Neural Fractional Differential Equations: Optimising the Order of the Fractional Derivative

Cecília Coelho *, M. Fernanda P. Costa, Luís L. Ferrás

Posted Date: 31 July 2024

doi: 10.20944/preprints202407.2572.v1

Keywords: Fractional Differential Equations; Neural Networks; Optimisation

 doi:10.20944/preprints202407.2572.v1

Article

Neural Fractional Differential Equations: Optimising the Order of the Fractional Derivative

Cecília Coelho [1,*], M. Fernanda P. Costa [1] and Luís L. Ferrás [1,2]

[1] Centre of Mathematics (CMAT), University of Minho, Portugal; cmartins@cmat.uminho.pt; mfc@math.uminho.pt
[2] Department of Mechanical Engineering (Section of Mathematics) and CEFT - Centro de Estudos de Fenómenos de Transporte, FEUP, University of Porto, Portugal; lferras@fe.up.pt
* Correspondence: cmartins@cmat.uminho.pt

Abstract: Neural Fractional Differential Equations (Neural FDE) represent a neural network architecture specifically designed to fit the solution of a fractional differential equation to given data. This architecture combines an analytical component, represented by a fractional derivative, with a neural network component, forming an initial value problem. During the learning process, both the order of the derivative and the parameters of the neural network must be optimised. In this work, we investigate the non-uniqueness of the optimal order of the derivative and its interaction with the neural network component. Based on our findings, we perform a numerical analysis to examine how different initialisations and values of the order of the derivative (in the optimisation process) impact its final optimal value. Results show that the neural network on the right-hand side of the Neural FDE struggles to adjust its parameters to fit the FDE to the data dynamics for any given order of the fractional derivative. Consequently, Neural FDEs do not require a unique α value, instead, they can use a wide range of α values to fit data. This flexibility is beneficial when fitting to given data is required, and the underlying physics is not known.

Keywords: fractional differential equations; neural networks; optimisation

1. Introduction

Real-world systems are often modelled using integral/differential equations, which are then numerically solved to predict the system behaviour and evolution. This process can be time-consuming, as numerical simulations sometimes take months, and, finding the *correct* model parameters is often challenging. However, with significant advancements in Neural Networks (NNs) that can learn patterns, real-world systems are increasingly being modelled using a combination of integral/differential models and NNs, or even NNs alone [1–4].

Neural Ordinary Differential Equations (Neural ODEs) were introduced in 2018 [5] (see also [6,7]) as a continuous version of the discrete Residual Neural Networks, and claimed to offer a continuous modelling solution for real-world systems that incorporate time-dependence, mimicking the dynamics of that system using only discrete data. Once trained, the Neural ODEs result in a *hybrid* ODE (part analytical, part NN-based) that can be used for making predictions, by numerically solving the resulting ODEs. The numerical solution of these *hybrid* ODEs is significantly simpler and less time-consuming compared to the numerical solution of complex governing equations, making Neural ODEs an excellent choice for modelling time-dependent real-world systems. However, the simplicity of ODEs sometimes limits their effectiveness in capturing complex behaviours characterised by intricate dynamics, non-linear interactions, and memory. To address this, Neural Fractional Differential Equations (Neural FDEs) were recently proposed [8,9].

Neural FDEs, as described by Equation (1), are a NN architecture designed to fit the solution $\boldsymbol{h}(t)$ to given data $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ (for example, experimental data), over a specified time range $[t_0, T]$. The Neural FDE combines an analytical part, ${}_0^C D_t^\alpha \boldsymbol{h}(t)$, with a NN-based part, $\boldsymbol{f}_{\boldsymbol{\theta}}(t, \boldsymbol{h}(t))$, leading to the initial value problem,

 doi:10.20944/preprints202407.2572.v1

$$\begin{cases} {}_0^C D_t^\alpha \boldsymbol{h}(t) = \boldsymbol{f}_{\boldsymbol{\theta}}(t, \boldsymbol{h}(t)), \\ \boldsymbol{h}(t_0) = \boldsymbol{x}_0, \quad t \in [t_0, T]. \end{cases} \tag{1}$$

Here, ${}_0^C D_t^\alpha g(t)$ denotes the Caputo fractional derivative [10,11], defined for $0 < \alpha < 1$ (and considering a generic scalar function $g(t)$) as:

$${}_0^C D_t^\alpha g(t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t (t-s)^{-\alpha} g'(s)\, ds, \tag{2}$$

where Γ is the Gamma function.

An important feature of Neural FDEs is their ability to learn not only the optimal parameters $\boldsymbol{\theta}$ of the NN $\boldsymbol{f}_{\boldsymbol{\theta}}(t, \boldsymbol{h}(t))$, but also the order of the derivative α (when $\alpha = 1$ we obtain a Neural ODE). This is achieved using only information from the time-series dataset $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, where each $\boldsymbol{x}_i = (x_i^1, x_i^2, \ldots, x_i^d) \in \mathbb{R}^d$, $i = 0, \ldots, N$ is associated with a time instant t_i.

In [9] the α value is learned from another NN $\alpha_{\boldsymbol{\phi}}$ with parameters $\boldsymbol{\phi}$. Therefore, if $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$ represents the loss function, we can train the Neural FDE by solving the minimisation problem (3). The parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are optimised by minimising the error between the predicted $\hat{\boldsymbol{h}}(t_i)$, $i = 1, \ldots, N$ and ground-truth $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ values[1]:

$$\begin{aligned} \underset{\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}, \boldsymbol{\phi} \in \mathbb{R}^{n_\phi}}{\text{minimize}} \quad & \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{1}{N} \sum_{i=1}^{N} ||\hat{\boldsymbol{h}}(t_i) - \boldsymbol{x}_i||_2^2 \\ \text{subject to} \quad & \hat{\boldsymbol{h}}(t_i) = \text{FDESolve}(\alpha_{\boldsymbol{\phi}}, \boldsymbol{f}_{\boldsymbol{\theta}}, \boldsymbol{x}_0, \{t_0, t_1, \ldots, t_N\}), \end{aligned} \tag{3}$$

The popular Mean Squared Error (MSE) loss function was considered in [9] and also in this work. Here, $\text{FDESolve}(\boldsymbol{f}_{\boldsymbol{\theta}}, \boldsymbol{x}_0, \{t_0, t_1, \ldots, t_N\})$ refers to any numerical solver used to obtain the numerical solution $\hat{\boldsymbol{h}}(t)$ for each instant t_i.

Since Neural FDEs are a recent research topic, there are no studies on the uniqueness of the parameter α and its interaction with the NN $\boldsymbol{f}_{\boldsymbol{\theta}}(t, \boldsymbol{h}(t))$. In [9], the authors provide the values of α learned by Neural FDEs for each dataset, however, a closer examination reveals that these values differ significantly from the ground truth values, which were derived from synthetic datasets. The authors attribute this discrepancy to the approximation capabilities of NNs, meaning that, during training, $\boldsymbol{f}_{\boldsymbol{\theta}}(t, \boldsymbol{h}(t))$ adapts to any given α (this is a complex interaction since in [9], α is also learned by another NN). Additionally, α must be initialised in the optimisation procedure, and yet no studies have investigated how the initialisation of α affects the learned optimal α and the overall performance of Neural FDEs.

In this work, we address these key open questions about the order of the fractional derivative α in Neural FDEs. We show that Neural FDEs are capable of modelling data dynamics effectively, even when the learned value of α deviates significantly from the *true* value. Furthermore, we perform a numerical analysis to investigate how the initialisation of α affects the performance of Neural FDEs.

This paper is organised as follows: In Section 2, we provide a brief overview of FDEs and Neural FDEs, highlighting the theoretical results regarding the existence and uniqueness of solutions. We also discuss how the solution depends on the given data. Section 3 presents a series of numerical experiments on the non-uniqueness of the learnt α values. The paper ends with the discussion and conclusions in Section 4.

