TRACING FOOTPRINTS: NEURAL NETWORKS MEET NON-INTEGER ORDER DIFFERENTIAL EQUATIONS FOR MODELLING SYSTEMS WITH MEMORY

C. Coelho [1], M. Fernanda P. Costa [1], L.L. Ferrás [1,2]
[1]Centre of Mathematics (CMAT), University of Minho
[2]Department of Mechanical Engineering (Section of Mathematics), FEUP - University of Porto
cmartins@cmat.uminho.pt, mfc@math.uminho.pt, lferras@fe.up.pt

ABSTRACT

Neural Ordinary Differential Equations (Neural ODEs) have gained popularity for modelling real-world systems, thanks to their ability to fit ODEs to data. However, numerous systems in science and engineering often exhibit intricate memory behaviours, being classical ODEs inadequate for such tasks due to their inability to handle strong and complex memory effects. In this work, we introduce the Neural Fractional Differential Equation (Neural FDE), a Neural Network (NN) architecture to fit a FDE to data. With this we leverage the capabilities of FDEs allowing the architecture to take into account all past states and their influence on a system's current and future behaviours. Neural FDE inherently exhibits memory, providing a more accurate representation of complex phenomena in systems with long-term dependencies. Numerical experiments show Neural FDE generalises better and has faster convergence than Neural ODEs.

1 INTRODUCTION

Real-world systems in science and engineering exhibit intricate dynamics, nonlinear interactions, and emergent phenomena. Mathematical models, often formulated as continuous-time functions using DEs, are essential for understanding and predicting these systems. DEs describe the dynamic evolution of system variables over time, capturing interactions and external influences without the need for time-consuming or expensive experiments. Building mathematical models is challenging due to their complex and nonlinear relationships between variables. With the emergence of NNs, Chen et al. (2018) proposed Neural ODEs, a NN with a continuous-depth that adjusts a ODE to the dynamics of the data. Although ODEs effectively describe instantaneous rates of change, often systems exhibit intricate memory or a complex dependence on past states beyond the immediate previous state, making ODEs inadequate for modelling such dynamics. These complexities are observed in various contexts, such as the interactions of molecules in a cell (Amilo et al., 2023), turbulent flows (Ming et al., 2016), the fluctuations of financial markets (Ara et al., 2018) and the dynamics of populations (Rivero et al., 2011). In this work, we introduce Neural FDE, a NN that fits a FDE to the dynamics of data. FDEs extend the concept of derivatives to noninteger (or fractional) orders, having an inherent memory thus taking into account all past states of a system and its influence on the current and future behaviour (Herrmann, 2011). To the best of our knowledge, this is the first time a NN architecture is proposed to fully model a FDE to the dynamics of data, including the order of the fractional derivative.

2 METHOD

Neural FDE is an architecture composed of three components: an arbitrary NN $\boldsymbol{f_\theta}$ that builds the dynamics of the FDE, with parameters $\boldsymbol{\theta}$; an arbitrary NN $\boldsymbol{\alpha_\phi}$ that adjusts the fractional order of the derivative $\alpha \in (0, 1)$ by receiving as input the last value of α, with parameters $\boldsymbol{\phi}$; a numerical solver *FDESolve*$(\alpha, \boldsymbol{f_\theta}, \boldsymbol{h}_0, [t_0, t_f])$ that solves the FDE in times $[t_0, t_f]$ with initial state $\boldsymbol{h}_0$ given by the first state of the training data at $t = t_0$. For this purpose we implemented a Predictor-Corrector FDE

Table 1: Performance at modelling a population growth dynamics (MSE $\pm$ std).

	RECONSTRUCTION		EXTRAPOLATION		COMPLETION	
DATASET	Neural ODE	Neural FDE	Neural ODE	Neural FDE	Neural ODE	Neural FDE
$P_{\alpha=0.3}$	7.70E-03 $\pm$ 2.15E-05	1.58E-03 $\pm$ 7.34E-04	1.04E-02 $\pm$ 1.54E-05	**2.97E-03** $\pm$ **9.06E-04**	7.67E-03 $\pm$ 2.15E-05	1.96E-03 $\pm$ 7.33E-04
$P_{\alpha=0.4}$	3.90E-02 $\pm$ 2.45E-02	**4.03E-03** $\pm$ **7.78E-04**	4.17E-02 $\pm$ 1.91E-02	**7.36E-03** $\pm$ **1.25E-03**	3.90E-02 $\pm$ 2.45E-02	**5.01E-03** $\pm$ **6.51E-04**
$P_{\alpha=0.99}$	4.58E-02 $\pm$ 1.50E-03	1.32E-02 $\pm$ 2.05E-03	3.45E-02 $\pm$ 4.44E-04	**8.97E-03** $\pm$ **9.89E-04**	4.58E-02 $\pm$ 1.50E-03	1.45E-02 $\pm$ 1.91E-03
P_{ODE}	8.33E-02 $\pm$ 5.19E-02	1.15E-02 $\pm$ 1.49E-03	7.91E-02 $\pm$ 6.29E-02	**7.84E-03** $\pm$ **9.49E-04**	8.34E-02 $\pm$ 5.19E-02	1.30E-02 $\pm$ 1.37E-03

