Research Paper
Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. *, No. *, *, pp. 1-13
DOI:10.22034/cmde.2025.66444.3102

Solving fractional differential equations involving YAC operator using ANN

Manoj Kumar[1,*] **and Pranay Goswami**[2]

[1]School of Sciences, IILM University, Greater Noida, India.

[2]School of Liberal Studies, Ambedkar University Delhi, New Delhi, India.

Abstract

Most real-world phenomena, particularly in biological systems, are modeled using differential equations. However, certain classes of differential equations, especially nonlinear fractional differential equations (FDEs), pose significant analytical challenges. In this paper, we propose a novel Artificial Neural Network (ANN) framework for solving nonlinear FDEs based on the Yang-Abdel-Cattani (YAC) fractional derivative operator. The proposed method employs a truncated power series expansion as a trial solution, with unknown coefficients determined through an iterative optimization process guided by the ANN architecture. By integrating neural networks with fractional-order modeling, the proposed approach offers an efficient and accurate solution strategy for complex FDEs. Numerical results demonstrate that the method achieves higher accuracy and reliability compared to existing works [2].

Keywords. Fractional differential equations, Approximate numerical solutions, Neural networks for approximation, unsupervised machine learning.
2010 Mathematics Subject Classification. 65L05, 34K06, 34K28.

Uncorrected

1. Introduction

In recent decades, fractional integrals and derivatives have emerged as a significant topic in fractional calculus. The subject has a remarkable his- tory of applications across various fields of engineering, research [3, 6, 15, 33, 34] and many others. There has been substantial interest and ongoing developments in the study of fractional differential equations, or FDEs. There is no universally accepted definition of fractional differentiation [22]. Fractional-order differential equations are often challenging to solve analytically, leading to a growing interest among researchers in their approximations and numerical solutions [1, 4, 13, 19, 20, 25, 26]. As a result, developing robust and efficient methods for handling these equations is essential.

ANN-based approximation methods appear less sensitive to physical dimensions than conventional numerical techniques. ANN provides a flexible mesh, as it only solves the optimization problem without directly interacting with the mesh. Lagaris et al. [18] developed an ANN-based methodology for approximating the solution of initial or boundary value problems. A swarm technique inspired by ANN is used in [29] to solve a second-order perturbed delay Lane-Emden model that originated in astrophysics. In [32], a unique third-order perturbed delay differential model with two variants was constructed using the traditional Lane-Emden model, and the ANN approach is utilized to solve it. A neuro-structure is presented in [30] to solve the single variable boundary value problems. The authors presented a computational technique to solve a third-type nonlinear pantograph Lane-Emden differential equation in [31]. Using the ANN approach and genetic algorithm optimization, a singular delay differential equation was solved in [28]. To solve differential equations using artificial neural networks, a Python module called NeuroDiffEq is suggested in [7]. In [10], the author applied a feedforward neural network structure as a deep learning model to approximate the solution of a system of ordinary differential equations. Panghal and Kumar [24] employed an artificial neural network (ANN)

Received: 17 March 2025; Accepted: 26 January 2026.
∗ Corresponding author. Email: manojkumar7281@gmail.com.

architecture to approximate the solution of a system of delay ordinary differential equations. In our previous work [17], we used an ANN architecture to solve a system of Lane-Emden differential equations.

ANN-based techniques have also been extended to solve fractional-order differential equations. Several academics and experts have shown significant interest in using ANNs to solve FDEs in recent years [2, 8, 12, 16, 23, 27]. In [8], the author employed an ANN architecture to approximate the solution of fractional differential equations. In [2], we explored and utilized an ANN architecture to approximate the solution of singular and higher-order nonlinear fractional differential equations, finding better performance than the architecture used in [8]. Later, in [16], we proposed a multilayered neural network as a solver for fractional delay differential equations, achieving an absolute error as low as 10^{-5} compared to the exact solutions.

Artificial neural networks (ANNs) have gained significant attention due to their ability to model complex nonlinear relationships, handle errors robustly, and adapt to unfamiliar or intricate systems. Their resilience lies in capturing both quantitative and qualitative features within neurons and distributing them efficiently across the network. Moreover, ANNs utilize parallel distributed processing to accelerate large-scale computations. Fractional differential equations (FDEs) are widely used to model real-world phenomena, especially those involving memory and hereditary properties. While analytical methods can handle only simple FDEs, more complex and nonlinear variants pose serious challenges to both analytical and numerical techniques. To address this limitation, this study introduces a novel hybrid framework that combines the Yang–Abdel–Cattani (YAC) fractional derivative with artificial neural networks and a truncated power series method. This integration allows the ANN to determine series coefficients effectively through an iterative error minimization strategy, thereby solving challenging initial and boundary value problems with improved precision and flexibility. The key contributions of this study are summarized below

(1) Application of the Yang-Abdel-Cattani (YAC) fractional derivative to solve complex FDEs, including nonlinear and higher-order cases that are difficult to tackle with conventional methods.
(2) Development of a novel computational framework that integrates the truncated power series method with artificial neural networks, enhancing adaptability and learning efficiency.
(3) Implementation of an iterative error minimization technique to determine optimal series coefficients, enabling accurate solutions over a wide range of fractional problems.
(4) Demonstration of improved accuracy, robustness, and generalization compared to existing ANN-based or analytical approaches, thus extending the practical utility of ANNs in solving complex fractional models in applied science and engineering.

The remaining structure of the paper is organized as follows: Section 2 outlines the methodology, including preliminaries (definitions and properties of fractional derivatives), the problem statement (types of fractional differential equations), and the formulation of an ANN-based solution using a truncated series expansion. Section 3 presents numerical examples to illustrate the effectiveness of the method. Finally, section 4 concludes the study with a discussion of the results, limitations, and future directions of the approach.