2. Neural Networks and Theory of Fractional Initial Value Problems

As shown in the introduction, a Neural FDE is composed of two NN:

[1] In this work we consider that the solution outputted by the solver is the predicted output, although this might not always be the case (e.g. image classification).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 July 2024 doi:10.20944/preprints202407.2572.v1

- a NN with parameters $\boldsymbol{\theta}$, denoted as $\boldsymbol{f_\theta}$, that models the right-hand side of a FDE,

$$ {}_0^C D_t^\alpha \boldsymbol{h}(t) = \boldsymbol{f_\theta}(t, \boldsymbol{h}(t)), \tag{4} $$

  where $\boldsymbol{h}(t)$ is the state of the system at time-step t;
- a NN with parameters $\boldsymbol{\phi}$ (or a learnable parameter), referred to as $\alpha_{\boldsymbol{\phi}}$, that models α,

$$ \alpha = \alpha_{\boldsymbol{\phi}}. \tag{5} $$

As shown in Figure 1 in this work we constructed $\boldsymbol{f_\theta}$ with 3 layers: input layer with 1 neuron and hyperbolic tangent (tanh) activation function; hidden layer with 64 neurons and tanh; output layer with 1 neuron. For learning α we considered a NN $\alpha_{\boldsymbol{\phi}}$ with 3 layers: input layer with 1 neuron and hyperbolic tangent (tanh) activation function; hidden layer with 32 neurons and tanh; output layer with 1 neuron and sigmoid activation function (that helps keeping the value of α in the interval $(0,1)$). For ease of understanding we consider $\hat{h}(t))$ to be a scalar in Figure 2 and in Equations (6) and (7). We use $\hat{h}(t))$ instead of $h(t)$, because it is assumed that $h(t)$ is being evaluated during the training of the Neural FDE, where a numerical solution is needed ($\hat{h}(t)$ is a numericall approximation of $h(t)$ [9]). After training, we obtain the final/optimal Neural FDE model and use again the notation $\boldsymbol{f_\theta}(t, h(t))$.

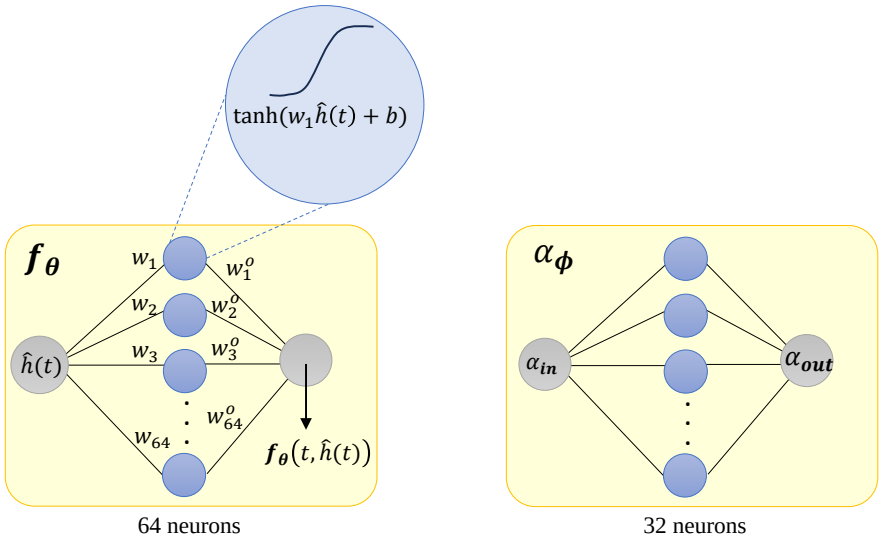

Figure 1. Schematic of the NNs ($\boldsymbol{f_\theta}$ and $\alpha_{\boldsymbol{\phi}}$) used in the Neural FDE. (Left) $\boldsymbol{f_\theta}$: The input $\hat{h}(t)$ refers to a single value while the output $\boldsymbol{f_\theta}(t, \hat{h}(t))$ refers to a NN that will approximate a continuous function $\boldsymbol{f}(t, h(t))$. w_i and w_i^o are the weights associated with the different layers. (Right) $\alpha_{\boldsymbol{\phi}}$: the value of α_{in} is initialised.

In the NN $\boldsymbol{f_\theta}(t, \hat{h}(t))$ (Figure 1), the values w_i and w_i^o for $i = 1, \ldots, 64$ are the weights of the hidden and output layers, respectively, and b is the bias, $\boldsymbol{\theta} = \{w_1, \ldots, w_{64}, w_1^o, \ldots, w_{64}^o, b\}$. The output of the NN $\boldsymbol{f_\theta}(t, \hat{h}(t))$ can be written as,

$$ w_1^o \tanh\left(w_1 \hat{h}(t) + b\right) + \ldots w_{64}^o \tanh\left(w_{64} \hat{h}(t) + b\right) = \sum_{i=1}^{64} w_i^o \tanh\left(w_i \hat{h}(t) + b\right). \tag{6} $$

An important feature is that there is no activation function in the output layer of $\boldsymbol{f_\theta}(t, \hat{h}(t))$, allowing the NN to approximate any function $f(t, h(t))$. If we opt to use, for example, a tanh in the output layer, it would constrain the fractional derivative ${}_0^C D_t^\alpha h(t)$ to vary only from -1 to 1, thus limiting the fitting capabilities of the Neural FDE. This limitation could be mitigated by normalising the given data $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$.

 doi:10.20944/preprints202407.2572.v1

Remark: In Figure 1 and Equation (6), we observe h as a function of t. However, the NN depicted on the left side of Figure 1 does not use t as an input. Instead, the NN is called at each iteration of the numerical solver that addresses a discretised version of Equation (1). This solver defines all time instants through a mesh over the interval $[t_0, T]$ [9], and, consequently, each evaluation of $\boldsymbol{f_\theta}$ is always associated with a specific time instant.

Since for a differentiable function $g(h)$ with continuous derivative on $[h_1, h_2]$ we have that (using the mean value Theorem),

$$|g(h_1) - g(h_2)| \leq \max_{h \in [h_1, h_2]} |g'(h)||h_1 - h_2|$$

and $|\tanh'(h)| = |1 - \tanh^2(h)| \leq 1$ ($-1 \leq \tanh(h) \leq 1$), we can say that tanh is 1-Lipschitz, that is,

$$|\tanh(h_1) - \tanh(h_2)| \leq 1\,|h_1 - h_2|.$$

Define $g(h)$ as,

$$g(h) = \sum_{i=1}^{64} w_i^o \tanh(w_i h + b), \tag{7}$$

which is a weighted sum of 64 tanh functions. Then, $g(h)$ is L-Lipschitz, with $L < 64 w^o_{\max}$ ($w^o_{max} = \max\{w^o_1, \ldots, w^o_{64}\}$). This property will be important to guarantee the uniqueness of the solution of (1).

2.1. Fractional Differential Equations

We now provide some theoretical results that will be fundamental in understanding the expected behaviour of the Neural FDE model. Consider the following fractional initial value problem,

$$\begin{cases} {}_0^C D_t^\alpha z(t) = f(t, z(t)) \\ z(t_0) = z_0. \end{cases} \tag{8}$$

For ease of understanding we restrict the analyses to the case where $z(t)$ is a scalar function and $0 < \alpha < 1$.

2.1.1. Existence and Uniqueness

The following Theorem [11,12] provides information on the existence of a continuous solution to problem (8):

Theorem 1. *Let $0 < \alpha$, $x_0 \in \mathbb{R}$, $Q > 0$ and $T^* > 0$. Define $G = [0, T^*] \times [x_0 - Q, x_0 + Q]$, and let the function $f : G \to \mathbb{R}$ be continuous. Furthermore, we define $M := \sup_{(x,y) \in G} |f(x, y)|$ and*

$$T := \begin{cases} T^*, & \text{if } M = 0 \\ \min\left\{T^*, \left(\frac{Q\Gamma(\alpha+1)}{M}\right)^{1/\alpha}\right\} & \text{otherwise} \end{cases}$$

Then, there exists a function $z(t) \in C^0[0, T]$ solving the initial value problem (8).

Note that this continuous solution may be defined in a smaller interval $[0, T]$ compared to the interval $[0, T^*]$ where the function $f(t, z(t))$ is defined ($T \leq T^*$). From Theorem (1), we can infer that high values of $|\boldsymbol{f_\theta}(t, \boldsymbol{h}(t))|$ (see Equation (1)) decrease the interval within which we can guarantee a continuous solution. However, these conclusions should be approached with caution. As will be shown later, we will only access discrete values of the function $\boldsymbol{f_\theta}(t, \boldsymbol{h}(t))$ in the numerical solution of (1). Note that α also affects the size of the interval. Its contribution can be either positive or negative, depending on the values of Q and M.

 doi:10.20944/preprints202407.2572.v1

The following Theorem establishes the conditions for which we can guarantee the uniqueness of the solution $z(t)$ [11,12]:

Theorem 2. *Let $0 < \alpha$, $x_0 \in \mathbb{R}$, and $T^* > 0$. Define the set $G = [0, T^*] \times [x_0 - Q, x_0 + Q]$ and let the function $f : G \to \mathbb{R}$ be continuous and satisfy a Lipschitz condition with respect to the second variable:*

$$|f(t, z_1) - f(t, z_2)| \leq L|z_1 - z_2|,$$

where $L > 0$ is a constant independent of t, z_1, and z_2. Then, there exists a uniquely defined function $z \in C^0[0, T]$ solving the initial value problem (8).

As shown above, the function on the right-hand side of the Neural FDE model is Lipschitz (see Equation (7)). Therefore, we can conclude that the solution to Equation (1) is unique.