solver in *Pytorch* (see Diethelm et al. (2002) for more details). The goal of Neural FDE is to fit the curve of solutions of an initial value problem: ${}^{C}D^{\alpha}\boldsymbol{h}(t) = \boldsymbol{f_\theta}(\boldsymbol{h}(t), t)$ (with $\boldsymbol{h}(0) = \boldsymbol{h}_0$) where ${}^{C}D^{\alpha}$ denotes the Caputo fractional derivative of order α (see Appendix A) (Herrmann, 2011). To train the Neural FDE, the parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$ are jointly optimised, since both directly contribute to the predictions of the model, by minimising a loss function defined by the error between the predicted and ground-truth values, Algorithm 1. The result of training a Neural FDE is a FDE of order α. To make predictions, a numerical solver is used with initial condition $(\boldsymbol{h}_0, t_0)$ in time interval (t_0, t_f)

Algorithm 1 Neural FDE training process.

Input: initial condition $(\boldsymbol{h}_0, t_0)$, time interval $[t_0, t_f]$, maximum number of iterations *MAXITER*;
$\boldsymbol{f_\theta} \leftarrow$ *DynamicsNN*();
$\alpha_{\boldsymbol{\phi}} \leftarrow$ *AlphaNN()*;
Initialise $\boldsymbol{\theta}, \boldsymbol{\phi}, \alpha$;
for $k = 0:$ *MAXITER* **do**
  $\alpha \leftarrow \alpha_{\boldsymbol{\phi}}(\alpha)$;
  $\{\boldsymbol{h}_t\}_{t=1,\ldots,f} \leftarrow$ *FDESolve*$(\alpha, \boldsymbol{f_\theta}, \boldsymbol{h}_0, [t_0, t_f])$;
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ *Optimiser.Step*$(\nabla \mathcal{L})$;
end for
Return: $\boldsymbol{\theta}, \alpha$;

To evaluate the Neural FDE we developed four toy time-series datasets, that describe a synthetic population growth dynamics. Three were created by numerically solving a FDE with $\alpha = 0.3, 0.4, 0.99$ and one by solving an ODE, denoted by $P_{0.3}, P_{0.4}, P_{0.99}, P_{\text{ODE}}$ respectively. Each dataset has available data for three experiments: reconstruction; extrapolation; completion. For comparison we used the Neural ODE as a baseline (see Appendix B for details on the experimental setup). For each model and dataset three independent runs were performed and the average Mean Squared Error (MSE) and respective standard deviations (std) were computed. The numerical results in Table 1 show that Neural ODE and Neural FDE exhibit similar performance in modelling the training dataset (reconstruction) and inputting missing data within the training time interval (completion). Notably, Neural FDE outperforms Neural ODE, particularly with respect to $P_{\alpha=0.4}$. However, when predicting beyond the training time interval (extrapolation), Neural FDE is consistently better than Neural ODE demonstrating higher generalisation capabilities. This outcome aligns with expectations, as Neural FDE leverages the entirety of the historical information, thereby enhancing its modelling capabilities. Furthermore, by analysing the training process (see Appendix C), it becomes evident that Neural FDE exhibits significantly accelerated convergence, attributable to the intrinsic memory concept inherent in FDEs.