Uncorrected Proof

2. Methodology

The overall process for solving the fractional differential equation is outlined in the flowchart shown in Figure 1. The methodology begins with defining the problem, including the formulation of the fractional differential equation and specifying appropriate initial or boundary conditions based on the Yang–Abdel–Cattani fractional derivative.

Following this, the domain is discretized to prepare the training data required for the model. The next steps involve designing the solution framework, initializing necessary parameters, and formulating a loss function that captures the residuals of the differential equation. An ANN architecture, specifically a Multi-Layer Perceptron Neural Network (MLPNN), is implemented to train the model by reducing the loss function.

Once training is complete, the model's accuracy is assessed using appropriate evaluation metrics or by comparing it with exact or numerical solutions. The final output is an approximate solution to the test case of the fractional differential equation, generated by the trained MLPNN model.

2.1. **Preliminaries.**

As an introduction to fractional calculus, this section starts with some basic notation and definitions.

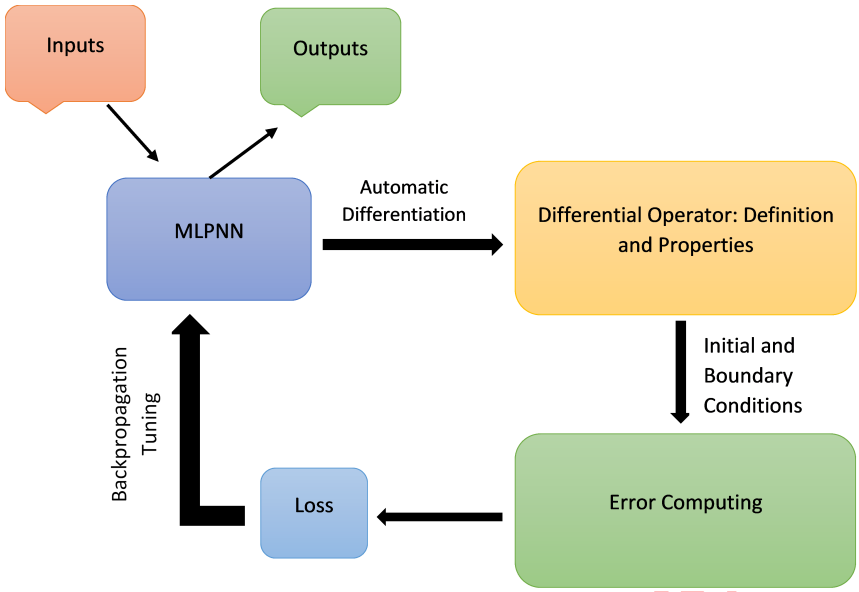


FIGURE 1. Flowchart of the methodology for solving fractional differential equations using MLPNN.

Definition 2.1. Let $h(t)$ be an absolutely continuous function over the interval $[a, b]$. The fractional integral of $h(t)$ is given by the definition[16]

$$ {}_a\mathcal{I}_t^{\varsigma} h(t) = \frac{1}{\Gamma(\varsigma)} \int_a^t \frac{h(\xi)}{(t-\xi)^{1-\varsigma}} d\xi, \varsigma > 0, \tag{2.1} $$

where ς denotes order of integration, $\Gamma(\cdot)$ is known Gamma function.

Definition 2.2. Let $h(t)$ be a continuous function over the interval $[a, b]$. The Caputo fractional derivatives of $h(t)$ are given by the definition[16]

$$ {}_a^c\mathcal{D}_t^{\varsigma} h(t) = \frac{1}{\Gamma(n-\varsigma)} \int_a^t \frac{h^n(\xi)}{(t-\xi)^{\varsigma-n+1}} d\xi, \tag{2.2} $$

where ς denotes order of derivative, n is a natural number and $n - \varsigma > 0$.

Definition 2.3. Let $h(t)$ be a continuous function over the interval $[a, b]$. The YAC fractional derivatives of $h(t)$ are given by the definition[35]

$$ {}_a^{yac}\mathcal{D}_t^{\varsigma} h(t) = \int_a^t \phi_{\varsigma}(-\lambda(t-\xi)^{\varsigma}) h'(\xi) d\xi, \tag{2.3} $$

where

$$ \phi_{\varsigma}(\lambda w^{\varsigma}) = \sum_{n=0}^{\infty} \frac{\lambda^n w^{(n+1)(\varsigma+1)-1}}{\Gamma[(n+1)(\varsigma+1)]}, \tag{2.4} $$

where ϕ_{ς} describes the exponential kernel of Rabotnov of fractional order ς.

The YAC fractional derivative of $h(t) = t^i$ has been computed using the following steps,

$$ {}_a^{yac}\mathcal{D}_t^{\varsigma} A = 0, \tag{2.5} $$

Uncorrected Pr

and

$$ {}_{a}^{yac}\mathcal{D}_{t}^{\varsigma}\left[t^{i}\right]=\begin{cases}0, & i<\lceil\varsigma\rceil,\\ \sum_{n=0}^{\infty}\frac{(-\lambda)^{n}\Gamma(i+1)}{\Gamma(i+\varsigma+1+n(\varsigma+1))}t^{(n+1)(\varsigma+1)+i}, & i\geq\lceil\varsigma\rceil,\end{cases} \tag{2.6}$$

where, i is a natural number, and A represents a constant. In the Caputo sense, a fractional derivative of $t^i, i \in N$ can be found in [2].

