Neural Variable-Order Fractional Differential Equation Networks

Wenjun Cui[*13], **Qiyu Kang**[*2], **Xuhao Li**[4], **Kai Zhao**[5], **Wee Peng Tay**[5], **Weihua Deng**[6], **Yidong Li**[†13]

[1] Key Laboratory of Big Data & Artificial Intelligence in Transportation (Beijing Jiaotong University)
[2] University of Science and Technology of China
[3] Beijing Jiaotong University
[4] Anhui University
[5] Nanyang Technological University
[6] Lanzhou University

Abstract

Neural differential equation models have garnered significant attention in recent years for their effectiveness in machine learning applications. Among these, fractional differential equations (FDEs) have emerged as a promising tool due to their ability to capture memory-dependent dynamics, which are often challenging to model with traditional integer-order approaches. While existing models have primarily focused on constant-order fractional derivatives, variable-order fractional operators offer a more flexible and expressive framework for modeling complex memory patterns. In this work, we introduce the Neural Variable-Order Fractional Differential Equation network (NvoFDE), a novel neural network framework that integrates variable-order fractional derivatives with learnable neural networks. Our framework allows for the modeling of adaptive derivative orders dependent on hidden features, capturing more complex feature-updating dynamics and providing enhanced flexibility. We conduct extensive experiments across multiple graph datasets to validate the effectiveness of our approach. Our results demonstrate that NvoFDE outperforms traditional constant-order fractional and integer models across a range of tasks, showcasing its superior adaptability and performance.

1 Introduction

The intersection of differential equations and machine learning has opened a new frontier for developing algorithms that benefit both communities. Machine learning, for example, has been adeptly applied to solve high-dimensional partial differential equations (PDEs), particularly valuable in ill-posed and inverse problems where traditional numerical methods falter. Notable examples include PINNs and their variants (Raissi, Perdikaris, and Karniadakis 2019; Zhu et al. 2019), which leverage both measurement data and PDE constraints through automatic differentiation, as enabled by modern platforms like TensorFlow (Abadi et al. 2016) and PyTorch (Paszke et al. 2019). This progress has wide-ranging applications in areas such as quantum chemistry (Pfau et al. 2020), materials science (Shukla et al. 2020), geophysics (Zhu et al. 2021), and molecular simulations (Zhang et al. 2018). Conversely, differential equations have enriched machine learning (Weinan 2017), exemplified by integer-order neural ordinary differential equations (ODEs) (Chen et al. 2018) that model continuous residual layers. This integration has spurred enhancements in neural network performance (Dupont, Doucet, and Teh 2019; Dai et al. 2024), gradient stability (Haber and Ruthotto 2017; Gravina, Bacciu, and Gallicchio 2022), and robustness (Yan et al. 2018; Kang et al. 2021; Cui et al. 2023). Moreover, recent applications of stochastic differential equations in machine learning include generating data from noise, modeling intricate financial dynamics and enabling uncertainty quantification via Bayesian approaches (Song et al. 2021; Xu et al. 2022).

Fractional differential operators have shown considerable promise in describing real-world phenomena more effectively than their integer-order counterparts. Unlike traditional integer-order differential equations, fractional differential equations (FDEs) provide a way to incorporate a continuum of past states into the present state, offering a rich and flexible modeling framework. FDEs have found applications across diverse fields, including material science (Coleman and Noll 1961), signal processing (Machado, Kiryakova, and Mainardi 2011), finance (Scalas, Gorenflo, and Mainardi 2000), and porous and fractal phenomena modeling (Nigmatullin 1986). In machine learning, fractional derivatives have been used to optimize neural network parameters (Liu et al. 2022), diverging from the conventional integer-order derivatives employed in algorithms such as SGD or Adam (Kingma and Ba 2014). Moreover, fractional calculus has been applied to enhance densely connected graph neural networks (GNNs), addressing issues like non-smooth data and the vanishing gradient problem (Antil et al. 2020). Recently, (Nobis et al. 2023) proposes generative fractional diffusion models that achieve greater pixel-wise diversity and improved image quality. Additionally, FDE-based GNNs, employing fractional diffusion and oscillator mechanisms to propagate information over graphs, have demonstrated superior capabilities in graph representation learning and task performance compared to their integer-order counterparts (Kang et al. 2024a,b).

Existing work on employing FDEs within machine learning has predominantly focused on *constant-order* fractional

[*]These authors contributed equally.
[†]Corresponding author (ydli@bjtu.edu.cn).

derivatives, which assume the order remains fixed over time. This assumption often falls short of capturing the varying impact of historical information across different contexts and systems. Recent advancements have introduced *variable-order* fractional differential equations (Samko and Ross 1993; Coimbra 2003), which allow the order α of the fractional-order derivative $\frac{\mathrm{d}^\alpha}{\mathrm{d}t^\alpha}$ to vary over time. This generalization has proved essential for modeling a range of real-world phenomena in science and engineering (Coimbra 2003; Obembe, Hossain, and Abu-Khamsin 2017). In such systems, the order of the differential equation may depend on specific variables like temperature, concentration, or density. For example, (Glöckle and Nonnenmacher 1995) found the differential order governing protein relaxation varies with temperature changes. Similarly, (Meng et al. 2016) illustrated that a variable-order fractional viscoelastic model more precisely describes the time-dependent evolution of mechanical properties in viscoelastic materials. Work that combines variable-order FDEs with neural networks has been more related to computational neuroscience (Anastasio 1994) or Hopfield networks (Kaslik and Sivasundaram 2012), focusing on numerical simulations and the analysis of bifurcation and stability behavior within connected networks (Yang et al. 2018; Xu et al. 2019). Additionally, researchers have applied neural networks to solve variable-order FDEs, with notable examples found in (Lasaki, Ebrahimi, and Ilie 2023).

Building upon these developments, our work introduces the Neural Variable-Order Fractional Differential Equation network (NvoFDE), a novel framework that integrates the flexibility of VoFDEs with the expressive power of learnable neural networks. In contrast to previous approaches like (Chen et al. 2018; Kang et al. 2024a) that utilize a constant derivative order α, NvoFDE allows the order to be of the form $\alpha(t, \mathbf{x})$, which varies according to the time and hidden features. This adaptation offers a powerful tool for modeling non-uniform memory effects, enabling the neural network to adjust its sensitivity to preceding hidden features, which could enhance task performance. To validate our framework, we conduct extensive experiments across multiple graph datasets and benchmark NvoFDE against traditional fixed-order models. Our results demonstrate that NvoFDE not only adapts more effectively to varying datasets but also delivers better performance.

Our main contributions are summarized as follows:

- We propose a generalized differential equation-driven neural network framework named NvoFDE that incorporates a variable-order differential operator $\frac{\mathrm{d}^{\alpha(t,\mathbf{x})}}{\mathrm{d}t^{\alpha(t,\mathbf{x})}}$. This framework extends the prior classes of constant integer- and fractional-order neural differential equations, subsuming them as special cases with $\alpha(t, \mathbf{x}) \equiv \alpha$ being a constant integer or a constant positive real value, respectively. The order $\alpha(t, \mathbf{x}(t))$ is learnable and dependent on both time t and the hidden feature $\mathbf{x}(t)$. Our approach enables flexible and learnable hidden feature updating dynamics stemming from the adaptive differential operator throughout the updating process.
- We apply NvoFDE to graph neural networks, extending the capabilities of existing constant-order fractional GNNs. We demonstrate the effectiveness of our framework across multiple graph datasets, showcasing its superior adaptability and performance.
- Our application of NvoFDE introduces a dynamic approach for solving variable-order FDEs, where the system's order is learnable from empirical observations. This enhancement improves the model's adaptability and predictive accuracy in complex scenarios.