2.1.2. Analysing the Behaviour of Solutions with Perturbed Data

Other results of interest for Neural FDEs pertain to the dependencies of the solution on $f(t, z(t))$ and α. In Neural FDEs, both $f(t, z(t))$ and α are substituted by NNs that vary with each iteration of the Neural FDE training [9].

Let $u(t)$ be the solution of the initial value problem,

$$\begin{cases} {}_0^C D_t^\alpha u(t) = \tilde{f}(t, u(t)) \\ u(t_0) = z_0. \end{cases} \tag{9}$$

where $\tilde{f}(t, u(t))$ is a perturbed version of f, that satisfies the same hypotheses as f.

Theorem 3. *Let*

$$\varepsilon := \max_{(x_1, x_2) \in G} \left| f(x_1, x_2) - \tilde{f}(x_1, x_2) \right|.$$

If ε is sufficiently small, there exists some $T > 0$ such that both functions z (Equation (8)) and u (Equation (9)) are defined on $[0, T]$, and we have

$$\sup_{0 \leq t \leq T} |z(t) - u(t)| = O\left(\max_{(x_1, x_2) \in G} \left| f(x_1, x_2) - \tilde{f}(x_1, x_2) \right| \right),$$

where $z(t)$ is the solution of (8).

This Theorem provides insight into how the solution of (1) changes in response to variations in the NN $\boldsymbol{f_\theta}(t, \boldsymbol{h}(t))$. While the variations in both the solution and the function are of the same order for small changes of the function, it is crucial to carefully interpret these results given the NN $\boldsymbol{f_\theta}(t, \boldsymbol{h}(t))$ as defined by Equation (7).

When training the Neural ODE, one must solve the optimisation problem (3), where the weights and biases are adjusted until an optimal Neural FDE model is obtained (training 1). If a second, independent training (training 2) of the Neural FDE is conducted with the same stopping criterion for the optimisation process, a new Neural FDE model with different weights and biases may be obtained.

The NN learns model parameters based on a set of ordered data, meaning the number of elements in the set will significantly influence the difference between $z(t)$ and $u(t)$ as in Theorem (3). This effect is illustrated in Figure 2, where a training dataset of only two observations can be fitted by two distinct functions.

Therefore, Figure 2 tells us that when modelling with Neural FDEs, it is important to have some prior knowledge of the underlying physics of the data. This is crucial because the number of data points available for the training process might be beyond our control. For instance, the data could originate from real experiments where obtaining results is challenging.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 July 2024 doi:10.20944/preprints202407.2572.v1

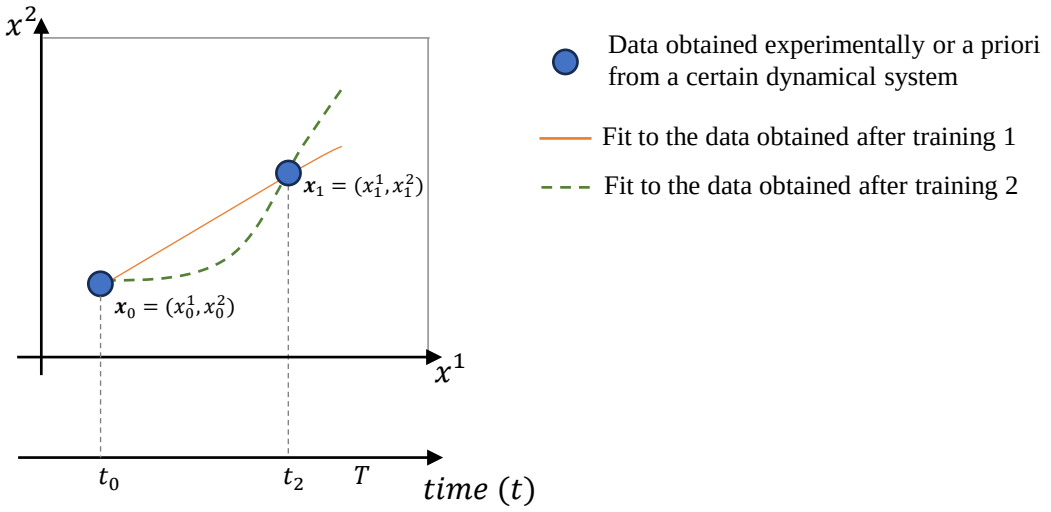

Figure 2. Schematic of a Neural FDE model fitted to data, considering two different training runs.

Regarding the influence of the order of the derivative on the solution, we have that [11,12]:

Theorem 4. *Let $u(t)$ be the solution of the initial value problem,*

$$\begin{cases} {}_0^C D_t^{\tilde{\alpha}} u(t) = f(t, u(t)) \\ u(t_0) = z_0. \end{cases} \tag{10}$$

where $\tilde{\alpha}$ is a perturbed α value. Let $\varepsilon := \tilde{\alpha} - \alpha$. Then, if ε is sufficiently small, there exists some $T > 0$ such that both the functions u and z are defined on $[0, T]$, and we have that

$$\sup_{0 \le t \le T} |u(t) - z(t)| = O(\tilde{\alpha} - \alpha)$$

where $z(t)$ is the solution of (8).

Once again, for small changes in α, the variations in both the solution and the order of the derivative are of the same order. This property will be explored numerically, and with more detail, later in this work when solving (3). It is important to note that the NN $f_{\boldsymbol{\theta}}(t, \boldsymbol{h}(t))$ is not fixed in our problem (1) (its structure is fixed, but the weights and bias change along the training optimisation procedure). However, Theorem (4) assumes that the function $f(t, u(t))$ is fixed. Therefore, Theorem (4) gives us an idea of the changes in our solution, but does not allow the full understanding of its variation along the optimisation procedure.

2.1.3. Smoothness of the Solution

For the classical case, $\alpha = 1$ in Equation (8), we have (under some hypotheses on the interval $[a, b]$) that if $f \in C^{k-1}([a, b])$, then $z(t)$ is k times differentiable.

For the fractional case, even if $f \in C^\infty$, it may happen that $z(t) \notin C^1$. This means that the solutions may not behave well, and solutions with singular derivatives are quite common. See [13] for more results on the smoothness properties of solutions to fractional initial value problems.

These smoothness properties (or lack of smoothness) make it difficult for numerical methods to provide fast and accurate solutions for (1), thus making Neural FDEs more difficult to handle compared to Neural ODEs. It should be highlighted that during the learning process, the NN $f_{\boldsymbol{\theta}}(t, \boldsymbol{h}(t))$ will always adjust the Neural FDE model to the data, independent of the *amount* of error obtained in the numerical solution of the FDE.

 doi:10.20944/preprints202407.2572.v1

3. Neural Fractional Differential Equations and the Learned Order of the Derivative - α

In Neural FDEs, the goal is to model systems where the order of the derivative, α, is a parameter that needs to be learned along with the parameters of a NN $f_{\boldsymbol{\theta}}(t, \boldsymbol{h}(t))$. A key challenge in this approach is the potential variability and non-uniqueness of the learned α values, especially when the ground-truth α used to generate synthetic datasets is known a priori. This variability arises from the highly flexible nature of NNs, which can adjust their parameters to fit the data well, regardless of the specific value of α.

3.1. Numerical Experiments

The results obtained in [9] propose the existence of parameters $\alpha_1, \alpha_2, \ldots, \alpha_n$ and corresponding parameter vectors $\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, \ldots, \boldsymbol{\theta_n}$, that satisfy the following condition:

$$\mathcal{L}(\boldsymbol{\theta}_i, \alpha_i) \approx 0 \quad \text{for} \quad i = 1, 2, \ldots, n. \tag{11}$$

In theory, this implies that for each i, the loss function $\mathcal{L}$ can converge to zero.

There are some results in the literature on the universal approximation capabilities of Neural ODEs and ResNets. Interested readers should consult [14,15] for more information.

Based on these observations, we now conduct numerical experiments to observe in detail the practical outcomes. Specifically, we employ a Neural FDE to model population dynamics with different ground-truth α values. Our goal is to examine how different initialisations of α impact the final learned α values.

We analyse the α values learned across multiple runs and observe how α evolves during training. Additionally, we fix the α value and allow the Neural FDE to learn only the right-hand side of the FDE, comparing the loss for different α values.

Consider a population of organisms that follows a fractional-order logistic growth. The population size $P(t)$ at time t is governed by the following FDE of order α:

$${}_0^C D_t^\alpha = rP(t)\left(1 - \frac{P(t)}{K}\right), \tag{12}$$

with initial condition $P(0) = P(t_0) = 100$, a growth-rate $r = 0.1$ and carrying capacity $K = 1000$.