2.2. **Problem statement.** Let $y = y(t)$, we consider the fractional differential equation in YAC sense such as

$$P_1(t, y, {}_{a}^{yac}\mathcal{D}_{t}^{\varsigma}y) = P_2(t, y), a \leq t \leq b, \tag{2.7}$$

with initial conditions

$$y(t_0) = y_0, y'(t_0) = y_1, \ldots, y^{\lceil\varsigma\rceil-1}(t_0) = y_{\lceil\varsigma\rceil-1},$$

where ς denotes order of derivative, $P_1(\cdot)$ and $P_2(\cdot)$ are known analytical functions of real value. We applied an architecture of MLPNN to solve Eq. (2.7) as fractional differential equations in the YAC sense. A trial function, expressed as a series of truncated powers, is considered for the solution of Eq. (2.7), where the unknown coefficients in the series are replaced after MLPNN structure training. Subsequently, some FDEs were solved using the proposed ANN technique.

Consider a trial series such that

$$y(t) = y_0 + \sum_{i=1}^{n} a_i t^i, \tag{2.8}$$

where a_i are unknown coefficients that are computed through the proposed MLPNN structure. Then, we derive Eq.(2.7) as follows

$$P_1(t, y_0 + \sum_{i=1}^{n} a_i t^i, {}_{a}^{yac}\mathcal{D}_{t}^{\varsigma}[y_0 + \sum_{i=1}^{n} a_i t^i]) = P_2(t, y_0 + \sum_{i=1}^{n} a_i t^i), \tag{2.9}$$

Here, $t_i = a + ik$ represents the mesh points that are equally spaced on the t-axis in the interval $[a, b]$, with spacing $k = \frac{b-a}{N+1}$ and $i = 0, 1, \ldots, N + 1$.

This leads to the discretization method shown below

$$P_1(t, y_0 + \sum_{i=1}^{n} a_i t^i, {}_{a}^{yac}\mathcal{D}_{t}^{\varsigma}[y_0 + \sum_{i=1}^{n} a_i t^i]) \approx P_2(t, y_0 + \sum_{i=1}^{n} a_i t^i), \tag{2.10}$$

where n is the order of trial series polynomial.

2.3. **ANNs approach.** We applied a fully connected MLPNN structure consisting of an input, hidden, and output layer. The number of hidden layers and the number of neurons in each hidden layer were refined through hyperparameter tuning.

The general structure of the MLPNN is depicted in Figure 2, where each hidden neuron is represented by a mathematical formula as follows

$$\mathbf{t}^m = \sigma_m(\mathbf{w}^m.\mathbf{t}^{m-1} + \mathbf{b}_m), m = 1, 2, \ldots, M, \mathbf{t}^0 = \mathbf{t}, \tag{2.11}$$

$$a_i = \mathbf{w}.\mathbf{t}^M + \mathbf{b}, \tag{2.12}$$

where $\mathbf{t} \in \mathbb{R}^k$ is the input vector, represented as $(t_1, t_2, \ldots, t_k)$; $\mathbf{w}^m$ is the weight matrix of the m-th layer, represented as $(w_1^m, w_2^m, \ldots, w_k^m)$; b_m denotes the bias term; and $\sigma_m \colon \mathbb{R} \to [0, \infty)$ is the ReLU activation function for the m-th layer; M denotes the number of hidden layers.

To define the loss function, it is essential to calculate the derivatives of the variable with respect to the spatial variable at each discrete point within the domain. Afterward, the accuracy of the FDE solution approximation is evaluated by minimizing the cost function. The goal of the optimization process is to identify the optimal network

Uncorrected Proof

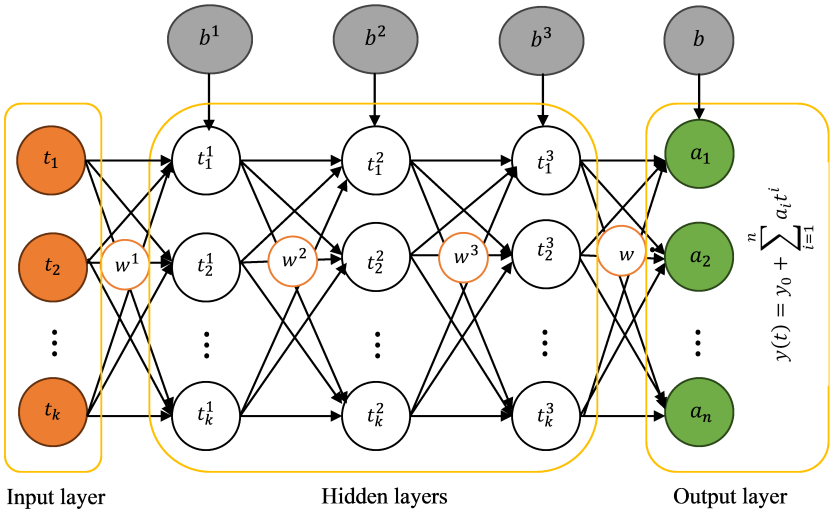


FIGURE 2. Structure of MLPNN with single input, output, and multiple hidden layers.

parameters. When MLPNN is employed to compute the numerical solution of the initial value problem for the fractional differential equation, the loss function for Eq. (2.10) is defined accordingly.

$$E_j = \frac{1}{2}\left(P_1(t_j, y_0 + \sum_{i=1}^{n} a_i t_j^i, \chi_{s,j}) - P_2(t_j, y_0 + \sum_{i=1}^{n} a_i t_j^i)\right)^2 + \lambda \|w\|_2^2, \tag{2.13}$$

Here, λ represents an artificial parameter, and L_2 is a regularization technique that helps to reduce the influence of outlier weights, pushing them closer to 0, though not entirely to 0. For simplicity, the previously mentioned mathematical symbols $\chi_{s,j}$ denotes fractional derivative in the YAC sense.