The structure of this paper is as follows: We provide a comprehensive overview of related research in the Appendix. In Section 2, we review the mathematical foundations of fractional calculus, underscoring the shift from constant to variable orders and its significance for dynamic system modeling. In Section 3, we describe the architecture of NvoFDE, highlighting its unique features and the reasoning behind its design. Section 4 includes our experimental setup, results, and a detailed analysis of our findings. The implementation code is available at https://github.com/cuiwjTech/AAAI2025_NvoFDE. Finally, we conclude the paper and discuss the potential implications of our work for future research in Section 5.

2 Preliminaries

In this section, we offer a concise overview of fractional calculus. Throughout this paper, we assume that all necessary conditions are met to ensure the well-posedness of the formulations discussed.

Cauchy Formula for Integer Order Integration: We define the operator J_{t_0} as mapping a function f, Riemann integrable over $[t_0, t_1]$, to its integral from t_0 to t: $J_{t_0} f(t) \coloneqq \int_{t_0}^{t} f(\tau)\mathrm{d}\tau$, for every $t \in [t_0, t_1]$. For $n \in \mathbb{N}$, the notation $J_{t_0}^n$ represents the n-fold iteration of J_{t_0} i.e. we set $J_{t_0}^n \coloneqq J_{t_0} J_{t_0}^{n-1}$ with $J_{t_0}^1 \coloneqq J_{t_0}$. According to (Diethelm 2010)[Lemma 1.1], induction can be employed to show equivalently that:

$$J_{t_0}^n f(t) = \frac{1}{(n-1)!} \int_{t_0}^{t} (t-\tau)^{n-1} f(\tau)\mathrm{d}\tau, \quad n \in \mathbb{N}. \quad (1)$$

Riemann-Liouville Fractional Integral Operator: Let $\alpha \in \mathbb{R}_+$. By extending $\mathbb{N}$ in (1) to $\mathbb{R}_+$, the Riemann-Liouville fractional integral operator of order α is naturally defined as

$$J_{t_0}^\alpha f(t) := \frac{1}{\Gamma(\alpha)} \int_{t_0}^{t} (t-\tau)^{\alpha-1} f(\tau)\mathrm{d}\tau, \quad \alpha \in \mathbb{R}_+, \quad (2)$$

where $\Gamma(\cdot)$ is the Gamma function. When $\alpha = 0$, we define $J_{t_0}^0 \coloneqq I$, the identity operator. The generalized variable-order Riemann-Liouville time integration operator whose fractional order varies with time is defined as

$$J_{t_0}^{\alpha(t)} f(t) \coloneqq \frac{1}{\Gamma(\alpha(t))} \int_{t_0}^{t} (t-\tau)^{\alpha(t)-1} f(\tau)\mathrm{d}\tau, \ \alpha(t) \in \mathbb{R}_+. \quad (3)$$

Here, $\alpha(t)$ represents the dynamically varying order, allowing for a flexible adjustment of the integration process to accommodate changes in the behavior or characteristics of the function $f(t)$ over time.

(Constant-Order) Caputo Fractional Derivative: The Caputo fractional derivative is particularly useful in various application fields because it maintains the same initial conditions as traditional integer order differential equations. The Caputo fractional derivative of a function $f(t)$ is defined as:

$$ {}_{t_0}D_t^{\alpha} f(t) \coloneqq J_{t_0}^{n-\alpha} f^{[n]} = \frac{1}{\Gamma(n-\alpha)} \int_{t_0}^{t} \frac{f^{[n]}(\tau)}{(t-\tau)^{\alpha-n+1}} d\tau, $$

where $n \in \mathbb{N}$ is such that $n-1 < \alpha \leq n$, and $f^{[n]}$ represents the n-th integer order derivative $\frac{\mathrm{d}^n f}{\mathrm{d}^n t}$. When $\alpha = n \in \mathbb{N}$, the Caputo fractional derivative simplifies to the classical derivative, i.e., ${}_{t_0}D_t^{\alpha} f(t) \equiv f^{[n]}$.

Notation Clarification: In Section 1, we use the fractional derivative notation $\frac{\mathrm{d}^{\alpha}}{\mathrm{d}t^{\alpha}}$ for readers who may not be familiar with the derivative ${}_{t_0}D_t^{\alpha}$. Moving forward, we will consistently use ${}_{t_0}D_t^{\alpha}$ to denote the fractional derivative to streamline our notation.

Remark 1. *From the definition of ${}_{t_0}D_t^{\alpha} f(t)$, it is evident that fractional derivatives incorporate the historical states of the function via the integral term, highlighting their non-local, memory-dependent nature. In contrast, the integer-order derivative only represents the local rate of change of the function.*

***Variable-Order* Caputo Fractional Derivative:** When the fractional order is allowed to vary with time, the generalized *variable-order* Caputo fractional derivative can be written as below

$$ {}_{t_0}D_t^{\alpha(t)} f(t) \coloneqq \frac{1}{\Gamma(n-\alpha(t))} \int_{t_0}^{t} \frac{f^{[n]}(\tau)}{(t-\tau)^{\alpha(t)-n+1}} d\tau, \quad (4) $$

with $n-1 < \alpha(t) \leq n$.

Compared to the constant-order Caputo fractional derivative, (4) employs a non-stationary power-law kernel to dynamically adjust its memory structure, influenced by previous values of the differentiation orders. Owing to these properties, the *variable-order* Caputo fractional derivative exhibits a unique memory feature, enabling it to accurately characterize complex physical systems and processes (Coimbra 2003; Obembe, Hossain, and Abu-Khamsin 2017). In more general settings, the $\alpha(t)$ in (3) and (4) can be extended to $\alpha(t, \cdot)$, wallowing for dependence on parameters beyond t.

Constant-Order Fractional Graph Neural Network: Recent dynamic process-inspired continuous GNNs (Kang et al. 2024a,b; Chamberlain et al. 2021b; Song et al. 2022; Zhao et al. 2023; Kang et al. 2023, 2025) are based on constant-order FDEs. We provide a brief introduction here and will apply our framework to initialize new continuous GNNs based on variable-order FDEs in Section 3.5.