3 Conclusion

In this paper, we propose the Neural FDE, a novel architecture to adjust a continuous-depth NN to data, by using a FDE. While Neural ODEs model the system behaviour instantaneously by adjusting an ODE that relies solely on the immediate previous state, Neural FDEs take a different approach, and consider the entire history of the system, establishing relationships between all past states and the current or future states. This unique characteristic enables Neural FDE to effectively model systems with memory or long-term dependencies. Preliminary numerical results show that Neural FDE has greater generalisation capabilities being able to better extract the underlying dynamics of a system. This is due to the influence of all historical past to a current or future state of a system. The computational cost of the Neural FDE is inherently higher than that of the Neural ODE, due to the need to store and use the complete past data, resulting in the creation of a memory effect. Despite this, the Neural FDE exhibits faster convergence, reaching significantly lower training loss values than the Neural ODE at a notably quicker pace. Analysing and addressing the increase of computational cost is an important future direction.

ACKNOWLEDGEMENTS

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 for the funding through the scholarship with reference 2021.05201.BD.

This work is 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.

URM STATEMENT

The authors acknowledge that at least one key author of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

REFERENCES

David Amilo, Bilgen Kaymakamzade, and Evren Hincal. A fractional-order mathematical model for lung cancer incorporating integrated therapeutic approaches. *Scientific Reports*, 13(1):12426, August 2023. ISSN 2045-2322. doi: 10.1038/s41598-023-38814-2.

Asmat Ara, Najeeb Alam Khan, Oyoon Abdul Razzaq, Tooba Hameed, and Muhammad Asif Zahoor Raja. Wavelets optimization method for evaluation of fractional partial differential equations: An application to financial modelling. *Advances in Difference Equations*, 2018(1):8, January 2018. ISSN 1687-1847. doi: 10.1186/s13662-017-1461-2.

Ricky T. Q. Chen. torchdiffeq, 2018. URL `https://github.com/rtqichen/torchdiffeq`.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Kai Diethelm, Neville J Ford, and Alan D Freed. A predictor-corrector approach for the numerical solution of fractional differential equations. *Nonlinear Dynamics*, 29:3–22, 2002.

Richard Herrmann. *Fractional calculus: an introduction for physicists*. World Scientific, 2011.

Chunying Ming, Fawang Liu, Liancun Zheng, Ian Turner, and Vo Anh. Analytical solutions of multi-term time fractional differential equations and application to unsteady flows of generalized viscoelastic fluid. *Computers & Mathematics with Applications*, 72(9):2084–2097, November 2016. ISSN 0898-1221. doi: 10.1016/j.camwa.2016.08.012.

Margarita Rivero, Juan J. Trujillo, Luis Vázquez, and M. Pilar Velasco. Fractional dynamics of populations. *Applied Mathematics and Computation*, 218(3):1089–1095, October 2011. ISSN 0096-3003. doi: 10.1016/j.amc.2011.03.017.

A CAPUTO FRACTIONAL DERIVATIVE DEFINITION

Fractional derivatives are mathematical operators that extend the concept of traditional derivatives to noninteger orders. Various definitions exist for fractional derivatives, which differ in their formulations. The choice of a particular definition is often dictated by the specific problem at hand. In this work we used Caputo's definition (Herrmann, 2011):

$$ {}^{C}D^{\alpha}\boldsymbol{h}(t) = \frac{1}{\Gamma(\lceil\alpha\rceil - \alpha)} \int_0^t (t-s)^{\lceil\alpha\rceil-\alpha} h^{\lceil\alpha\rceil}(s) ds $$

In our work, we consider $\alpha \in (0, 1)$ since this is the case in many applications Diethelm et al. (2002).

B EXPERIMENTAL SETUP

Population Growth Dynamics Datasets To create the toy datasets, the FDE (1) that models the dynamics of population growth was numerically solved with fractional orders $\alpha = 0.3, 0.4, 0.99$.

$$ {}^{C}D^{\alpha}P(t) = rP(t)\left(1 - \frac{P(t)}{K}\right), \; P(0) = 100 \tag{1}$$

where $r = 0.1$ is the rate of growth, $K = 1000$ is the carrying capacity of the environment and $P(0) = 100$ is the initial condition.

To evaluate the performance of the Neural FDE at modelling data generated using an ODE we also created a fourth toy dataset described by the ODE (2).

$$ \frac{dP(t)}{dt} = rP(t)\left(1 - \frac{P(t)}{K}\right), \; P(0) = 100 \tag{2}$$

For each dataset, three distinct experiments were conducted:

- *Reconstruction:* to evaluate the performance of the models at learning the training data by using the same set of data points for training and testing. The set has 200 regularly sampled points in time interval $(0, 300)$;
- *Extrapolation:* to evaluate the performance of the models at predicting for unseen time horizons by using a larger time interval for testing. The training set has 200 regularly sampled points in $t = (0, 300)$ and the testing set has 200 regularly sampled points in $t = (0, 400)$;
- *Completion:* to evaluate the performance of the models at inputting missing data by using a higher time frequency for testing. The training set has 200 regularly sampled points in $t = (0, 300)$ and the testing set has 300 regularly sampled points in $t = (0, 300)$.