According to Eq. (2.13), the parameter a_i represents the trial series coefficients instructed using the neural network. Next, compute the value of loss function E_j using Eq. (2.13) at each discrete point of $t_j, j = 1, 2, \ldots, k$. The total error, or total cost, can then be determined as follows:

$$E = \sum_{j=0}^{k} E_j, \tag{2.14}$$

thus, the following optimization problem is formulated

$$\arg\min_{a_i} E = \sum_{j=0}^{k} \frac{1}{2}\left(\sum_{s=0}^{p} P_s(t_j, y_0 + \sum_{i=1}^{n} a_i t_j^i)\chi_{s,j} - f(t_j, y_0 + \sum_{i=1}^{n} a_i t_j^i)\right)^2 + \lambda \|w\|_2^2. \tag{2.15}$$

Figure 2 illustrates the neural network's architecture, with the hidden neurons in the hidden layers defined by Eq. (2.11). The coefficient a_i of the trial series is computed through forward propagation, as shown in Figure 2. By substituting a_i and point t_j on both sides of Eq. (2.10), we proceed to apply Eq. (2.13) and Eq. (2.15).

For each layer $m = 1, 2, \ldots, M$, the backpropagation process computes the partial derivatives of the total cost function E (as defined in Eq. (2.15)) with respect to the network parameters by applying the chain rule of calculus. The local error term for the output layer is:

$$\delta^{(M)} = \frac{\partial E}{\partial \mathbf{t}^{(M)}} \odot \sigma_M'(\mathbf{w}^M \mathbf{t}^{M-1} + \mathbf{b}_M),$$

Uncorrected Pro

and for each hidden layer $m = M-1, M-2, \ldots, 1$:

$$\delta^{(m)} = \left((\mathbf{w}^{(m+1)})^T \delta^{(m+1)}\right) \odot \sigma'_m(\mathbf{w}^m \mathbf{t}^{m-1} + \mathbf{b}_m).$$

The gradients of the cost function with respect to the network parameters are computed as:

$$\frac{\partial E}{\partial \mathbf{w}^{(m)}} = \delta^{(m)}(\mathbf{t}^{(m-1)})^T, \quad \frac{\partial E}{\partial \mathbf{b}^{(m)}} = \delta^{(m)}.$$

These relations explicitly describe how the gradients of the cost function are propagated backward through the network layers. The backpropagation algorithm performs effectively for relatively simple MLPNN architectures; however, its efficiency decreases when applied to more complex network structures. To address this limitation, automatic differentiation (AD) [5] is employed to perform backpropagation using the chain rule, enabling accurate and efficient computation of cost function gradients. The computed gradients are subsequently used to update the network parameters, including weights (w) and biases (b), via the Adam optimization algorithm. The mathematical formulation of the ADAM is provided in our previous work [17].

The current implementation follows the PyTorch framework, where the `loss.backward()` function computes the gradients and `torch.optim.Adam` performs the optimization. The Adam (Adaptive Moment Estimation) optimizer has demonstrated superior performance in practical applications, providing faster convergence and more stable learning compared to other adaptive optimization methods. In addition, Adam effectively mitigates common issues such as learning rate decay, slow convergence, and fluctuations in the cost function caused by high variance in parameter updates.

The ReLU activation function is selected for this network because of its numerous benefits. First, ReLU simplifies network training, as its derivative is more manageable to compute than functions like sigmoid or tanh. Second, it improves the non-linearity of the network, providing non-linear grid-fitting mapping when applied to neural networks. Third, it prevents the vanishing gradient problem, commonly occurring with sigmoid and tanh, whose derivatives approach zero for extreme values. Since ReLU is unbounded, it avoids this issue. Additionally, it can reduce overfitting by making the grid sparse, as values below zero become zero while those above retain positive values [11].

3. Numerical Simulations

Initially, we improved the structure of an artificial neural network (ANN) to approximate the solution of a fractional differential equation (FDE). The refinement process involved fine-tuning several key parameters, including the number of hidden layers, the number of neurons per hidden layer, and the total number of iterations. Python was utilized to implement these adjustments, with the resulting model structure tuning illustrated in Figure 3. The structure with a single hidden layer performs well with 17, 75, and 120 neurons, as illustrated in Figure 3(a). Beyond this, the model's performance was further explored by incorporating two or three hidden layers, combining the neurons 17, 75, and 120, as depicted in Figure 3. The analysis reveals an optimal structure with three hidden layers of 17, 120, and 17 neurons. This architecture demonstrates strong performance, achieving a minimum loss of 0.0024, making it a promising candidate for further investigation. The loss is calculated using Eq. (2.15), which requires setting the number of iterations. We trained the ANN architecture with various iteration counts to determine the optimal value, as shown in Figure 4. Through this process, we identified 5,500 iterations as the optimal choice for refining the architecture in future studies.

Eq. (2.15) is minimized using the Adam optimizer with a batch size 64. Adam is a first-order gradient-based optimizer and it is recognized for its reliability, computational efficiency, and effectiveness in training deep neural networks, particularly those with numerous parameters [14]. Achieving optimal results requires carefully tuning parameters. The refined model is implemented in the following FDEs to approximate the solution in the range [0, 1].