An undirected graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathbf{W})$, where $\mathcal{V}$ is the set of $|\mathcal{V}|$ nodes and $\mathbf{Y} = \left(\left[\mathbf{y}^{(1)}\right]^{\intercal}, \ldots, \left[\mathbf{y}^{(|\mathcal{V}|)}\right]^{\intercal} \right)^{\intercal} \in \mathbb{R}^{|\mathcal{V}| \times d}$ consists of the node feature vectors $\mathbf{y}^{(i)} \in \mathbb{R}^{1 \times d}$. The adjacency matrix $\mathbf{W}$, an $|\mathcal{V}| \times |\mathcal{V}|$ matrix, has elements W_{ij} representing the edge weight between the i-th and j-th nodes with $W_{ij} = W_{ji}$. In the continuous GNNs, we define $\mathbf{Y}(t) = \left(\left[\mathbf{y}^{(1)}(t)\right]^{\intercal}, \ldots, \left[\mathbf{y}^{(|\mathcal{V}|)}(t)\right]^{\intercal} \right)^{\intercal} \in \mathbb{R}^{|\mathcal{V}| \times d}$ as the node features at time t, with $\mathbf{Y}(0) = \mathbf{Y}$ as the initial condition. Here, the time t serves as an analog to the layer index (Chen et al. 2018; Chamberlain et al. 2021a; Kang et al. 2024a). The dynamics of node features are typically described by the equation:

$$ {}_{0}D_t^{\alpha} \mathbf{Y}(t) = \mathcal{F}(\mathbf{W}, \mathbf{Y}(t)), \text{ with constant } \alpha, \quad (5) $$

where the function $\mathcal{F}$ is specifically designed for graph dynamics. For example, in fractional diffusion-inspired GNN models (Chamberlain et al. 2021a; Kang et al. 2024a), $\mathcal{F}$ is defined as $\mathcal{F}(\mathbf{W}, \mathbf{Y}(t)) = (\mathbf{A}(\mathbf{Y}(t)) - \mathbf{I})\mathbf{Y}(t)$, where $\mathbf{A}(\mathbf{Y}(t))$ is either a learnable attention matrix or a fixed normalized matrix, and $\mathbf{I}$ is an identity matrix. By setting an integration time T and integrating this equation, the updated feature matrix $\mathbf{Y}(t)$ can be obtained, which can subsequently be utilized for downstream tasks such as node classification.

3 Neural Variable-Order Fractional Differential Equation Networks

In this section, we introduce NvoFDE, which utilizes a neural network to parameterize the variable-order fractional derivative of the hidden state. This approach allows for the integration of a continuum of past states into the present state, enabling rich and flexible modeling of hidden features. We then outline a numerical scheme to solve this model. Furthermore, we present variable-order counterparts of several established constant-order continuous fractional GNNs. Additionally, we explore a detailed application of NvoFDE within the PINN methodology (Raissi, Perdikaris, and Karniadakis 2019), effectively solving variable-order FDEs.

3.1 Framework

We denote the hidden feature as $\mathbf{x}(t)$, where similar to (Chen et al. 2018; Kang et al. 2024a), the time t serves as an analog to the layer index. We propose the following continuous hidden feature updating scheme as a variable-order FDE:

$$ {}_{t_0}D_t^{\alpha(t,\mathbf{x}(t))} \mathbf{x}(t) = f_{\boldsymbol{\theta}}(t, \mathbf{x}(t)), \quad t_0 \leq t \leq t_1, \quad (6) $$

where $f_{\boldsymbol{\theta}}(t, \mathbf{x}(t))$ is a parameterized neural network that outputs the variable-order fractional derivative value at time t for hidden state $\mathbf{x}(t)$. In NvoFDE, the differential order $\alpha(t, \mathbf{x}(t))$ is determined by both t and $\mathbf{x}(t)$, and can be configured as a learnable neural network with scalar output. This design allows the order to be adaptive, evolving based on time and the dynamics of the hidden features. Without loss of generality, we restrict the range of the function $\alpha(t, \mathbf{x}(t))$ to the interval $(0, 1]$. This is justified by methodologies in (Diethelm 2010), which demonstrate that higher-order dynamics can be effectively converted to this range. Moreover, we define the initial state of our system with $\mathbf{x}(t_0) = \mathbf{x}_0$, considering $\mathbf{x}_0$ as the initial feature input.

By setting $\alpha(t, \mathbf{x}(t)) \equiv 1$, our model simplifies to the first-order neural ODE described in (Chen et al. 2018). Furthermore, setting it to a constant real value $\alpha(t, \mathbf{x}(t)) \equiv \alpha$ aligns it with the constant-order neural FDE discussed in (Kang et al. 2024a). Thus, NvoFDE effectively generalizes the class of fixed integer- or fractional-order neural differential equation models found in the literature.

Alternatively, motivated by equivalence between the differential form and its integral form (Diethelm 2010) for constant orders, we can formulate a continuous hidden feature updating scheme as a variable-order fractional integral equation (FIE) given by

$$\mathbf{x}(t) = \mathbf{x}(t_0) + J_{t_0}^{\alpha(t,\mathbf{x}(t))} f_{\boldsymbol{\theta}}(t, \mathbf{x}(t)). \tag{7}$$

It is clear that this framework integrates a continuum of past states into the current state and dynamically adjusts its sensitivity to previous hidden features through the variable order $\alpha(t, \mathbf{x}(t))$. We note that if $\alpha(t, \mathbf{x}(t))$ is constant, then (7) reduces to (6). To illustrate, if $\alpha(t, \mathbf{x}(t)) \equiv 1$, the integral equation becomes $\mathbf{x}(t) = \mathbf{x}(t_0) + \int_{t_0}^{t} f_{\boldsymbol{\theta}}(t, \mathbf{x}(t))$, which is equivalent to the differential equation $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = f_{\boldsymbol{\theta}}(t, \mathbf{x}(t))$. For a general $\alpha(t, \mathbf{x}(t))$, the scenarios differ because $\alpha(t, \mathbf{x}(t))$ is time-dependent, and transitioning between differential and integral operators is non-trivial.

3.2 Designing $\alpha(t, \mathbf{x}(t))$

In this section, we explore various designs for the function $\alpha(t, \mathbf{x}(t))$. One straightforward approach is to concatenate the time variable t with the state variable $\mathbf{x}(t)$ and process this combined vector through a Multilayer Perceptron (MLP) or convolutional layers to produce a scalar value representing the differential order. To ensure that $\alpha(t, \mathbf{x}(t))$ remains within the interval $(0, 1]$, we can apply a sigmoid activation function at the output. Alternatively, a more sophisticated method involves using the Transformer's sinusoidal position embedding (Vaswani et al. 2017) for the time variable t. This embedding can be adjusted to match the dimensionality of $\mathbf{x}(t)$, allowing for the direct addition of the time position embedding to $\mathbf{x}(t)$. The enhanced state representation is then processed through either an MLP or convolutional layer to yield a scalar. In simpler scenarios, $\alpha(t, \mathbf{x}(t))$ may depend solely on t, where $\alpha(t)$ could be a parameterized function of t or determined by setting learnable values on a uniform grid of t values over the interval $[t_0, t_1]$, with interpolation used to determine $\alpha(t)$.

3.3 Solving NvoFDE

Existing numerical solvers for integer-order neural ODEs and constant fractional-order neural networks have been extensively studied (Chen et al. 2018; Kang et al. 2024a; Diethelm, Ford, and Freed 2004). However, NvoFDE introduces a variable-order FDE/FIE and requires corresponding numerical solutions. In this paper, we examine the explicit L1 solver from (Sun et al. 2019) for (6) and two variants of the explicit Adams-Bashforth-Moulton (ABM) solver from (Moghaddam, Yaghoobi, and Tenreiro Machado 2016) for (7). Here, we discuss the L1 predictor as well as the ABM predictor variant. A comprehensive description of the ABM predictor-corrector variant is provided in the Appendix.