To conduct the experiments, three training datasets were generated by numerically solving (12) over the interval $t \in (0, 10)$ with a step size of 0.1 for three different values of α, namely $\alpha = 0.3, 0.5, 0.99$. These datasets are denoted as $P_{\alpha=0.3}$, $P_{\alpha=0.5}$, and $P_{\alpha=0.99}$, respectively. For the experiments, Adam optimiser [16] was used with a starting learning rate of 1E-03.

3.1.1. Case Study 1: varying α and fixed number of iterations

The generated datasets were used to train the Neural FDE model (see Equation (1)) with four different initialisations of the derivative order: $\alpha = 0.1$, 0.3, 0.5, and 0.99. For each initialisation, three independent runs were conducted considering 200 iterations.

The results obtained for each dataset, $P_{\alpha=0.3}$, $P_{\alpha=0.5}$, and $P_{\alpha=0.99}$, using the various α initialisation values are presented in Tables 1-3 and the evolution of the training losses and α values can be visually observed in Figures 3, A1-A2 and Figures 4, A3, A4, respectively.

Table 1. Learned α and training loss obtained for three runs of a Neural FDE model, considering different α initialisations. Case $P_{\alpha=0.3}$.

α Initialisation	Learnt α	Training Loss
0.1	0.3498 | 0.3417 | 0.2639	3.60E-04 | 2.50E-04 | 3.89E-04
0.3	0.2248 | 0.478 | 0.4374	1.44E-04 | 8.40E-05 | 8.00E-05
0.5	0.3507 | 0.3878 | 0.2921	3.90E-04 | 1.67E-04 | 3.53E-04
0.99	0.4427 | 0.3367 | 0.4497	3.50E-05 | 1.30E-04 | 6.00E-06

 doi:10.20944/preprints202407.2572.v1

Table 2. Learned α and training loss obtained for three runs of a Neural FDE model, considering different α initialisations. Case $P_{\alpha=0.5}$.

α **Initialisation**	**Learnt** α	**Training Loss**
0.1	0.2927 | 0.4924 | 0.4873	1.36E-03 | 2.00E-06 | 5.20E-05
0.3	0.455 | 0.4744 | 0.3923	3.00E-06 | 6.00E-06 | 9.52E-04
0.5	0.5162 | 0.2955 | 0.468	6.76E-04 | 1.02E-03 | 3.17E-04
0.99	0.4191 | 0.5372 | 0.503	9.20E-05 | 5.00E-06 | 3.00E-06

Table 3. Learned α and training loss obtained for three runs of a Neural FDE model, considering different α initialisations. Case $P_{\alpha=0.99}$.

α **Initialisation**	**Learnt** α	**Training Loss**
0.1	0.6216 | 0.5173 | 0.5407	7.00E-06 | 3.11E-03 | 1.26E-03
0.3	0.2738 | 0.5429 | 0.5364	7.48E-03 | 4.00E-06 | 8.73E-04
0.5	0.518 | 0.5586 | 0.5876	5.00E-06 | 1.00E-05 | 8.00E-06
0.99	0.5652 | 0.5141 | 0.5666	1.60E-05 | 4.92E-04 | 1.00E-05

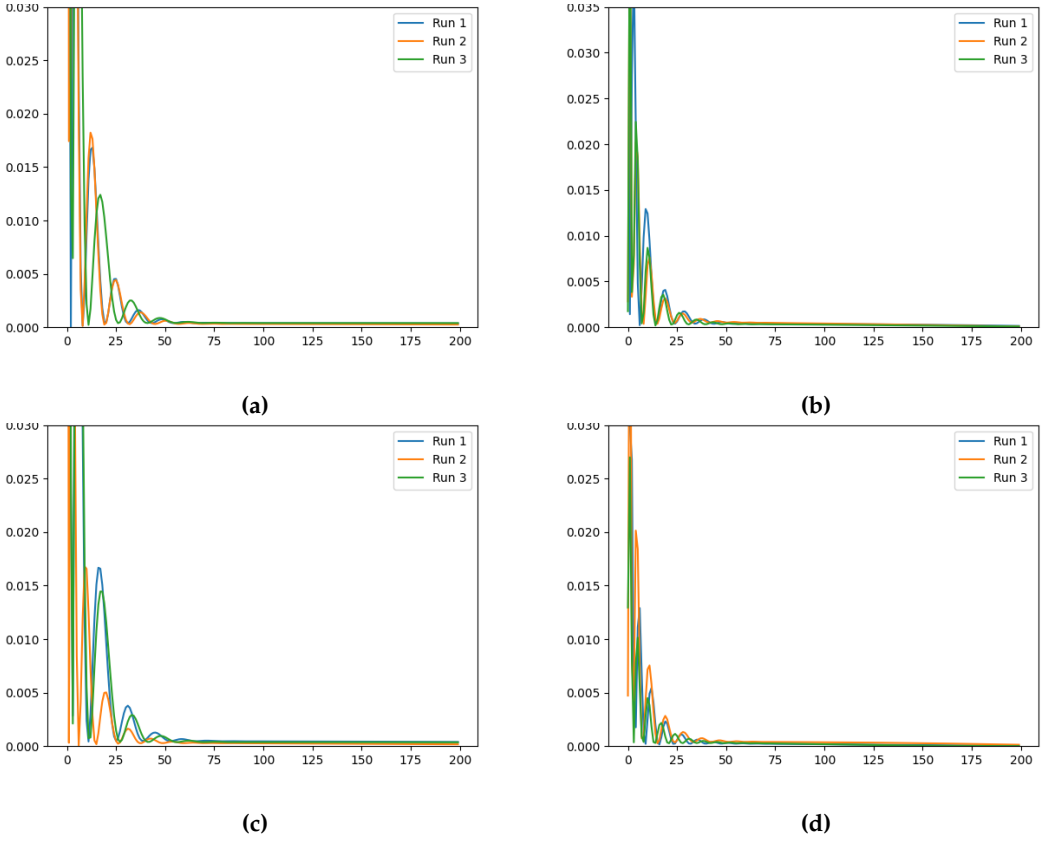

Figure 3. Training loss evolution for Neural FDE when modelling $P_{\alpha=0.3}$ for (a) $\alpha = 0.1$, (b) $\alpha = 0.3$, (c) $\alpha = 0.5$ and (d) $\alpha = 0.99$ initialisation (Loss (vertical axis) vs Number of Iterations (horizontal axis)).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 July 2024 doi:10.20944/preprints202407.2572.v1

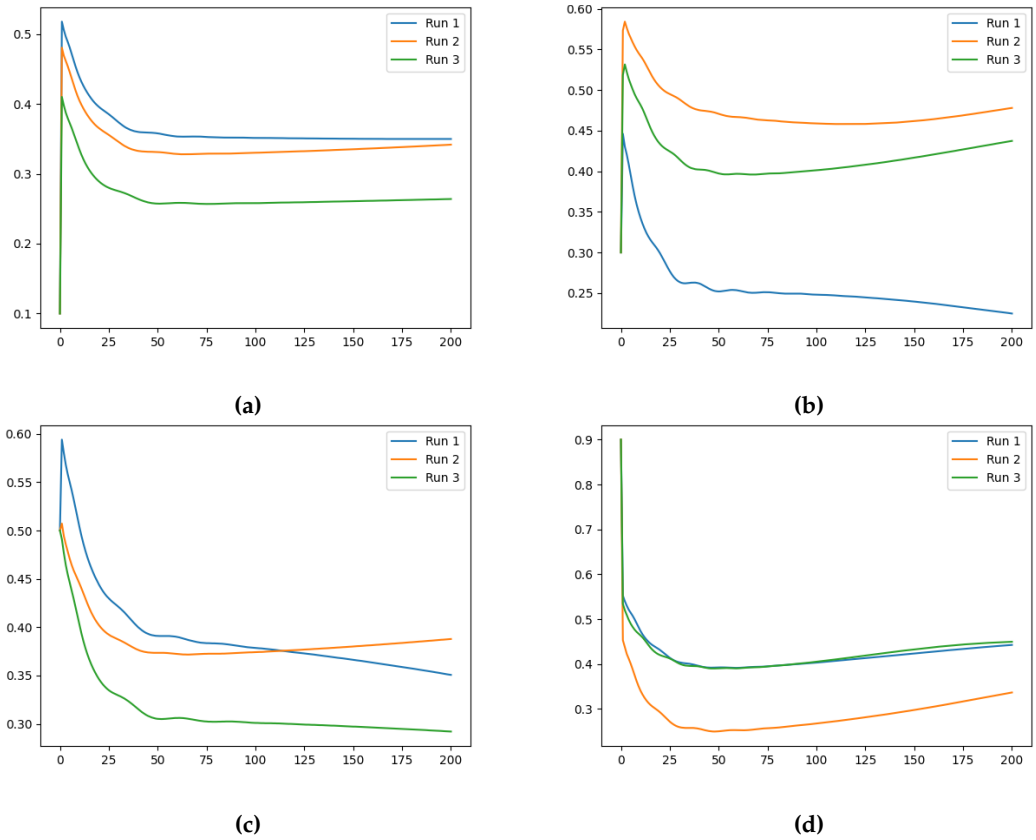

Figure 4. Evolution of α along the iterations. Case $P_{\alpha=0.3}$ for (a) $\alpha = 0.1$, (b) $\alpha = 0.3$, (c) $\alpha = 0.5$ and (d) $\alpha = 0.99$ initialisation (α (vertical axis) vs Number of Iterations (horizontal axis)).