Uncorrected Proof

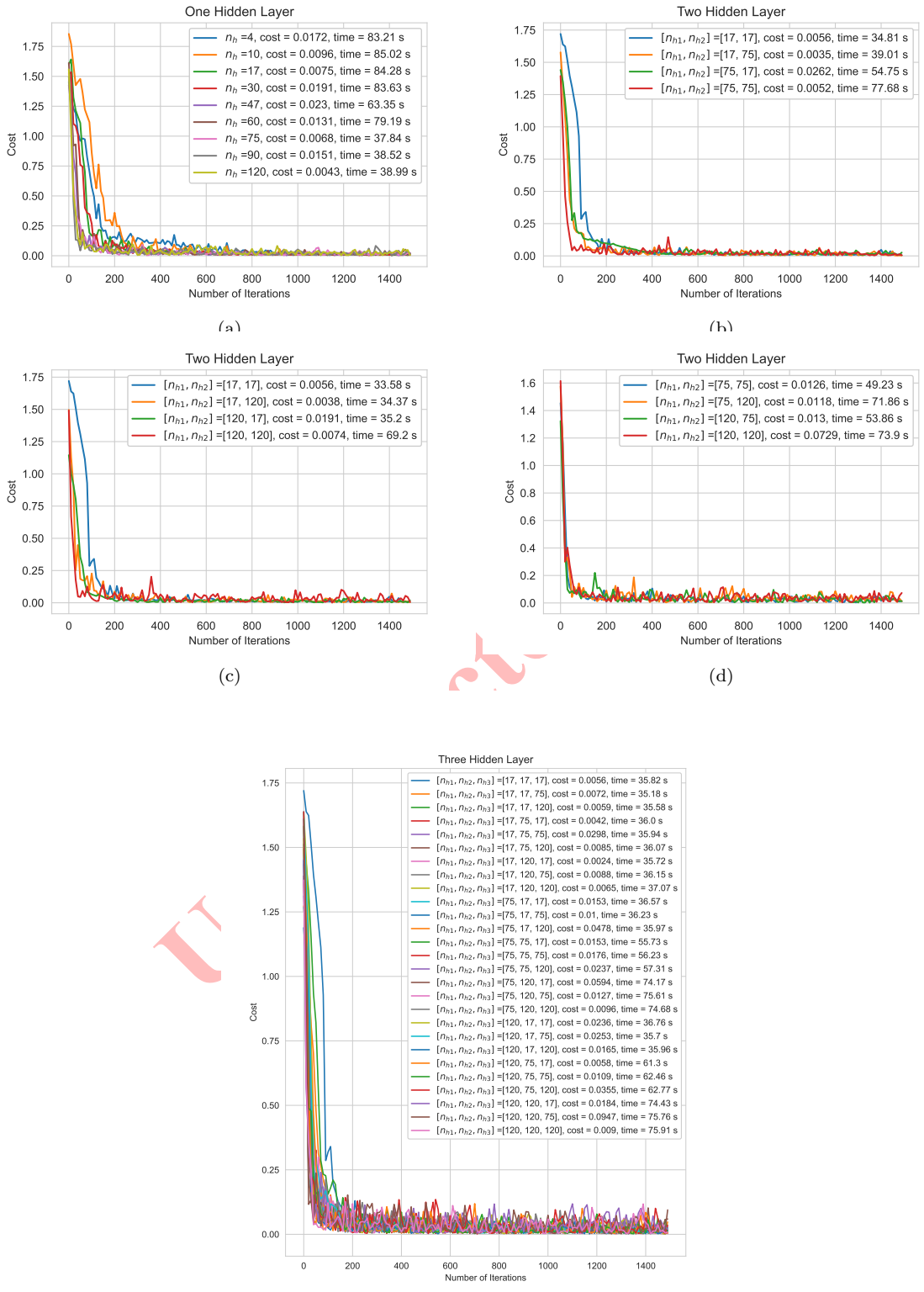


FIGURE 3. Tuning of the structure

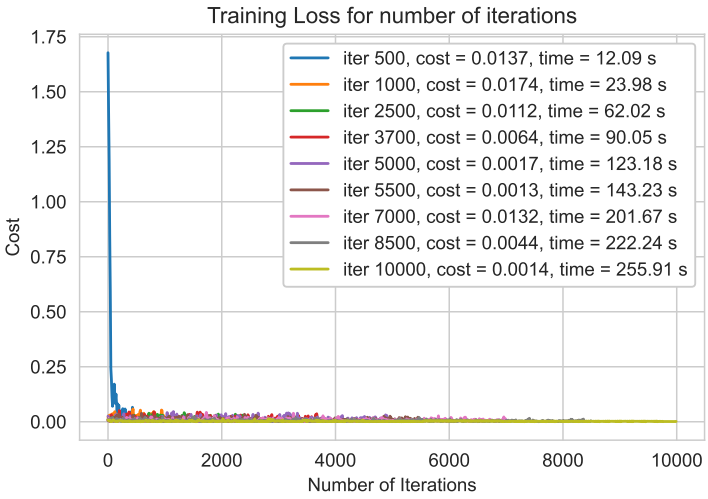


FIGURE 4. Tuning of the structure.

Example 3.1. Consider a Bagley–Torvik equation [21]

$$ {}_0^{yac}\mathcal{D}_t^2 y(t) + {}_0^{yac}\mathcal{D}_t^{1.5} y(t) = 1 + t, 0 < \nu \leq 1 \text{ and } 0 \leq t \leq 1 \tag{3.1} $$

with the initial conditions $y(0) = 1$ and $y'(0) = 1$.

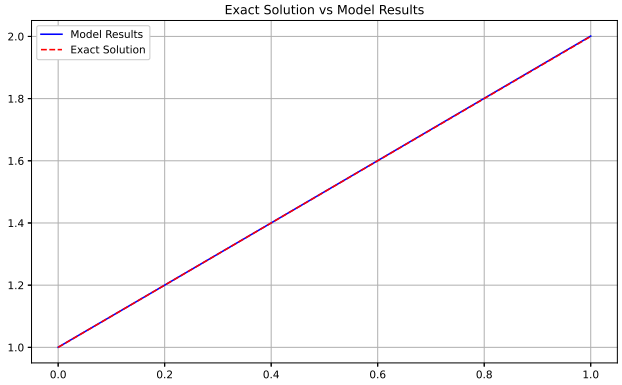
Table 1 presents the results for Example 1, including the exact solution $y(x)$, the approximate solution $\bar{y}$ obtained using the proposed method, and the corresponding absolute error $|y - \bar{y}|$ at various sample points t. For the numerical simulation, the proposed model architecture was trained for 5500 epochs. The detailed description of the architecture is provided in sections 3 and 2.3.