In the following discussion, we consider a uniform grid spanning $[t_0, t_1]$ defined by $\{t_n = t_0 + nh \,:\, n = 0, \ldots, N\}$, where $h = \frac{t_1 - t_0}{N}$. Furthermore, let $\mathbf{x}_n$ be an approximation of $\mathbf{x}(t_n)$.

L1 Predictor: The solution to (6) based on the L1 approximation of the variable-order Caputo fractional derivative (Sun et al. 2019) is formulated as:

$$\mathbf{x}_{n+1} = \sum_{j=0}^{n} a_{j,n+1}\mathbf{x}_j + c_{n+1} f_\theta(t_n, \mathbf{x}_n), \tag{8}$$

where $c_{n+1} = \Gamma(2 - \alpha(t_n, \mathbf{x}_n))h^{\alpha(t_n,\mathbf{x}_n)}$, $a_{0,n+1} = (n+1)^{1-\alpha(t_n,\mathbf{x}_n)} - n^{1-\alpha(t_n,\mathbf{x}_n)}$, and $a_{j,n+1} = 2(n+1-j)^{1-\alpha(t_n,\mathbf{x}_n)} - (n-j)^{1-\alpha(t_n,\mathbf{x}_n)} - (n+2-j)^{1-\alpha(t_n,\mathbf{x}_n)}$.

ABM Predictor: The preliminary approximation solution to the variable-order FIE (7) in NvoFDE is formulated as (Diethelm, Ford, and Freed 2004; Moghaddam, Yaghoobi, and Tenreiro Machado 2016):

$$\mathbf{x}_{n+1} = \mathbf{x}_0 + \frac{h^{\alpha(t_n,\mathbf{x}_n)}}{\Gamma(1 + \alpha(t_n, \mathbf{x}_n))} \sum_{j=0}^{n} b_{j,n+1} f_{\boldsymbol{\theta}}(t_j, \mathbf{x}_j), \tag{9}$$

where $b_{j,n+1} = (n+1-j)^{\alpha(t_n,\mathbf{x}_n)} - (n-j)^{\alpha(t_n,\mathbf{x}_n)}, 1 \leq j \leq n$. Note that (9) can be regarded as the fractional Euler's method.

3.4 NvoFDE for Solving Variable-Order FDEs

Since finding the analytical solutions to general FDEs is difficult or impossible, recently, neural networks have been taken as a universal function to approximate the solutions. Variable-order FDEs describe the variable memory of dynamical systems and do well in representing memory characteristics that change with time or space (Lorenzo and Hartley 2002; Sun et al. 2011). In this section, we consider minimization problems with variable-order FDEs as constraints. It is vital to solve variable-order FDEs by a general approach. To this end, we aim to solve variable-order FDEs using the NvoFDE framework. Existing numerical methods that leverage neural networks to solve variable-order FDEs typically rely on specific mathematical formulations of the order $\alpha(t, \mathbf{x}(t))$, often assuming $\alpha(t, \mathbf{x}(t))$ to be a linear or periodic function over time (Moghaddam, Yaghoobi, and Tenreiro Machado 2016; Sun et al. 2011). In contrast, our approach departs from these restrictive assumptions on $\alpha(t, \mathbf{x}(t))$, offering a more flexible and generalized treatment.

We first consider the following variable-order FDE:

$$\sum_{k=0}^{m} Q_k(t)\, {}_{t_0}D_t^{\alpha_k(t)}[u(t)] = h(t, u(t)), \quad t \in [t_0, t_1], \tag{10}$$

with the initial value $u(t_0)$ and m being a constant. Here, the undetermined $\alpha_k(t)$ is the fractional order varying with t, while $Q_k(t)$ and $h(t, u)$ are given real-valued analytical functions. Since finding analytical solutions to FDEs is challenging, we focus on using power series expansions to approximate the solution. Thus, we express $u(t)$ as $u(t) = u(0) + \sum_{i=0}^{r} a_i t^i$, where r is a positive constant that tends to infinity, and a_i are the unknown coefficients to be determined. In practice, we consider the truncated power series expansion of $u(t)$, specifically setting r to 5 in our experiment. When neural networks are used to solve the numerical solution to (10), the loss function is defined as

$$\mathcal{J}[u] = \left\| \sum_{k=0}^{m} Q_k(t)\, {}_{t_0}D_t^{\alpha_k(t)}[u(0) + \sum_{i=0}^{r} a_i t^i] - h(t, u(t)) \right\|,$$

aiming to minimize a functional of u, i.e., $\mathcal{J}[u]$, under the constraint (10). A common algorithm is to solve (10) numerically (e.g., using the L1 or ABM Predictor), and then optimize $\mathcal{J}[u]$ with respect to $\alpha_k(t)$ and other possible parameters. The above functional $\mathcal{J}[u]$ is equivalent to

$$\mathcal{J}[u] = \left\| \sum_{k=0}^{m} Q_k(t)\zeta_k(t) - h(t,u) \right\|^2 \tag{11}$$

with $\zeta_k(t) = \sum_{i=0}^{r} \frac{\Gamma(i+1)}{\Gamma(i+1-\alpha_k(t))} a_i t^{i-\alpha_k(t)}$, where a_i are unknown coefficients that will be optimized using the neural networks introduced later. For (11), we utilize the variable-order fractional derivative expression of each power term t^n for $0 < \alpha(t) \leq 1$ (Akgül, Baleanu et al. 2017):

$${}_0D_t^{\alpha(t)} t^n = \begin{cases} 0, & \text{if } n = 0, \\ \frac{\Gamma(n+1)}{\Gamma(n+1-\alpha(t))} t^{n-\alpha(t)}, & \text{if } n \in \mathbb{N}, \end{cases}$$

where $\mathbb{N}$ denotes the set of non-negative integers. Therefore, we approximate $\mathcal{J}[u]$ as follows:

$$L_{\text{eqn}} = \sum_{j=1}^{N} \left(\sum_{k=0}^{m} Q_k(t_j)\zeta_k(t_j) - h(t_j, u(t_j)) \right)^2, \tag{12}$$

which is referred to as the equation residual for (10).

As an example, we consider the following variable-order FDE, which is the well-known Verhulst-Pearl equation (Moghaddam, Yaghoobi, and Tenreiro Machado 2016):

$$\begin{cases} {}_0D_t^{\alpha(t)} u(t) = 0.3u(t) - 0.3u(t)u(t-\delta), & 0 < t \leq T, \\ u(t) = 0.1, & -\delta \leq t \leq 0, \end{cases} \tag{13}$$

where δ is a constant delay. The equation (13) models the population growth under environmental constraints. In this paper, we study the case where $\delta = 0$ and solve the equation by NvoFDE. Our framework is presented in Fig. 1.