The results presented in Tables 1 through 3 indicate that the initial value of α does not significantly impact the final learned value of α. While there is some effect, it is challenging to discern a clear pattern, and the observed variations fall within a certain range of randomness. However, the ground-truth value of α in the datasets influences the learnt value, with higher ground-truth values leading to higher learnt values. For example, in Table 3, all learnt α values are above 0.50, whereas for other datasets, the learnt α values are lower. This is expected since the Neural FDE model should fit the given data (that depends on α).

Furthermore, the results demonstrate the approximation capabilities of NNs, which allow them to model the data dynamics with low errors even when the learnt α is far from the ground-truth value. For example, in Table 1, the ground-truth α is 0.3, but the lowest training loss was achieved with a learnt α of 0.4497. Similarly, in Table 3, the ground-truth α is 0.99, but the lowest training loss values were achieved with learnt α of 0.518 and 0.5876. It is interesting to note that the values of α are significantly different from 1. The results obtained suggest that the optimisation procedure may be placing greater importance on the NN $\boldsymbol{f}_{\boldsymbol{\theta}}(t, \boldsymbol{h}(t))$.

The evolution of the loss and the learned α values along training (Figures 3 and 4, respectively, for the case $P_{\alpha=0.3}$), indicate that the α initialisation and the ground-truth α do not significantly influence the Neural FDE training process. In Figure 4 a pattern is observed. Initially the α values increase drastically, and then slowly decrease approaching a plateau. In some cases, after around 100 iterations, the values begin to increase again. This behaviour results from the complex interplay between the two different NNs. The behaviour is even more irregular for the cases $P_{\alpha=0.5}, P_{\alpha=0.99}$, as shown in Appendix A.

 doi:10.20944/preprints202407.2572.v1

3.1.2. Case Study 2: fixed α and fixed number of iterations

To demonstrate the effectiveness of Neural FDEs in minimising the loss function $\mathcal{L}(\boldsymbol{\theta}, \alpha)$ for various α values, we performed experiments with fixed α values of 0.1, 0.3, 0.5 and 0.99. In these experiments, the Neural FDEs used exclusively the NN $\boldsymbol{f}_{\boldsymbol{\theta}}$, to fit the solution $\boldsymbol{h}(t)$ to the three distinct datasets $P_{\alpha=0.3}$, $P_{\alpha=0.5}$, $P_{\alpha=0.99}$. The experimental conditions remained the same, except that α was not learnable. Note that the stopping criterion was set to a fixed number of iterations, specifically 200.

The final training losses obtained for three runs, for each fixed α, are summarised in Tables 4-6. The evolution of the training losses can be visualised in Figures 5-7.

Table 4. Final training loss for three runs of the Neural FDE (fixed α). Case $P_{\alpha=0.3}$.

Fixed α	**Training Loss**
0.1	3.50E-05 | 4.50E-05 | 4.90E-05
0.3	7.70E-05 | 2.44E-04 | 1.00E-06
0.5	4.36E-04 | 1.00E-05 | 4.30E-05
0.99	3.60E-05 | 2.10E-05 | 4.00E-05

Table 5. Final training loss for three runs of the Neural FDE (fixed α). Case $P_{\alpha=0.5}$.

Fixed α	**Training Loss**
0.1	4.53E-04 | 3.78E-04 | 4.27E-04
0.3	1.00E-06 | 1.70E-03 | 3.00E-06
0.5	1.70E-04 | 3.00E-06 | 4.00E-06
0.99	3.60E-04 | 6.70E-05 | 1.00E-03

Table 6. Final training loss for three runs of the Neural FDE (fixed α). Case $P_{\alpha=0.99}$.

Fixed α	**Training Loss**
0.1	5.44E-03 | 3.10E-03 | 5.22E-03
0.3	1.02E-02 | 8.68E-03 | 7.37E-03
0.5	4.00E-06 | 6.10E-05 | 7.05E-04
0.99	2.00E-06 | 8.18E-03 | 1.00E-08

 doi:10.20944/preprints202407.2572.v1

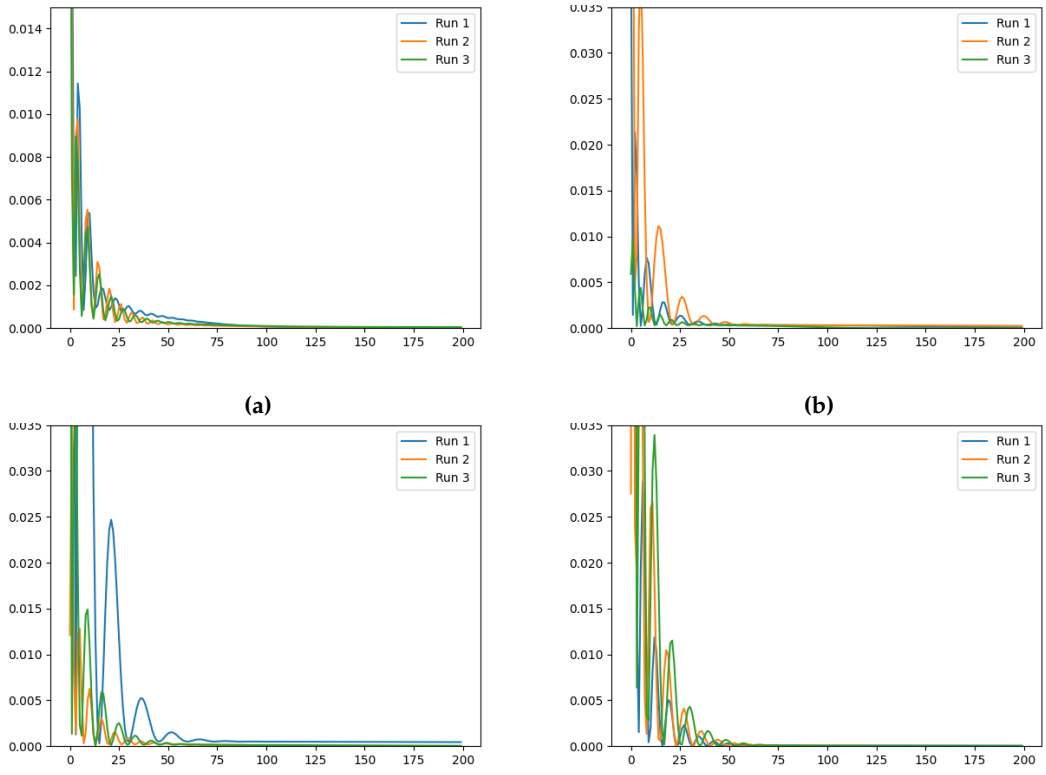

Figure 5. Training loss evolution of the Neural FDE when modelling $P_{\alpha=0.3}$ for a fixed (a) $\alpha = 0.1$, (b) $\alpha = 0.3$, (c) $\alpha = 0.5$ and (d) $\alpha = 0.99$ (Loss (vertical axis) vs Number of Iterations (horizontal axis)).