Among the methods compared, the proposed model yields approximate results ($\bar{y}$) with significantly higher accuracy than those reported in [2]. Figure 5 illustrates a visual comparison between the exact and approximate solutions, clearly showing that the proposed method closely tracks the exact solution, with both curves exhibiting strong alignment across the domain.

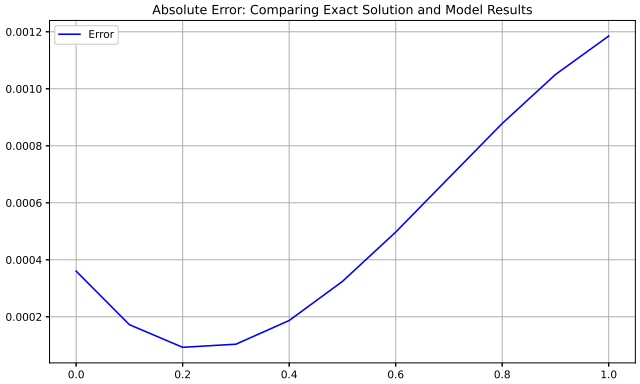
TABLE 1. Absolute error: comparing exact and approximation solutions.

t	y	$\bar{y}$[2]	$\bar{y}$	$\lvert y - \bar{y}\rvert$[2]	$\lvert y - \bar{y}\rvert$
0	1	0.999834	1.000360131	1.6600E-04	3.6013E-04
0.1	1.1	1.099184	1.100172292	8.1600E-04	1.7229E-04
0.2	1.2	1.198526	1.200092919	1.4740E-03	9.2919E-05
0.3	1.3	1.297862	1.300103866	2.1380E-03	1.0387E-04
0.4	1.4	1.397191	1.400186986	2.8090E-03	1.8699E-04
0.5	1.5	1.496719	1.500324132	3.2810E-03	3.2413E-04
0.6	1.6	1.596471	1.600497157	3.5290E-03	4.9716E-04
0.7	1.7	1.696224	1.700687913	3.7760E-03	6.8791E-04
0.8	1.8	1.795982	1.800878255	4.0180E-03	8.7825E-04
0.9	1.9	1.895748	1.901050034	4.2520E-03	1.0500E-03
1	2	1.995527	2.001185104	4.4730E-03	1.1851E-03

Uncorrected Proof



(a)



(b)

FIGURE 5. Proposed Model performance with the exact solution for Example 1

Example 3.2. Consider a nonlinear FDE of higher order [9]

$$ {}_0^{yac}\mathcal{D}_t^3 y(t) + {}_0^{yac}\mathcal{D}_t^{2.5} y(t) + y^2(t) = t^4, 0 < \nu \leq 1 \; and \; 0 \leq t \leq 1 \tag{3.2} $$

with $y(0) = 0$, $y'(0) = 0$, and $y''(0) = 2$.

Test Example 3.2 involves a nonlinear fractional differential equation (FDE) of higher order, distinguishing it from the earlier test case Example 3.1. For the numerical simulation, the proposed model architecture was trained for 5500 epochs. A detailed description of the model architecture is provided in sections 3 and 2.3.

As with the previous examples, the numerical results for Example 3.2 are summarized in Table 2 and visually illustrated in Figure 6.

Among the compared methods, the proposed model yields approximate results ($\bar{y}$) with significantly higher accuracy than those reported in [2]. Figure 6 provides a visual comparison between the exact and approximate solutions, clearly demonstrating that the proposed method closely approximates the exact solution across the entire domain.

TABLE 2. Absolute error: comparing exact and approximation solutions.

t	y	$\bar{y}$[2]	$\bar{y}$	$\lvert y-\bar{y}\rvert$[2]	$\lvert y-\bar{y}\rvert$
0	0	0.000072	0.000186801	7.2000E-05	1.8680E-04
0.1	0.01	0.010013	0.010207217	1.3000E-05	2.0722E-04
0.2	0.04	0.040106	0.040231928	1.0600E-04	2.3193E-04
0.3	0.09	0.090206	0.090263146	2.0600E-04	2.6315E-04
0.4	0.16	0.160314	0.160303086	3.1400E-04	3.0309E-04
0.5	0.25	0.250427	0.250353964	4.2700E-04	3.5396E-04
0.6	0.36	0.360546	0.360417994	5.4600E-04	4.1799E-04
0.7	0.49	0.490671	0.490497390	6.7100E-04	4.9739E-04
0.8	0.64	0.640801	0.640594367	8.0100E-04	5.9437E-04
0.9	0.81	0.810936	0.810711139	9.3600E-04	7.1114E-04
1	1	1.001075	1.000849920	1.0750E-03	8.4992E-04

4. CONCLUSION

This study proposed a Multi-Layer Perceptron Neural Network (MLPNN) framework to solve fractional differential equations (FDEs) within the context of the Yang-Abdel-Cattani (YAC) derivative. The performance of the developed ANN algorithm was evaluated using several benchmark FDEs. In the tested examples, the proposed model demonstrated higher accuracy compared to the method presented by [2], achieving satisfactory approximate solutions without the need for mesh discretization. The results obtained in this work showed an accuracy of up to 10^{-5} compared to the analytical solutions. However, a limitation of this approach is that it is currently applicable only to FDEs involving the YAC derivative. Despite this, the methodology can potentially be extended to other complex fractional differential problems. The main source code for this work is available on GitHub: https://github.com/amanojup/.

