Neural Integro-Differential Equations

Emanuele Zappala[*,1], Antonio H. de O. Fonseca[*,1], Andrew H. Moberly[1], Michael J. Higley[1], Chadi Abdallah[2], Jessica A. Cardin[1], David van Dijk[1]

[1]Yale University, New Haven, CT, USA
[2] Baylor College of Medicine, Houston, TX, USA
emanuele.zappala@yale.edu, antonio.fonseca@yale.edu, andrew.moberly@yale.edu, m.higley@yale.edu, chadi.abdallah@bcm.edu, jess.cardin@yale.edu, david.vandijk@yale.edu

Abstract

Modeling continuous dynamical systems from discretely sampled observations is a fundamental problem in data science. Often, such dynamics are the result of non-local processes that present an integral over time. As such, these systems are modeled with Integro-Differential Equations (IDEs); generalizations of differential equations that comprise both an integral and a differential component. For example, brain dynamics are not accurately modeled by differential equations since their behavior is non-Markovian, i.e. dynamics are in part dictated by history. Here, we introduce the Neural IDE (NIDE), a novel deep learning framework based on the theory of IDEs where integral operators are learned using neural networks. We test NIDE on several toy and brain activity datasets and demonstrate that NIDE outperforms other models. These tasks include time extrapolation as well as predicting dynamics from unseen initial conditions, which we test on whole-cortex activity recordings in freely behaving mice. Further, we show that NIDE can decompose dynamics into their Markovian and non-Markovian constituents via the learned integral operator, which we test on fMRI brain activity recordings of people on ketamine. Finally, the integrand of the integral operator provides a latent space that gives insight into the underlying dynamics, which we demonstrate on wide-field brain imaging recordings. Altogether, NIDE is a novel approach that enables modeling of complex non-local dynamics with neural networks.

Introduction

Integro-differential equations (IDEs) are a class of functional differential equations that are non-local in time. IDEs naturally describe many dynamical systems, including population dynamics, nuclear reactor physics, and visco-elastic fluids, as considered in detail in Lakshmikantham (1995). Further examples are models of brain dynamics (Martin et al. 2018; Wilson and Cowan 1972; Amari 1977), as well as infectious disease spreading (Medlock and Kot 2003).

Functional differential equations, and IDEs in particular, are equations determined by non-local operators. These are mappings, or *functionals*, from a space of functions into itself that require knowledge of the input function at non-infinitesimal neighbourhoods in order to be computed. Integral operators epitomize such functionals, and the corresponding theory of IDEs has become central, for instance, in kinetic theory (Grigoriev et al. 2010). Examples of such applications are the Boltzmann kinetic equation, the Vlasov equation, and the Landau kinetic equation. In contrast, differential operators such as time derivatives are local operators in that the notion of derivative at one point requires information from the "immediate vicinity" of the point. This is formalized through the concept of limit.

The theory of IDEs stems from the necessity of modeling systems that present spatio-temporal relations which transcend local modeling, and works of Volterra on population dynamics have motivated this since as early as the beginning of the 1900's (Volterra 1913, 1932). Consequently, IDEs present several properties that are unique to their purely non-local behavior (Grigoriev et al. 2010; Trenogin 2020). Despite their importance, no approach for learning IDE systems from data exists. Motivated by this, we develop a deep learning method called *Neural Integro-Differential Equation* (NIDE), which learns an IDE whose solution approximates data sampled from given non-local dynamics. A schematic representation of the functioning of NIDE is given in Figure 1A. To the best of our knowledge, this is the first deep learning framework for modeling of non-local continuous dynamics.

Contributions

The main contributions of this article are as follows:

- We implement an IDE solver fully supported by PyTorch.
- We introduce a deep learning method for modeling IDEs, namely *Neural IDE*, which learns an integral operator from a space of functions into itself.
- We derive the adjoint state of NIDEs and use this for backpropagation during training.
- We present a method for decomposing dynamics into local and non-local components.
- Finally, we use our method to model brain dynamics, including wide-field calcium imaging in mice and fMRI recordings of people on ketamine.