For (13), since the solution $u(t)$ is a function that only involves time t, we treat t as the input for the entire neural network. Specifically, we first uniformly select t_j points on the time interval $[0, T]$ as the input with j = 10, 20, 30, 40, etc. For simplicity, we set $T = 1$ in the experiment. The network then gives the output $\hat{u}(t)$. Considering the initial condition $u(0)$ = 0.1 in (13), the total loss function of (13) consists of two components. The first component is the equation residual L_{eqn} defined in (12). Based on (13), it can be seen that $h(t,u) = 0.3u - 0.3u^2$, $Q_k(t) = 1$ and $m = 1$ in (12), where $u(t)$ is provided by the ABM predictor solver discussed in Section 3.3. The second component is the initial condition residual L_{ini}, calculated as $L_{\text{ini}} = \sum_{i=1}^{N} \left\| \hat{u}_i - u(0) \right\|^2$. Thus, we have the total loss function $L_{\text{total}} = \lambda_1 L_{\text{eqn}} + \lambda_2 L_{\text{ini}}$, where λ_1 and λ_2 are predefined positive weights.

3.5 NvoFDE for GNNs

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathbf{W})$ as defined in Section 2. In contrast to the constant-order FROND model in (Kang et al. 2024a), we introduce a learnable fractional-order function $\alpha(t, \mathbf{Y}(t))$ to dynamically capture the memory mechanism

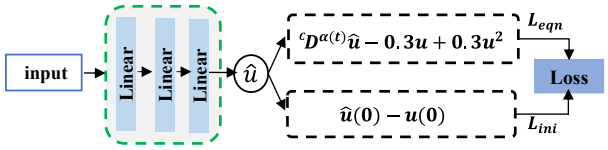

Figure 1: NvoFDE for the Verhulst-Pearl equation. Taking time t as the input to the neural network, $\hat{u}$ is obtained as the output. On the one hand, $\hat{u}$ is involved in (12) and (13) to compute the equation residual L_{eqn} by virtue of the ABM predictor; on the other hand, $\hat{u}$ is used to calculate the initial condition residual L_{ini} based on the initial value of (13).

of feature updating. This new GNN framework is called variable-order FROND (V-FROND). The dynamics of information propagation and feature updating in V-FROND are governed by the following NvoFDE:

$${}_{t_0}D_t^{\alpha(t,\mathbf{Y}(t))} \mathbf{Y}(t) = \mathcal{F}(\mathbf{W}, \mathbf{Y}(t)), \tag{14}$$

where $0 < \alpha(t, \mathbf{Y}(t)) \leq 1$, and the initial state $\mathbf{Y}(0) = \mathbf{Y}$ consists of the input initial node features. Here, $\mathcal{F}$ is the dynamic operator on the graph, as described in Section 2. By setting an integration time T and integrating this equation, the updated feature matrix $\mathbf{Y}(t)$ can be obtained using (9), which can subsequently be utilized for downstream tasks such as node classification.

Inspired by the models in (Kang et al. 2024a), we develop V-FROND variants, including Nvo-GRAND and Nvo-CDE. Similar to (Chamberlain et al. 2021b), Nvo-GRAND includes two versions. One is Nvo-GRAND-nl:

$${}_{t_0}D_t^{\alpha(t,\mathbf{Y}(t))} \mathbf{Y}(t) = (\mathbf{A}(\mathbf{Y}(t)) - \mathbf{I})\mathbf{Y}(t), \tag{15}$$

where $\mathbf{A}(\mathbf{Y}(t)) = (a_{i,j}(t))$ is given by a nonlinear attention mechanism as detailed in the Appendix. The other version is Nvo-GRAND-l, defined as:

$${}_{t_0}D_t^{\alpha(t,\mathbf{Y}(t))} \mathbf{Y}(t) = -\mathbf{L}\mathbf{Y}(t), \quad 0 < \alpha(t, \mathbf{Y}(t)) \leq 1. \tag{16}$$

where $\mathbf{L}$ is a time-invariant matrix, as detailed in the Appendix. This is a linear FDE.

Furthermore, building upon the CDE model (Zhao et al. 2023), the Nvo-CDE model is defined as follows:

$$\begin{aligned} {}_{t_0}D_t^{\alpha(t,\mathbf{Y}(t))} \mathbf{Y}(t) = &\left(\mathbf{A}(\mathbf{Y}(t)) - \mathbf{I}\right) \mathbf{Y}(t) \\ &+ \operatorname{div}\left(\mathbf{V}(t) \circ \mathbf{Y}(t)\right), \end{aligned} \tag{17}$$

where $\operatorname{div}(\cdot)$ is the divergence operator provided by (Song et al. 2022), and $\circ$ represents the element-wise (Hadamard) product. This model is crafted to handle heterophilic graphs, where connected nodes belong to different classes or have distinct features. In Section 4, we demonstrate the effectiveness of the proposed variable-order FROND framework across multiple graph datasets, showcasing its superior adaptability and performance.

4 Experiments

In this section, we conduct a series of experiments to demonstrate the effectiveness of our proposed methods. All experiments are implemented using the PyTorch framework (Paszke et al. 2019) on a single NVIDIA RTX A4000 16GB GPU. For further details, please refer to the Appendix.

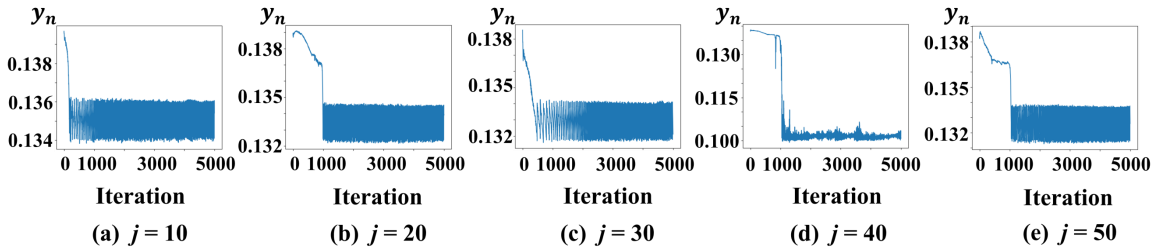

Figure 2: Numerical solutions of the Verhulst-Pearl equation over iterations on the evolution time [0, 1]

4.1 Experiments on the Verhulst-Pearl equation

Datasets and Training details. For the training set, we discretize the time interval $[0, 1]$ uniformly for simplicity, obtaining points t_j where j is a positive integer with j = 10, 20, 30, 40, etc. We use the Adam algorithm (Kingma and Ba 2014) with iterations of 200, 500, 1000, 1500 and 2000. The learning rate is set to 0.01. The model employs a three-layer neural network, as shown in Fig.1, with the hidden layer consisting of 30 neurons. The test set is formed by randomly selecting points t_j from the interval $[0, 1]$, where j takes values like 10, 20, 30, 40, and so on.

Performance and Analysis. As seen in Table 1, the loss value decreases significantly with the number of iterations. When the step size decreases, which corresponds to an increase of discrete points j, it can be observed that better results are obtained for $j = 20$ and $j = 40$. Additionally, during the experiments, it was found that for Iter $= 2000$, the numerical result reached an accuracy of 10^{-6} in 28 instances for $j = 20$ (with results printed every 30 iterations), and in 40 instances for $j = 40$. This indicates that the model is more stable while maintaining high accuracy for $j = 40$.