Uncorrected Proof

REFERENCES

[1] G. Agarwal, L. K. Yadav, K. S. Nisar, M. M. Alqarni, and E. E. Mahmoud, *A hybrid method for the analytical solution of time fractional Whitham–Broer–Kaup equations*, Applied and Computational Mathematics, *23*(1) (2024), 3–17.

[2] S. Althubiti, M. Kumar, P. Goswami, and K. Kumar, *Artificial neural network for solving the nonlinear singular fractional differential equations*, Applied Mathematics in Science and Engineering, *31*(1) (2023), 2187389.

[3] O. A. Arqub and M. Al-Smadi, *An adaptive numerical approach for the solutions of fractional advection–diffusion and dispersion equations in singular case under Riesz's derivative operator*, Physica A: Statistical Mechanics and its Applications, *540*(1) (2020), 123257.

[4] Z. Avazzadeh, O. Nikan, A. T. Nguyen, and V. T. Nguyen, *A localized hybrid kernel meshless technique for solving the fractional Rayleigh–Stokes problem for an edge in a viscoelastic fluid*, Engineering Analysis with Boundary Elements, *146*(1) (2023), 695–705.

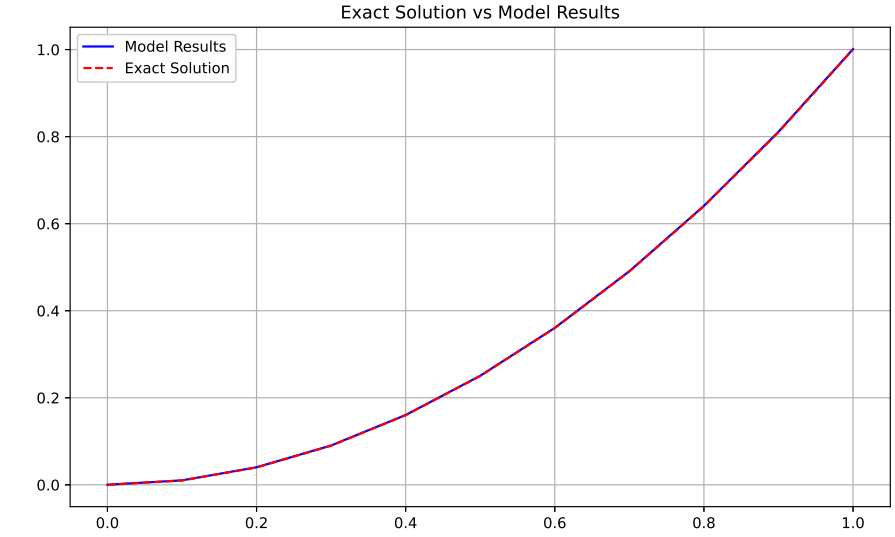
[5] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, *Automatic differentiation in machine learning: a survey*, Journal of Machine Learning Research, *18*(1) (2017), 5595–5637.

[6] E. Bazhlekova and I. Bazhlekov, *Viscoelastic flows with fractional derivative models: computational approach by convolutional calculus of Dimovski*, Fractional Calculus and Applied Analysis, *17*(4) (2014), 954–976.

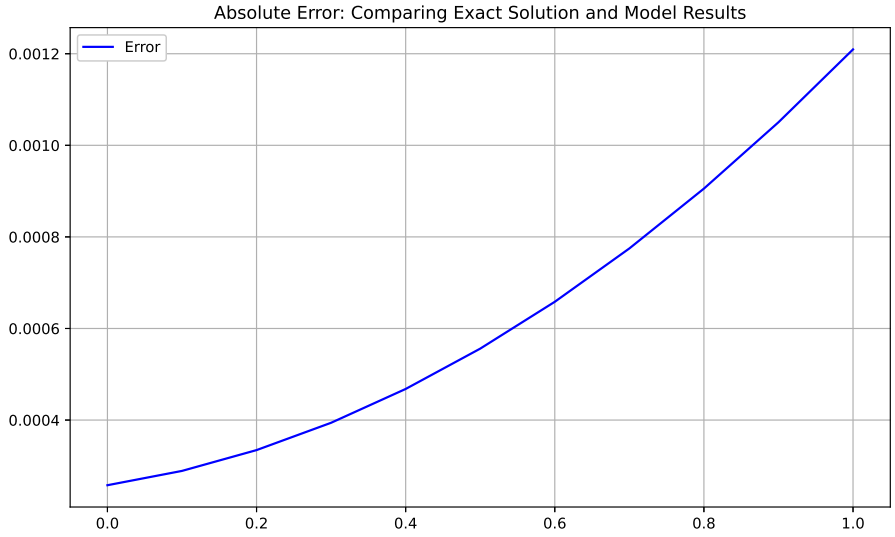
[7] F. Chen, D. Sondak, P. Protopapas, M. Mattheakis, S. Liu, D. Agarwal, and M. Di Giovanni, *NeuroDiffEq: A Python package for solving differential equations with neural networks*, Journal of Open Source Software, *5*(46) (2020), 1931.

[8] P. Dai and X. Yu, *An artificial neural network approach for solving space fractional differential equations*, Symmetry, *14*(3) (2022), 535.

[9] A. B. Deshi and G. A. Gudodagi, *Numerical solution of Bagley–Torvik, nonlinear and higher order fractional differential equations using Haar wavelet*, SeMA Journal, *79*(4) (2022), 663–675.



(a)



(b)

FIGURE 6. Proposed Model performance with the exact solution for Example 2

[10] T. T. Dufera, *Deep neural network for system of ordinary differential equations: vectorized algorithm and simulation*, Machine Learning with Applications, *5*(1) (2021), 100058.