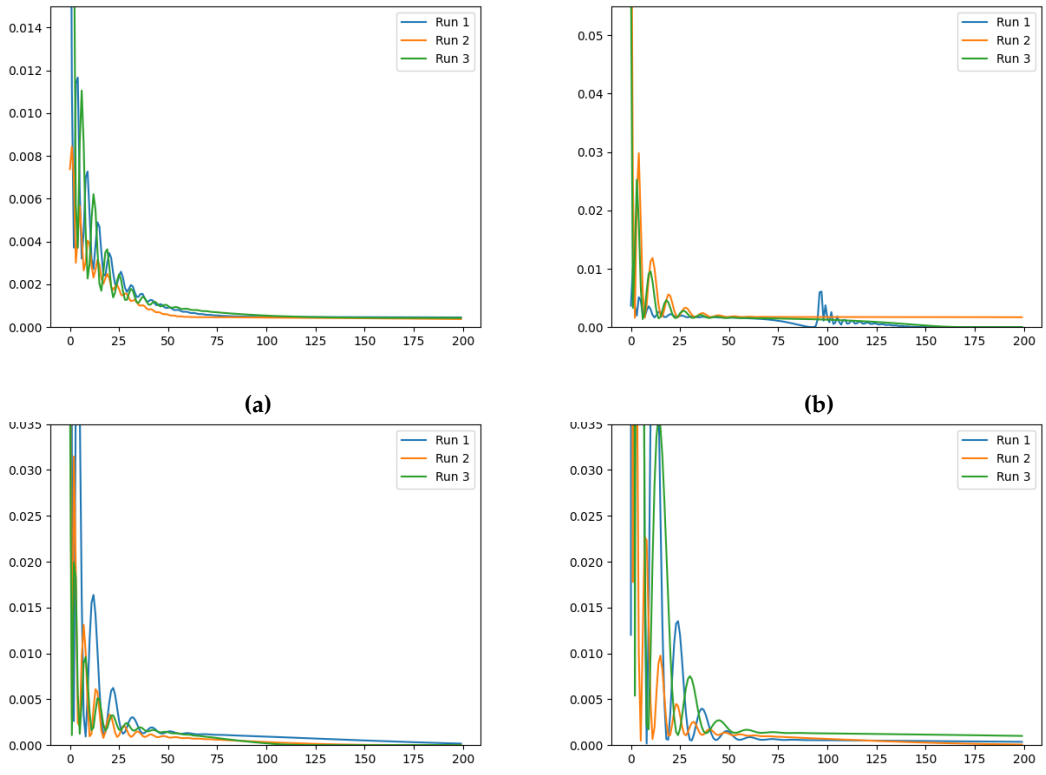

Figure 6. Training loss evolution of the Neural FDE when modelling $P_{\alpha=0.5}$ for a fixed (a) $\alpha = 0.1$, (b) $\alpha = 0.3$, (c) $\alpha = 0.5$ and (d) $\alpha = 0.99$ (Loss (vertical axis) vs Number of Iterations (horizontal axis)).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 July 2024 doi:10.20944/preprints202407.2572.v1

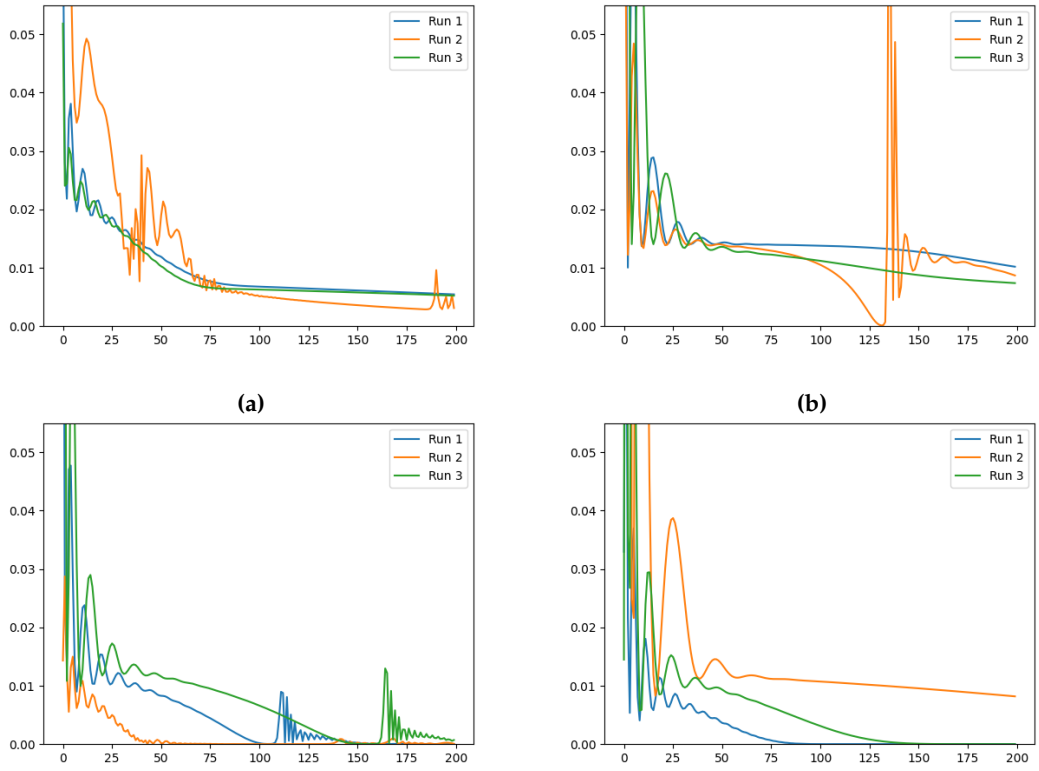

Figure 7. Training loss evolution of the Neural FDE when modelling $P_{\alpha=0.99}$ for a fixed (a) $\alpha = 0.1$, (b) $\alpha = 0.3$, (c) $\alpha = 0.5$ and (d) $\alpha = 0.99$ (Loss (vertical axis) vs Number of Iterations (horizontal axis)).

The results in Tables 4-6 show that the final training loss values are generally similar for different values of α. Even when the fixed α matches the ground-truth α of the dataset, in general, the final loss values are comparable to other fixed α values.

The training losses evolution in Figures 5-7 show similar behaviour for the different fixed α values.

3.1.3. Case Study 3: fixed α and fixed loss threshold

As a final experiment, we aimed to numerically demonstrate that Neural FDEs are capable of achieving the same fit to the data independent of the α order. To unequivocally show this, we modified the stopping criteria of our Neural FDE training from a maximum number of iterations to a loss threshold of 1E-05. This approach ensures that the Neural FDEs are trained until the threshold is achieved, allowing us to demonstrate that they can reach the same loss values regardless of the α value.

In this experiment, we performed one run for each dataset $P_{\alpha=0.3}$, $P_{\alpha=0.5}$, and $P_{\alpha=0.99}$ with fixed α values of 0.1, 0.3, 0.5, and 0.99. The results are organised in Tables 7-9

Table 7. Number of iterations needed to achieve a final training loss of 1E-05, with the different fixed α values, when modelling $P_{\alpha=0.3}$.

Fixed α	Number of Iterations
0.1	791 ± 90
0.3	934 ± 76
0.5	265 ± 2
0.99	1035 ± 462

 doi:10.20944/preprints202407.2572.v1

Table 8. Number of iterations needed to achieve a final training loss of 1E-05, with the different fixed α values, when modelling $P_{\alpha=0.5}$.

Fixed α	Number of Iterations
0.1	2837 ± 1697
0.3	4659 ± 1114
0.5	193 ± 47
0.99	3255 ± 2003

Table 9. Number of iterations needed to achieve a final training loss of 1E-05, with the different fixed α values, when modelling $P_{\alpha=0.99}$.

Fixed α	Number of Iterations
0.1	-*
0.3	3119 ± 460
0.5	128 ± 28
0.99	240 ± 137

* In 10000 iterations it was not possible to achieve a final training loss of 1e-05 in both runs. However the final training loss was 1.2e-05.

The results presented in Tables 7 through 9 show that the NN $f_{\boldsymbol{\theta}}$ is capable of optimising the parameters $\boldsymbol{\theta}$ to accurately model the Neural FDE to the data dynamics, regardless of whether the imposed value of α is close to or far from the ground truth.

These results might lead some to believe that changing α does not impact memory. They might also suggest that the Neural FDE can model systems with varying memory levels, independent of α. Although, this only happens because the NN $f_{\boldsymbol{\theta}}(t, \boldsymbol{h}(t))$, which models the FDE's right-hand side, adjusts its parameters θ to fit the FDE for any α value (effectively mimicking the memory effects associated with α). Therefore, Neural FDEs do not need a unique α value for each dataset. Instead, they can work with an infinite number of α values to fit the data. This flexibility is beneficial when fitting to given data is required, and the underlying physics is not known.

4. Conclusions

In this work, we present the theory of fractional initial value problems and explore its connection with Neural Fractional Differential Equations (Neural FDEs). We analyse both theoretically and numerically how the solution of a Neural FDE is influenced by two factors: the NN $f_{\boldsymbol{\theta}}$, which models the right-hand side of the FDE, and the NN that learns the order of the derivative.

We also investigate the numerical evolution of the order of the derivative (α) and training loss across several iterations, considering different initialisations of the α value. For this experiment, with a fixed number of 200 iterations, we create three synthetic datasets for a fractional initial value problem with ground-truth α values of 0.3, 0.5, and 0.99. We test four different initialisations for α: 0.1, 0.3, 0.5, and 0.99. The results indicate that both the initial α value and the ground-truth α have minimal impact on the Neural FDE training process. Initially, the α values increase sharply and then slowly decrease towards a plateau. In some cases, around 100 iterations, the values begin to rise again. This behaviour results from the complex interaction between the two NNs, and it is particularly irregular for $\alpha = 0.5$ and $\alpha = 0.99$. The loss values achieved are low across all cases.