Implementation and Training Details For the experimental results, a NN configuration was employed to model the dynamics of the right-hand side of the ODE and FDE, $\boldsymbol{f_\theta}$. This configuration consisted of an input and an output layer with 1 neuron and hyperbolic tangent ($tanh$) activation function. Additionally, 3 hidden layers with 64 neurons each and $tanh$ activation functions were used. The optimisation process used Adam optimiser with a learning rate of $1e-3$, and a total of 200 iterations were performed.

As for the NN that adjusts the α value, $\boldsymbol{\alpha_\phi}$, the architecture featured an input layer with a $tanh$ activation function, an output layer with a sigmoid activation function, and one neuron in each layer. Two hidden layers were included with 32 neurons each and $tanh$ activation functions. The initialisation of the α value was set to 0.99.

For the optimisation of both neural networks, the Mean Squared Error (MSE) loss function was employed.

All implementations were done in *Pytorch*, and we closely followed the work by Diethelm et al. (2002) to implement the Predictor-Corrector fractional solver (a fractional ordinary differential equations solver is not available in *Python*). Furthermore, *Torchdiffeq* library was used for the Neural ODE Chen (2018).

C AUXILIARY PLOTS

As a preliminary analysis, the training loss evolution for the four datasets was plotted, Figure 1. This allows the study of the rate of convergence of the Neural ODE and Neural FDE architectures. Out of the three runs executed, plots for the second run, chosen arbitrarily, are here presented. The plots show that the Neural FDE is capable of achieving lower loss values in fewer iterations than the Neural ODE. Also, the Neural FDE achieves lower loss values.

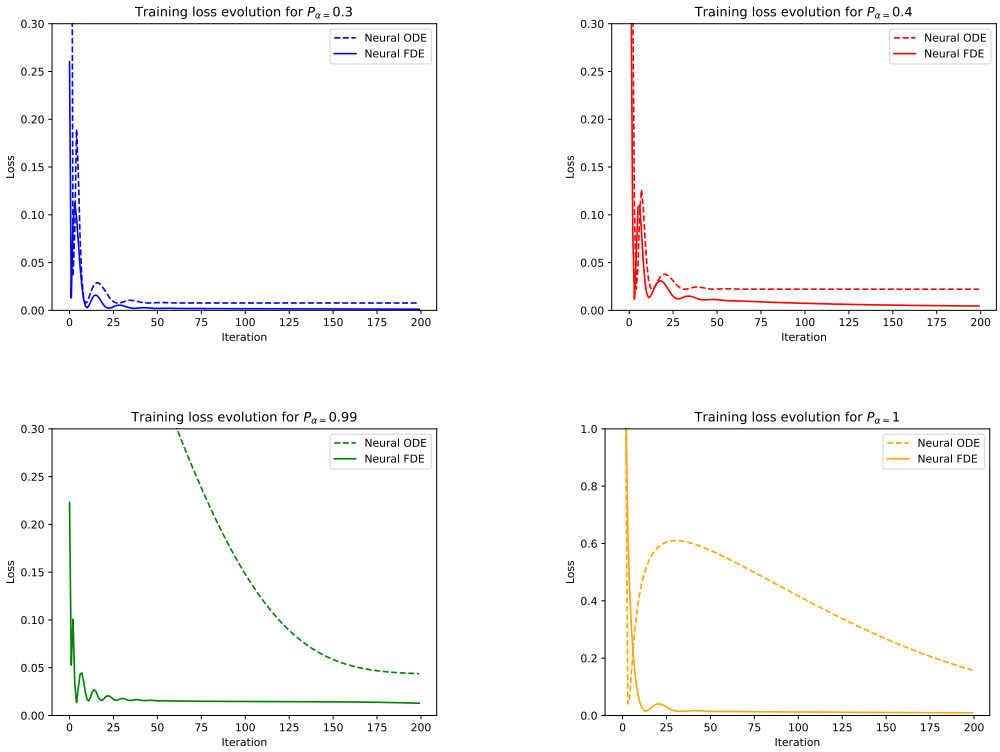

Figure 1: Evolution of the loss during training for the four datasets.