In Fig 2, we present the numerical solution of the Verhulst-Pearl equation over multiple iterations. By varying the step sizes ($j = 10, 20, 30, 40, 50$), we observe that after approximately 1000 iterations, all solutions reach a relatively stable state, exhibiting periodic variations. When $j = 40$, corresponding to a step size of $1/40$, the amplitude of the solution significantly diminishes after 1000 iterations compared to the solutions with other step sizes. This suggests the numerical solution tends towards a more stable state when $j = 40$, which is consistent with the loss analysis discussed above.

Iter	$j = 10$	$j = 20$	$j = 30$	$j = 40$	$j = 50$
200	8.39E-4	5.38E-4	4.66E-4	5.00E-4	4.01E-4
500	2.38E-4	3.79E-5	1.93E-4	7.38E-5	1.99E-4
1000	2.43E-5	4.62E-5	8.21E-5	9.60E-5	3.14E-5
1500	1.67E-5	7.46E-6	8.15E-6	8.60E-6	3.71E-5
2000	3.69E-6	4.50E-6	3.33E-6	4.58E-6	8.34E-6

Table 1: Test loss of the Verhulst-Pearl equation over iterations on the evolution time $[0, 1]$

4.2 Node Classification on Homophilic Graph

Datasets. Our study encompasses diverse datasets with various topologies. For the Disease and Airport datasets, we employ the same data splitting and pre-processing methods as detailed in (Chami et al. 2019). For the remaining datasets, we follow the experimental settings used in GRAND (Chamberlain et al. 2021a) and F-GRAND, applying random splits to the largest connected component of each dataset. For more details regarding the dataset, please refer to the Appendix.

Methods. For our comparative analysis, we select several prominent GNN models, including GCN (Kipf and Welling 2017), GAT (Veličković et al. 2018), HGCN (Chami et al. 2019), and GIL (Zhu et al. 2020). GRAND (Chamberlain et al. 2021a) serves as a specific instance of Nvo-GRAND, characterized by the order $\alpha(t, \mathbf{x}(t)) = 1$. We examine two variants of Nvo-GRAND: Nvo-GRAND-nl (15) and Nvo-GRAND-l (16). For consistency in comparison, we report baseline results from the FROND paper (Kang et al. 2024a).

Performance and Analysis. Table 2 presents the experimental results. As expected, even though F-GRAND already achieves excellent performance, Nvo-GRAND still surpasses F-GRAND on almost all datasets. The main reason lies in Nvo-GRAND's learnable ability to flexibly adjust its memory mechanism, allowing the model to find local optimal solutions and optimize its memory trajectory. Additionally, we illustrate the evolutionary trend of the order $\alpha(t, \mathbf{x}(t))$ in Figure 3. We select three different types of datasets, namely Computers, Pubmed and Disease. The initial order is set to $\alpha(t) \equiv 0.8$ at each time point. After training, we observe that $\alpha(t, \mathbf{x}(t))$ has dramatically altered its evolutionary path over time. Our method allows $\alpha(t, \mathbf{x}(t))$ to adapt actively, rather than relying on conventional mathematical designs such as linear functions or periodic functions with respect to t as used in prior work (Moghaddam, Yaghoobi, and Tenreiro Machado 2016; Sun et al. 2011).

4.3 Node Classification on Heterophilic Graph

Datasets. The paper (Platonov et al. 2023) demonstrated significant flaws in the standardized datasets used for evaluating models on heterophilic graphs and introduced six heterophilic graph datasets, namely Roman-empire, Wiki-cooc, Minesweeper, Questions, Workers, and Amazon-ratings. These datasets come from various domains, have low homophily scores, and exhibit diverse structural properties. The dataset splits used in this section follow the same approach as in (Platonov et al. 2023). Consistent with (Zhao et al. 2023), the ROC-AUC score is utilized as the evaluation metric for the Minesweeper, Workers, and Questions datasets, given that they entail binary classification.

Methods. To ensure a fair comparison, we adhere to the ex-

Method	Cora	Citeseer	Pubmed	CoauthorCS	Computer	Photo	CoauthorPhy	ogbn-arxiv	Airport	Disease
GCN	81.5±1.3	71.9±1.9	77.8±2.9	91.1±0.5	82.6±2.4	91.2±1.2	92.8±1.0	72.2±0.3	81.6±0.6	69.8±0.5
GAT	81.8±1.3	71.4±1.9	78.7±2.3	90.5±0.6	78.0±19.0	85.7±20.3	92.5±0.9	**73.7±0.1**	81.6±0.4	70.4±0.5
HGCN	78.7±1.0	65.8±2.0	76.4±0.8	90.6±0.3	80.6±1.8	88.2±1.4	90.8±1.5	59.6±0.4	85.4±0.7	89.9±1.1
GIL	82.1±1.1	71.1±1.2	77.8±0.6	89.4±1.5	–	89.6±1.3	–	–	91.5±1.7	90.8±0.5
GRAND-l	83.6±1.0	73.4±0.5	78.8±1.7	92.9±0.4	83.7±1.2	92.3±0.9	93.5±0.9	71.9±0.2	80.5±9.6	74.5±3.4
F-GRAND-l	84.8±1.1	74.0±1.5	79.4±1.5	93.0±0.3	84.4±1.5	92.8±0.6	94.5±0.4	72.6±0.1	98.1±0.2	92.4±3.9
Nvo-GRAND-l (ours)	**86.0±0.5**	75.6±0.8	**80.8±1.2**	**93.4±0.2**	**87.9±0.8**	**94.1±0.2**	**94.7±0.2**	71.8±0.1	**98.7±0.2**	**97.4±0.7**
GRAND-nl	82.3±1.6	70.9±1.0	77.5±1.8	92.4±0.3	82.4±2.1	92.4±0.8	91.4±1.3	71.2±0.2	90.9±1.6	81.0±6.7
F-GRAND-nl	83.2±1.1	74.7±1.9	79.2±0.7	92.9±0.4	84.1±0.9	93.1±0.9	93.9±0.5	71.4±0.3	96.1±0.7	85.5±2.5
Nvo-GRAND-nl (ours)	85.4±1.0	**75.9±0.6**	80.6±0.7	**93.4±0.2**	87.2±1.4	94.0±0.3	94.6±0.2	72.0±0.2	98.4±0.2	89.8±3.4

Table 2: Node classification results (%) for random train-val-test splits. The best and the second-best results for each criterion are highlighted in bold and underlined, respectively.

Model	Roman-empire	Wiki-cooc	Minesweeper	Questions	Workers	Amazon-ratings
GRAND-l	69.24±0.53	91.58±0.37	73.25±0.99	68.54±1.07	75.59±0.86	48.99±0.35
GRAND-nl	71.60±0.58	92.03±0.46	76.67±0.98	70.67±1.28	75.33±0.84	45.05±0.65
GraphBel	69.47±0.37	90.30±0.50	76.51±1.03	70.79±0.99	73.02±0.92	43.63±0.42
NSD	77.50±0.67	92.06±0.40	89.59±0.61	69.25±1.15	79.81±0.99	37.96±0.20
CDE	91.64±0.28	97.99±0.38	95.50±5.23	**75.17±0.99**	80.70±1.04	47.63±0.43
F-CDE	93.06±0.55	98.73±0.68	96.04±0.25	**75.17±0.99**	82.68±0.86	49.01±0.56
Nvo-CDE (ours)	**93.42±0.22**	**99.32±0.28**	**98.53±0.27**	74.87±0.23	**83.33±0.65**	**50.09±0.40**

Table 3: Node classification results (%) on large heterophilic datasets.

perimental settings outlined in the CDE paper (Zhao et al. 2023). We evaluate the performance of the Nvo-CDE model by comparing it against other graph neural ODE models, including GraphBel (Song et al. 2022), NSD (Bodnar et al. 2022), CDE (Zhao et al. 2023) and F-CDE (Kang et al. 2024a), as detailed in Table 3.