We then repeated the experiments, fixing the α value, meaning there was no initialisation of α, and the only parameters changing in the minimisation problem were those of the NN $f_{\boldsymbol{\theta}}(t, \boldsymbol{h}(t))$. The results confirm that the final training loss values are generally similar across different fixed values of α. Even when the fixed α matches the ground-truth α of the dataset, the final loss remains comparable to other fixed α values.

 doi:10.20944/preprints202407.2572.v1

In a final experiment, we modified the stopping criteria of our Neural FDE training from a maximum number of iterations to a loss threshold. This ensures that the Neural FDEs are trained until the loss threshold is achieved, demonstrating that they can reach similar loss values regardless of the α value. We conclude that, f_θ struggles to adjust its parameters θ to fit the FDE to the data for any given derivative order. Consequently, Neural FDEs do not require a unique α value for each dataset. Instead, they can use a wide range of α values to fit the data, suggesting that f_θ is a universal approximator.

If we train the model using data points obtained from an unknown experiment, then the flexibility of the Neural FDE proves to be an effective method for obtaining intermediate information about the system, provided the dataset contains sufficient information. If the physics involved in the given data is known, it is recommended to incorporate this knowledge into the loss function. This additional information helps to improve the extrapolation of results.

Funding: This work was also financially supported by national funds through the FCT/MCTES (PIDDAC), under the project 2022.06672.PTDC - iMAD - Improving the Modelling of Anomalous Diffusion and Viscoelasticity: solutions to industrial problems.

Acknowledgments: The authors acknowledge the funding by Fundação para a Ciência e Tecnologia (Portuguese Foundation for Science and Technology) through CMAT projects UIDB/00013/2020 and UIDP/00013/2020 and the funding by FCT and Google Cloud partnership through projects CPCA-IAC/AV/589164/2023 and CPCA-IAC/AF/589140/2023. C. Coelho would like to thank FCT the funding through the scholarship with reference 2021.05201.BD.

Appendix A. Evolution of loss and α along training

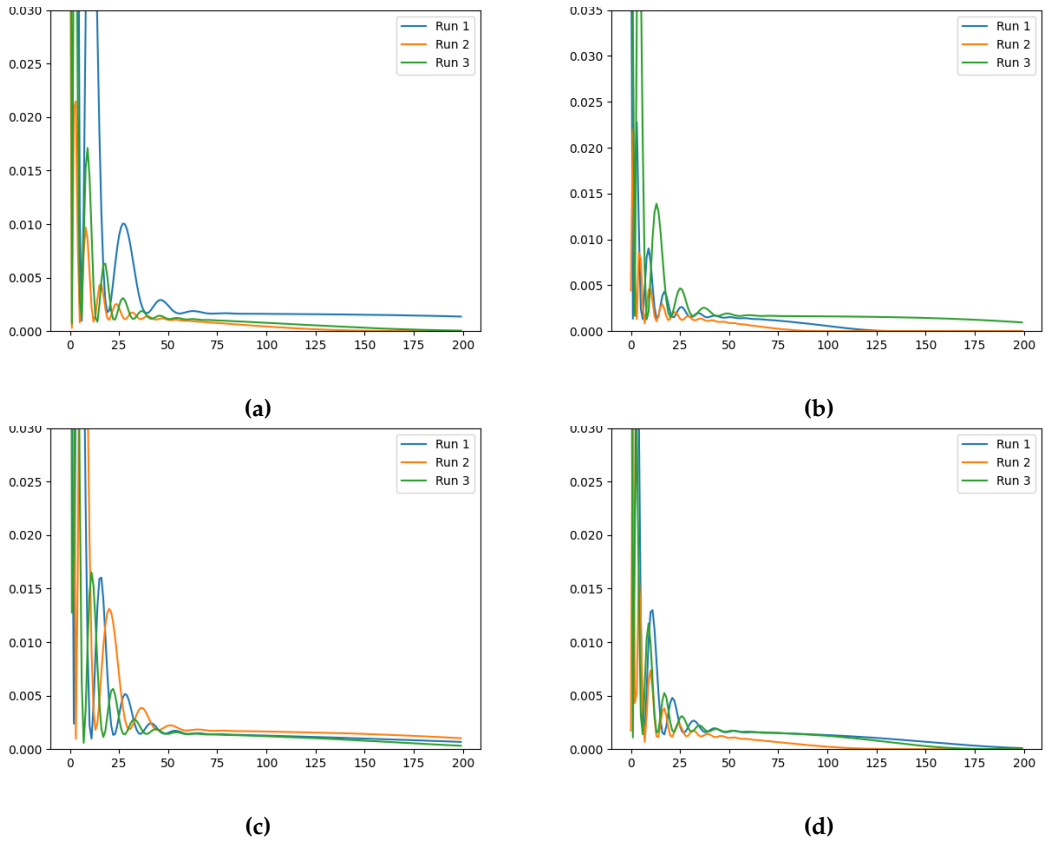

Figure A1. Training loss evolution for Neural FDE when modelling $P_{\alpha=0.5}$ for (a) $\alpha = 0.1$, (b) $\alpha = 0.3$, (c) $\alpha = 0.5$ and (d) $\alpha = 0.99$ initialisation (Loss (vertical axis) vs Number of Iterations (horizontal axis)).

 doi:10.20944/preprints202407.2572.v1

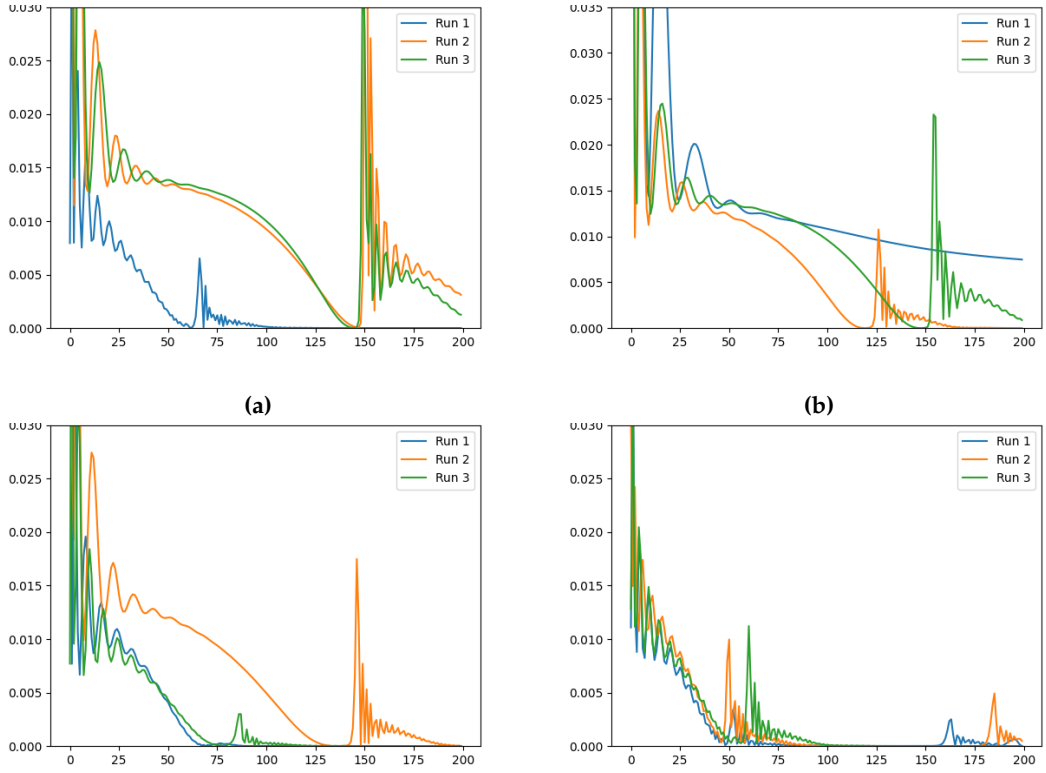

Figure A2. Training loss evolution for Neural FDE when modelling $P_{\alpha=0.99}$ for (a) $\alpha = 0.1$, (b) $\alpha = 0.3$, (c) $\alpha = 0.5$ and (d) $\alpha = 0.99$ initialisation (Loss (vertical axis) vs Number of Iterations (horizontal axis)).

