical methods for solving fractional differential equations including the initial problem, boundary value problem, and partial FDEs by using the triangle base neural network and gradient descending method. All the involved fractional derivatives in this work are considered as Caputo type. We first analyze the fractional derivative of the triangle base neural network. Then, based on the loss function, the proposed numerical methods reduce the fractional differential equation into the gradient descent process or the quadratic optimization problem. By carrying out the gradient descent process or the quadratic optimization process, we can get

the numerical solutions. Numerical tests show that this approach is easy to implement and the solution is accurate when applied to many types of FDEs.

Data Availability

All the data supporting this study are available within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

All authors contributed equally and significantly to this paper. All authors read and approved the final manuscript.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (No. 61772295).

References

- [1] K. S. Miller, *An Introduction to Fractional Calculus and Fractional Differential Equations*, J. Wiley and Sons, New York, 1993.
- [2] K. Oldham and J. Spanier, "The fractional calculus," in *Theory and Applications of Differentiation and Integration of Arbitrary Order*, Academic Press, USA, 1974.
- [3] A. Kilbas, H. Srivastava, and J. Trujillo, *Theory and Applications of Fractional Differential Equations*, Math. Studies, North-Holland, New York, 2006.
- [4] I. Podlubny, *Fractional Differential Equations*, Academic Press, USA, 1999.
- [5] M. D. Ortigueira and J. A. Tenreiro Machado, "What is a fractional derivative?," *Journal of Computational Physics*, vol. 293, pp. 4–13, 2015.
- [6] R. Metzler and J. Klafter, "Boundary value problems for fractional diffusion equations," *Physica A*, vol. 278, no. 1-2, pp. 107–125, 2000.
- [7] Z. Odibat and S. Momani, "Numerical methods for nonlinear partial differential equations of fractional order," *Applied Mathematical Modelling*, vol. 32, no. 1, pp. 28–39, 2008.
- [8] Q. Yang, F. Liu, and I. Turner, "Numerical methods for fractional partial differential equations with Riesz space fractional derivatives," *Applied Mathematical Modelling*, vol. 34, no. 1, pp. 200–218, 2010.
- [9] Z. Liu and J. Liang, "A class of boundary value problems for first-order impulsive integro- differential equations with deviating arguments," *Journal of Computational and Applied Mathematics*, vol. 237, no. 1, pp. 477–486, 2013.
- [10] Y. Cui, Q. Sun, and X. Su, "Monotone iterative technique for nonlinear boundary value problems of fractional order $p \in (2, 3]$," *Advances in Difference Equations*, vol. 2017, no. 1, Article ID 248, 2017.
- [11] Z. Liu, N. V. Loi, and V. Obukhovskii, "Existence and global bifurcation of periodic solutions to a class of differential variational inequalities," *International Journal of Bifurcation and Chaos*, vol. 23, no. 7, article 1350125, 2013.
- [12] H. Qu and X. Liu, "Existence of non-negative solutions for a fractional m-point boundary value problem at resonance," *Boundary Value Problems*, vol. 2013, no. 1, Article ID 127, 2013.
- [13] R. Metzler and J. Klafter, "The random walk's guide to anomalous diffusion: a fractional dynamics approach," *Physics Reports*, vol. 339, no. 1, pp. 1–77, 2000.
- [14] I. Podlubny, A. Chechkin, T. Skovranek, Y. Chen, and B. M. Vinagre Jara, "Matrix approach to discrete fractional calculus II: partial fractional differential equations," *Journal of Computational Physics*, vol. 228, no. 8, pp. 3137–3153, 2009.
- [15] Z. Odibat, S. Momani, and H. Xu, "A reliable algorithm of homotopy analysis method for solving nonlinear fractional differential equations," *Applied Mathematical Modelling*, vol. 34, no. 3, pp. 593–600, 2010.
- [16] S. Das and P. K. Gupta, "Homotopy analysis method for solving fractional hyperbolic partial differential equations," *International Journal of Computer Mathematics*, vol. 88, no. 3, pp. 578–588, 2011.
- [17] A. Elsaied, "Homotopy analysis method for solving a class of fractional partial differential equations," *Communications in Nonlinear Science and Numerical Simulation*, vol. 16, no. 9, pp. 3655–3664, 2011.
- [18] P. Kadam, G. Datkhile, and V. A. Vyawahare, "Artificial neural network approximation of fractional-order derivative operators: analysis and DSP implementation," in *Fractional Calculus and Fractional Differential Equations*, Birkhäuser, Singapore, 2019.
- [19] A. A. S. Almarashi, "Approximation solution of fractional partial differential equations by neural networks," *Advances in Numerical Analysis*, vol. 2012, Article ID 912810, 10 pages, 2012.
- [20] H. Qu and X. Liu, "A numerical method for solving fractional differential equations by using neural network," *Advances in Mathematical Physics*, vol. 2015, Article ID 439526, 12 pages, 2015.
- [21] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, 1998.
- [22] M. L. Piscopo, M. Spannowsky, and P. Waite, "Solving differential equations with neural networks: applications to the calculation of cosmological phase transitions," *Physical Review D*, vol. 100, no. 1, article 016002, 2019.