Performance and Analysis. From Table 3, it is evident that our Nvo-CDE model outperforms competitors on five out of six datasets, demonstrating the efficacy of the Nvo-CDE model. On the Minesweeper dataset, the use of variable-order derivatives improves performance by 2.49%. This advantage stems from the framework's flexibility in modeling adaptive derivative orders based on hidden features, capturing more complex feature-updating dynamics.

4.4 Image Classification of NvoFDE

In this part, we consider applying the NvoFDE framework to image classification tasks. We refer to the adapted model as the NvoFDE-based Image Classification, abbreviated as Nvo-IC. The architecture of Nvo-IC mirrors that of Neural ODEs (Chen et al. 2018), with a significant modification: the core component is a variable-order FDE block instead of the traditional integer-order ODE block. We evaluate the Nvo-IC model by conducting experiments on the Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017) and CIFAR (Krizhevsky and Hinton 2009) datasets, assessing classification accuracy on both clean and stochastically noisy images. The experimental outcomes indicate that Nvo-IC marginally outperforms traditional models in these tasks, underscoring the efficacy and potential of the NvoFDE framework for complex image classification scenarios. For detailed experimental results and further analysis, please refer to the Appendix.

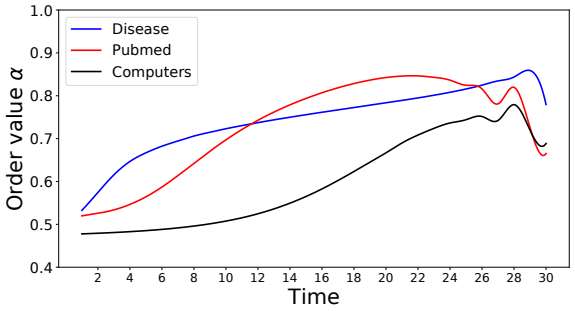

Figure 3: The order value evolution of $\alpha(t, \mathbf{x}(t))$.

5 Conclusion

We present the Neural Variable-Order Fractional Differential Equation network (NvoFDE), a novel framework that integrates variable-order fractional derivatives. Our approach enables the modeling of adaptive derivative orders based on hidden features, capturing complex feature-updating dynamics with greater flexibility. We apply NvoFDE to graph neural networks, enhancing the capabilities of existing constant-order fractional GNNs. Through extensive experiments on multiple graph datasets, we demonstrate the superior adaptability and performance of NvoFDE. Additionally, NvoFDE introduces a dynamic capability for solving variable-order FDEs, where the system's order is learnable from empirical observations. Our framework holds significant potential for various applications, such as learnable fractional viscoelastic modeling and adaptive control with learned orders.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant Nos. U2268203, 12301491, 12225107 and 12071195, the Major Science and Technology Projects in Gansu Province-Leading Talents in Science and Technology under Grant No. 23ZDKA0005. This research is also supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research and Development Programme.To improve the readability, parts of this paper have been grammatically revised using ChatGPT (OpenAI 2022).

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 265–283.

Akgül, A.; Baleanu, D.; et al. 2017. On solutions of variable-order fractional differential equations. *An International Journal of Optimization and Control: Theories & Applications (IJOCTA)*, 7(1): 112–116.

Anastasio, T. J. 1994. The fractional-order dynamics of brainstem vestibulo-oculomotor neurons. *Biological cybernetics*, 72(1): 69–79.

Antil, H.; Khatri, R.; Löhner, R.; and Verma, D. 2020. Fractional deep neural network via constrained optimization. *Mach. Learn.: Sci. Technol.*, 2(1): 015003.

Bodnar, C.; Giovanni, F. D.; Chamberlain, B. P.; Liò, P.; and Bronstein, M. M. 2022. Neural Sheaf Diffusion: A Topological Perspective on Heterophily and Oversmoothing in GNNs. In *Advances Neural Inf. Process. Syst.*

Chamberlain, B.; Rowbottom, J.; Gorinova, M. I.; Bronstein, M.; Webb, S.; and Rossi, E. 2021a. Grand: Graph neural diffusion. In *Proc. Int. Conf. Mach. Learn.*, 1407–1418.

Chamberlain, B. P.; Rowbottom, J.; Goronova, M.; Webb, S.; Rossi, E.; and Bronstein, M. M. 2021b. GRAND: Graph Neural Diffusion. In *Proc. Int. Conf. Mach. Learn.*

Chami, I.; Ying, Z.; Ré, C.; and Leskovec, J. 2019. Hyperbolic graph convolutional neural networks. In *Advances Neural Inf. Process. Syst.*

Chen, R. T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. 2018. Neural ordinary differential equations. In *Advances Neural Inf. Process. Syst.*

Coimbra, C. F. 2003. Mechanics with variable-order differential operators. *Annalen der Physik*, 515(11-12): 692–703.

Coleman, B. D.; and Noll, W. 1961. Foundations of linear viscoelasticity. *Rev. Modern Phys.*, 33(2): 239.

Cui, W.; Zhang, H.; Chu, H.; Hu, P.; and Li, Y. 2023. On robustness of neural ODEs image classifiers. *Information Sciences*, 632: 576–593.

Dai, S.; Qu, C.; Chen, S.; Zhang, X.; and Xu, J. 2024. Recode: Modeling repeat consumption with neural ode. In *Proc. Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, 2599–2603.

Diethelm, K. 2010. *The analysis of fractional differential equations: an application-oriented exposition using differential operators of Caputo type*, volume 2004. Springer.

Diethelm, K.; Ford, N. J.; and Freed, A. D. 2004. Detailed error analysis for a fractional Adams method. *Numer. Algorithms*, 36: 31–52.

Dupont, E.; Doucet, A.; and Teh, Y. W. 2019. Augmented neural odes. In *Advances Neural Inf. Process. Syst.*, 1–11.

Glöckle, W. G.; and Nonnenmacher, T. F. 1995. A fractional calculus approach to self-similar protein dynamics. *Biophysical Journal*, 68(1): 46–53.

Gravina, A.; Bacciu, D.; and Gallicchio, C. 2022. Anti-symmetric dgn: A stable architecture for deep graph networks. In *Proc. Int. Conf. Learn. Representations*.

Haber, E.; and Ruthotto, L. 2017. Stable architectures for deep neural networks. *Inverse Problems*, 34(1): 1–23.

Kang, Q.; Li, X.; Zhao, K.; Cui, W.; Zhao, Y.; Deng, W.; and Tay, W. P. 2025. Efficient Training of Neural Fractional-Order Differential Equation via Adjoint Backpropagation. In *Proc. AAAI Conference on Artificial Intelligence*. USA.