 doi:10.20944/preprints202407.2572.v1

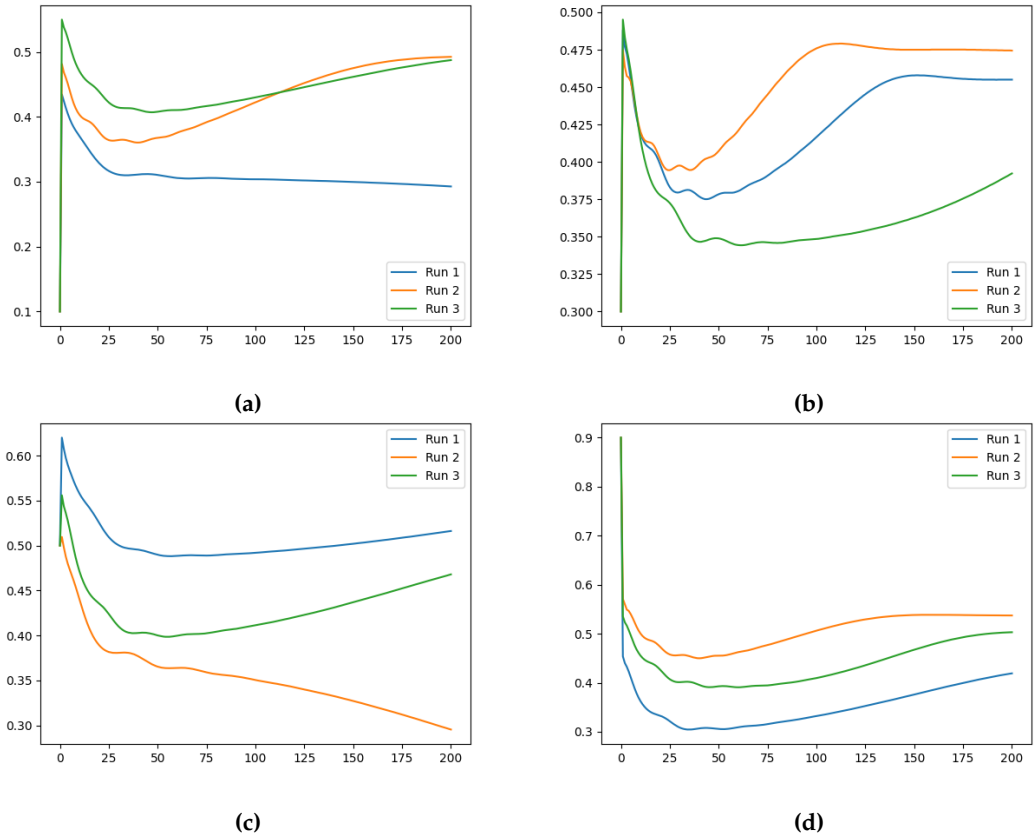

Figure A3. Evolution of α along the iterations. Case $P_{\alpha=0.5}$ for (a) $\alpha = 0.1$, (b) $\alpha = 0.3$, (c) $\alpha = 0.5$ and (d) $\alpha = 0.99$ initialisation (α (vertical axis) vs Number of Iterations (horizontal axis)).

 doi:10.20944/preprints202407.2572.v1

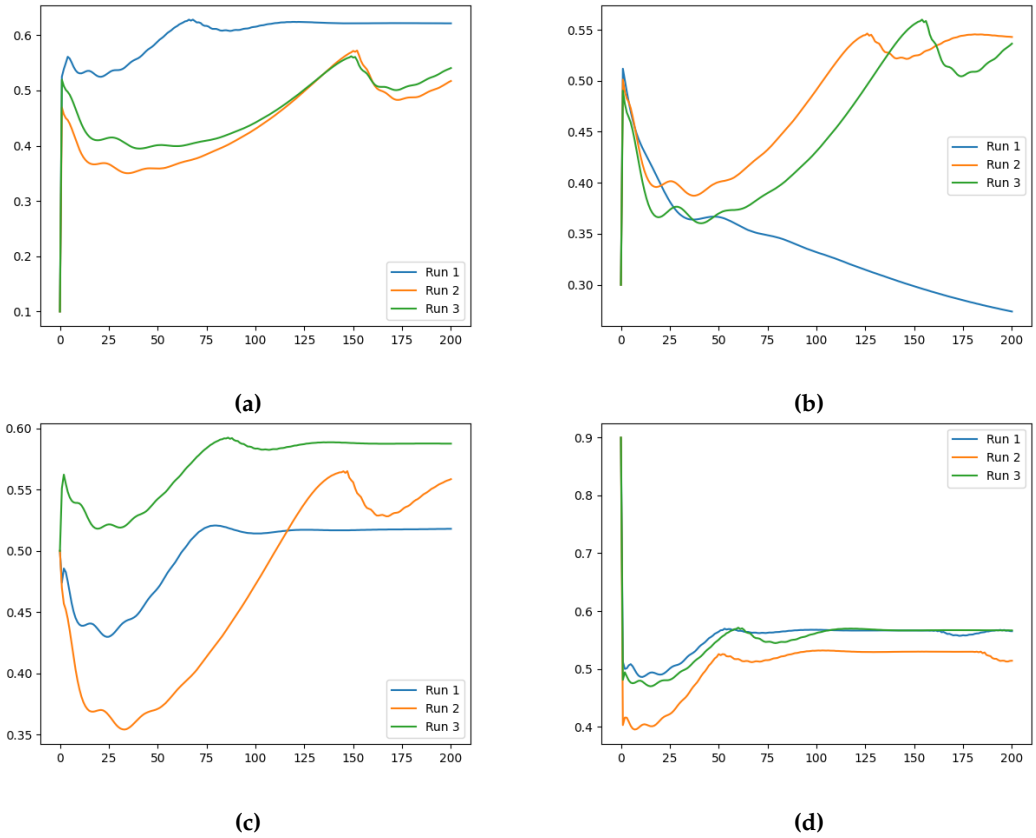

Figure A4. Evolution of α along the iterations. Case $P_{\alpha=0.99}$ for (a) $\alpha = 0.1$, (b) $\alpha = 0.3$, (c) $\alpha = 0.5$ and (d) $\alpha = 0.99$ initialisation (α (vertical axis) vs Number of Iterations (horizontal axis)).

References

1. Raissi, M.; Perdikaris, P.; Karniadakis, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* **2019**, *378*, 686–707. doi:10.1016/j.jcp.2018.10.045.
2. Salami, E.; Salari, M.; Ehteshami, M.; Bidokhti, N.; Ghadimi, H. Application of artificial neural networks and mathematical modeling for the prediction of water quality variables (case study: southwest of Iran). *Desalination and Water Treatment* **2016**, *57*, 27073–27084.
3. Jin, C.; Li, Y. Cryptocurrency Price Prediction Using Frequency Decomposition and Deep Learning. *Fractal and Fractional* **2023**, *7*, 708.
4. Ramadevi, B.; Kasi, V.R.; Bingi, K. Hybrid LSTM-Based Fractional-Order Neural Network for Jeju Island's Wind Farm Power Forecasting. *Fractal and Fractional* **2024**, *8*, 149.
5. Chen, R.T.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D.K. Neural ordinary differential equations. *Advances in neural information processing systems* **2018**, *31*.
6. Massaroli, S.; Poli, M.; Park, J.; Yamashita, A.; Asama, H. Dissecting neural odes. *Advances in Neural Information Processing Systems* **2020**, *33*, 3952–3963.
7. Dupont, E.; Doucet, A.; Teh, Y.W. Augmented neural odes. *Advances in neural information processing systems* **2019**, *32*.
8. Coelho, C.; Costa, M.F.P.; Ferrás, L.L. Tracing footprints: Neural networks meet non-integer order differential equations for modelling systems with memory. The Second Tiny Papers Track at ICLR 2024, 2024.
9. Coelho, C.; Costa, M.F.P.; Ferrás, L. Neural Fractional Differential Equations. *arXiv preprint arXiv:2403.02737* **2024**.
10. Caputo, M. Linear Models of Dissipation whose Q is almost Frequency Independent–II. *Geophysical Journal International* **1967**, *13*, 529–539. doi:10.1111/j.1365-246x.1967.tb02303.x.

 doi:10.20944/preprints202407.2572.v1

11. Diethelm, K., The Analysis of Fractional Differential Equations: An Application-Oriented Exposition Using Differential Operators of Caputo Type; Springer Berlin Heidelberg: Berlin, Heidelberg, 2010. doi:10.1007/978-3-642-14574-2.
12. Diethelm, K.; Ford, N.J. Analysis of fractional differential equations. *Journal of Mathematical Analysis and Applications* **2002**, *265*, 229–248.
13. Diethelm, K. Smoothness properties of solutions of Caputo-type fractional differential equations. *Fractional Calculus and Applied Analysis* **2007**, *10*, 151–160.
14. Zhang, H.; Gao, X.; Unterman, J.; Arodz, T. Approximation Capabilities of Neural ODEs and Invertible Residual Networks. Proceedings of the 37th International Conference on Machine Learning; III, H.D.; Singh, A., Eds. PMLR, 2020, Vol. 119, *Proceedings of Machine Learning Research*, pp. 11086–11095.
15. Augustine, M.T. A Survey on Universal Approximation Theorems. *arXiv preprint arXiv:2407.12895* **2024**.
16. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**.