[11] X. Glorot, A. Bordes, and Y. Bengio, *Deep sparse rectifier neural networks*, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, (2011), 315–323.
[12] A. H. Hadian-Rasanan, D. Rahmati, S. Gorgin, and K. Parand, *A single layer fractional orthogonal neural network for solving various types of Lane–Emden equation*, New Astronomy, *75*(1) (2020), 101307.
[13] E. Karapınar, R. Sevinik-Adıguzel, W. Aksoy, and I. M. Erhan, *A new approach to the existence and uniqueness of solutions for a class of nonlinear q-fractional boundary value problems*, Applied and Computational Mathematics, *24*(2) (2025), 235–249.
[14] D. P. Kingma and J. L. Ba, *Adam: A method for stochastic optimization*, International Conference on Learning Representations (ICLR), (2015).
[15] M. Kumar and K. Kumar, *Bi-state prediction network model for mixed traffic congestion prediction*, International Journal of Computational Intelligence and Applications, *24*(5) (2024), 2450011.
[16] M. Kumar, S. Kumar, K. Kumar, and P. Goswami, *Multilayered neural network for power series-based approximation of fractional delay differential equations*, Mathematical Methods in the Applied Sciences, *48*(1) (2024), 1–15.
[17] A. Kumar, M. Kumar, and P. Goswami, *Numerical solution of coupled system of Emden–Fowler equations using artificial neural network technique*, An International Journal of Optimization and Control: Theories & Applications (IJOCTA), *14*(1) (2024), 62–73.
[18] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, *Neural-network methods for boundary value problems with irregular boundaries*, IEEE Transactions on Neural Networks, *11*(5) (2000), 1041–1049.
[19] M. Lakestani, R. Ghasemkhani, and T. Allahviranloo, *Solving a system of fractional Volterra integro-differential equations using cubic Hermite spline functions*, Computational Methods for Differential Equations, *13*(3) (2025), 980–994.
[20] M. Luo, W. Qiu, O. Nikan, and Z. Avazzadeh, *Second-order accurate, robust and efficient ADI Galerkin technique for the three-dimensional nonlocal heat model arising in viscoelasticity*, Applied Mathematics and Computation, *440*(1) (2023), 127655.
[21] S. Momani and Z. Odibat, *Numerical comparison of methods for solving linear differential equations of fractional order*, Chaos, Solitons & Fractals, *31*(5) (2007), 1248–1255.
[22] E. C. de Oliveira and J. A. Machado, *A review of definitions for fractional derivatives and integral*, Mathematical Problems in Engineering, *2014*(1) (2014), 1–7.
[23] M. Pakdaman, A. Ahmadian, S. Effati, S. Salahshour, and D. Baleanu, *Solving differential equations of fractional order using an optimization technique based on training artificial neural network*, Applied Mathematics and Computation, *293*(1) (2017), 81–95.
[24] S. Panghal and M. Kumar, *Neural network method: delay and system of delay differential equations*, Engineering with Computers, *38*(Suppl 3) (2022), 2423–2432.
[25] C. Park, H. Rezaei, and M. H. Derakhshan, *An effective method for solving the multi time-fractional telegraph equation of distributed order based on the fractional order Gegenbauer wavelet*, Applied and Computational Mathematics, *24*(1) (2025), 16–37.
[26] W. Qiu, O. Nikan, and Z. Avazzadeh, *Numerical investigation of generalized tempered-type integrodifferential equations with respect to another function*, Fractional Calculus and Applied Analysis, *26*(6) (2023), 2580–2601.
[27] F. Rostami and A. Jafarian, *A new artificial neural network structure for solving high-order linear fractional differential equations*, International Journal of Computer Mathematics, *95*(1) (2018), 528–539.
[28] Z. Sabir, H. A. Wahab, T. G. Nguyen, G. C. Altamirano, F. Erdoğan, and M. R. Ali, *Intelligent computing technique for solving singular multi-pantograph delay differential equation*, Soft Computing, *27*(1) (2022), 1–13.
[29] Z. Sabir, S. B. Said, Q. Al-Mdallal, and M. R. Ali, *A neuro swarm procedure to solve the novel second order perturbed delay Lane–Emden model arising in astrophysics*, Scientific Reports, *12*(1) (2022), 1–15.
[30] Z. Sabir, T. Botmart, M. A. Z. Raja, W. Weera, and F. Erdoğan, *A stochastic numerical approach for a class of singular singularly perturbed system*, PLOS ONE, *17*(11) (2022), e0277291.
[31] Z. Sabir, M. A. Z. Raja, M. R. Ali, and R. Sadat, *An advance computational intelligent approach to solve the third kind of nonlinear pantograph Lane–Emden differential system*, Evolving Systems, *14*(3) (2023), 393–412.

Uncorrected Proof

[32] Z. Sabir and S. B. Said, *Heuristic computing for the novel singular third order perturbed delay differential model arising in thermal explosion theory*, Arabian Journal of Chemistry, *16*(3) (2023), 104509.
[33] T. P. Stefański and J. Gulgowski, *Signal propagation in electromagnetic media described by fractional-order models*, Communications in Nonlinear Science and Numerical Simulation, *82*(1) (2020), 105029.
[34] X. Su, W. Xu, W. Chen, and H. Yang, *Fractional creep and relaxation models of viscoelastic materials via a non-Newtonian time-varying viscosity: physical interpretation*, Mechanics of Materials, *140*(1) (2020), 103222.
[35] X.-J. Yang, M. Abdel-Aty, and C. Cattani, *A new general fractional-order derivative with Rabotnov fractional-exponential kernel applied to model the anomalous heat transfer*, Thermal Science, *23*(3A) (2019), 1677–1681.

Uncorrected Proof