Kang, Q.; Song, Y.; Ding, Q.; and Tay, W. P. 2021. Stable neural ODE with Lyapunov-stable equilibrium points for defending against adversarial attacks. In *Advances Neural Inf. Process. Syst.*

Kang, Q.; Zhao, K.; Ding, Q.; Ji, F.; Li, X.; Liang, W.; Song, Y.; and Tay, W. P. 2024a. Unleashing the Potential of Fractional Calculus in Graph Neural Networks with FROND. In *Proc. International Conference on Learning Representations*.

Kang, Q.; Zhao, K.; Song, Y.; Wang, S.; and Tay, W. P. 2023. Node Embedding from Neural Hamiltonian Orbits in Graph Neural Networks. In *Proc. International Conference on Machine Learning*, 15786–15808.

Kang, Q.; Zhao, K.; Song, Y.; Xie, Y.; Zhao, Y.; Wang, S.; She, R.; and Tay, W. P. 2024b. Coupling Graph Neural Networks with Fractional Order Continuous Dynamics: A Robustness Study. In *Proc. AAAI Conference on Artificial Intelligence*. Vancouver, Canada.

Kaslik, E.; and Sivasundaram, S. 2012. Nonlinear dynamics and chaos in fractional-order neural networks. *Neural networks*, 32: 245–256.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. Int. Conf. Learn. Representations*.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*.

Lasaki, F. G.; Ebrahimi, H.; and Ilie, M. 2023. A novel lagrange functional link neural network for solving variable-order fractional time-varying delay differential equations: a comparison with multilayer perceptron neural networks. *Soft Computing*, 27(17): 12595–12608.

Liu, Z.; Wang, Y.; Luo, Y.; and Luo, C. 2022. A Regularized Graph Neural Network Based on Approximate Fractional Order Gradients. *Mathematics*, 10(8): 1320.

Lorenzo, C. F.; and Hartley, T. T. 2002. Variable order and distributed order fractional operators. *Nonlinear dynamics*, 29: 57–98.

Machado, J. T.; Kiryakova, V.; and Mainardi, F. 2011. Recent history of fractional calculus. *Communications in nonlinear science and numerical simulation*, 16(3): 1140–1153.

Meng, R.; Yin, D.; Zhou, C.; and Wu, H. 2016. Fractional description of time-dependent mechanical property evolution in materials with strain softening behavior. *Applied Mathematical Modelling*, 40(1): 398–406.

Moghaddam, B. P.; Yaghoobi, S.; and Tenreiro Machado, J. 2016. An extended predictor–corrector algorithm for variable-order fractional delay differential equations. *Journal of Computational and Nonlinear Dynamics*, 11(6).

Nigmatullin, R. 1986. The realization of the generalized transfer equation in a medium with fractal geometry. *Physica status solidi (b)*, 133(1): 425–430.

Nobis, G.; Aversa, M.; Springenberg, M.; Detzel, M.; Ermon, S.; Nakajima, S.; Murray-Smith, R.; Lapuschkin, S.; Knochenhauer, C.; Oala, L.; et al. 2023. Generative Fractional Diffusion Models. *arXiv preprint arXiv:2310.17638*.

Obembe, A. D.; Hossain, M. E.; and Abu-Khamsin, S. A. 2017. Variable-order derivative time fractional diffusion model for heterogeneous porous media. *Journal of Petroleum Science and Engineering*, 152: 391–405.

OpenAI. 2022. ChatGPT-4. Available at: https://www.openai.com (Accessed: 10 April 2024).

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances Neural Inf. Process. Syst.*

Pfau, D.; Spencer, J. S.; Matthews, A. G.; and Foulkes, W. M. C. 2020. Ab initio solution of the many-electron Schrödinger equation with deep neural networks. *Physical review research*, 2(3): 033429.

Platonov, O.; Kuznedelev, D.; Diskin, M.; Babenko, A.; and Prokhorenkova, L. 2023. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? *arXiv preprint arXiv:2302.11640*.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378: 686–707.

Samko, S. G.; and Ross, B. 1993. Integration and differentiation to a variable fractional order. *Integral transforms and special functions*, 1(4): 277–300.

Scalas, E.; Gorenflo, R.; and Mainardi, F. 2000. Fractional calculus and continuous-time finance. *Physica A: Statistical Mechanics and its Applications*, 284(1-4): 376–384.

Shukla, K.; Di Leoni, P. C.; Blackshire, J.; Sparkman, D.; and Karniadakis, G. E. 2020. Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks. *Journal of Nondestructive Evaluation*, 39: 1–20.

Song, Y.; Kang, Q.; Wang, S.; Zhao, K.; and Tay, W. P. 2022. On the Robustness of Graph Neural Diffusion to Topology Perturbations. In *Advances Neural Inf. Process. Syst.*

Song, Y.; Sohl-Dickstein, J.; Kingma, D. P.; Kumar, A.; Ermon, S.; and Poole, B. 2021. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*.

Sun, H.; Chang, A.; Zhang, Y.; and Chen, W. 2019. A review on variable-order fractional differential equations: mathematical foundations, physical models, numerical methods and applications. *Fract. Calc. Appl. Anal.*, 22(1): 27–59.

Sun, H.; Chen, W.; Wei, H.; and Chen, Y. 2011. A comparative study of constant-order and variable-order fractional models in characterizing memory property of systems. *The european physical journal special topics*, 193(1): 185–192.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances Neural Inf. Process. Syst.*

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *Proc. Int. Conf. Learn. Representations*, 1–12.

Weinan, E. 2017. A proposal on machine learning via dynamical systems. *Commun. Math. Statist.*, 1(5): 1–11.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xu, C.; Liao, M.; Li, P.; Guo, Y.; Xiao, Q.; and Yuan, S. 2019. Influence of multiple time delays on bifurcation of fractional-order neural networks. *Applied Mathematics and Computation*, 361: 565–582.

Xu, W.; Chen, R. T.; Li, X.; and Duvenaud, D. 2022. Infinitely deep bayesian neural networks with stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, 721–738. PMLR.

Yan, H.; Du, J.; Tan, V. Y.; and Feng, J. 2018. On robustness of neural ordinary differential equations. In *Advances Neural Inf. Process. Syst.*, 1–13.

Yang, Y.; He, Y.; Wang, Y.; and Wu, M. 2018. Stability analysis of fractional-order neural networks: an LMI approach. *Neurocomputing*, 285: 82–93.

Zhang, L.; Han, J.; Wang, H.; Car, R.; and E, W. 2018. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Physical review letters*, 120(14): 143001.

Zhao, K.; Kang, Q.; Song, Y.; She, R.; Wang, S.; and Tay, W. P. 2023. Graph Neural Convection-Diffusion with Heterophily. In *Proc. Inter. Joint Conf. Artificial Intell.* China.

Zhu, S.; Pan, S.; Zhou, C.; Wu, J.; Cao, Y.; and Wang, B. 2020. Graph Geometry Interaction Learning. In *Advances Neural Inf. Process. Syst.*

Zhu, W.; Xu, K.; Darve, E.; and Beroza, G. C. 2021. A general approach to seismic inversion with automatic differentiation. *Computers & Geosciences*, 151: 104751.

Zhu, Y.; Zabaras, N.; Koutsourelakis, P.-S.; and Perdikaris, P. 2019. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *J. Comput. Phys.*, 394: 56–81.