[*]Co-first authors

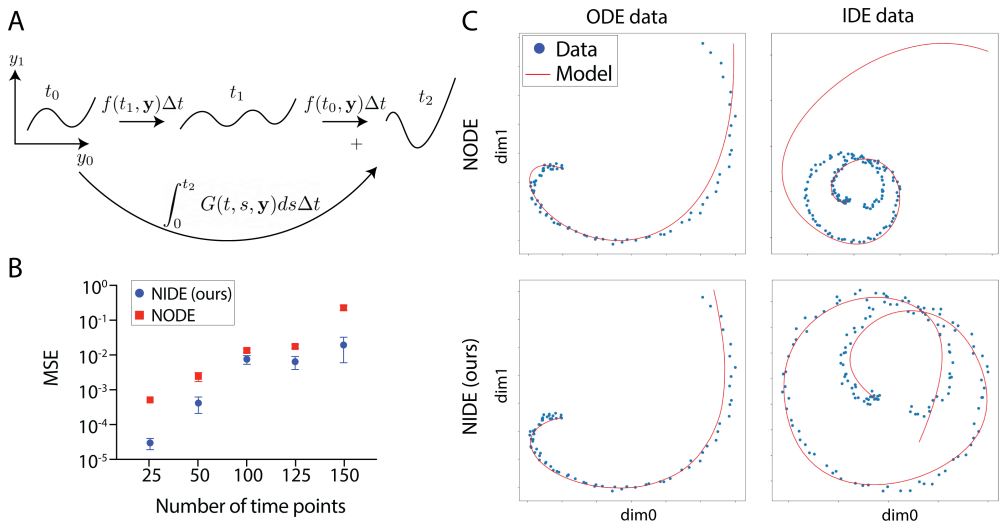

Figure 1: Panel A shows a schematic of the NIDE method, where time steps are determined through a superposition of local and non-local components. Panel B shows performance (MSE of model fits) of NIDE and NODE on data sampled at increasingly longer time segments from an IDE system. NIDE significantly ($P < 10^{-6}$) outperforms NODE, especially at longer time segments (see Table 1). Panel C shows 2D spirals generated with either an ODE (left) or an IDE (right) and fitted by NODE (top) or NIDE (bottom). While both models can fit the ODE spiral, only NIDE can accurately fit the IDE spiral.

Related Work

The use of differential equation solvers to learn dynamics through neural networks has been pioneered in Chen et al. (2018); Chen, Amos, and Nickel (2021) to address the need of having continuous deep learning models. These models, called *Neural Ordinary Differential Equations*, will be referred to as NODEs in the rest of the article. Our theory introduces a deep learning approach to the case of functional differential equations, where dynamics present global (i.e. non-local) behavior. An introduction to functional differential equations can be found in (Stech et al. 1981). A viable approach to incorporating non-local dependence in ODEs would be, for example, the use of LSTMs in NODEs. However, this is essentially equivalent to the use of delay differential equations (DDEs (Smith 2011)).

In various applications that span from computational biology to physics and engineering, however, both ODEs and DDEs are not capable of modeling dynamical systems with non-local properties, where the use of functional differential equations, such as IDEs, is employed instead (Volterra 1913, 1932; Grigoriev et al. 2010; Wazwaz 2011; Stech et al. 1981). In our setting, we learn an integral operator on a space of functions. Learning operators on function spaces is a deep learning problem that has been considered for instance in Lu et al. (2021a) and Goswami et al. (2022), where the case of integral operators is considered in detail as well. The main difference between our method and the latter is that the use of an IDE solver, as we will show, allows us to learn in a continuous manner, as well as produce continuous dynamics.

Operator learning is a widely developed approach that encompasses numerous machine learning techniques, and is applied in several disciplines. We emphasize that the usual setting of operator learning is over a fixed grid that approximates the domain of the functions, and that the infinite dimensional space of functions is usually projected (e.g. Galerkin method). Moreover, recovering the continuous limit (i.e. the grid steps going to zero) is a fundamental challenge. A general perspective is given in Kovachki et al. (2021), for instance, where the authors consider several problems arising from Partial Differential Equations (PDEs). In the introduction of Kovachki et al. (2021), a relatively comprehensive account of the different methodologies is also provided.

Solving IDEs by means of deep learning approaches has been considered in Fu and Hirsa (2021), and several methodologies are overviewed in Lu et al. (2021b). We point out, however, that the perspective of the present article is somehow inverted with respect to the previous works on IDEs appearing in the context of machine learning. In fact, to our knowledge, previous works have only focused on obtaining methods for solving IDEs, in supervised or unsupervised settings, that are given as input of the problem. Instead, the approach that we take learns an IDE whose solution approximates data sampled from a dynamical system. What is learned here is not a solution to a known system, but the system itself. Consequently, the input of our method is not an analytical IDE, but data without any a priori knowledge of the system that has generated it. The main novelty of the present work resides in the fact that no prior analytical knowledge of the system is required and, therefore, NIDE provides insight into dynamical systems whose underlying nature is not necessarily well understood.

Learning Dynamics with Integro-Differential Equations

We consider the problem of learning dynamics that depend nontrivially on non-local effects, as well as on instantaneous information. Such a dynamical system is expressed in terms of both the derivative of a function, and a temporal integral. The resulting equation of this type is an integro-differential equation, and its general form (see Lakshmikantham (1995); Wazwaz (2011)) is given as

$$\frac{d\mathbf{y}}{dt} = f(t, \mathbf{y}) + \int_{\alpha(t)}^{\beta(t)} G(t, s, \mathbf{y}) ds,$$

where $\mathbf{y} : \mathbb{R} \longrightarrow \mathbb{R}^n$ is a vector function of the independent time variable t. The functions $f : \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^n$ and $G : \mathbb{R}^{n+2} \longrightarrow \mathbb{R}^n$ generate the dynamics. Observe that G depends explicitly on t, so that differentiating the integral with respect to t produces, in general, a new expression where an integral appears, using the Leibniz integral rule. The derivative is parameterized by a neural network f along with a temporal integral of another neural network G, the main difference lying in the fact that past and future states explicitly appear in the present state. We call such a model *Neural Integro-Differential Equation*, or NIDE for short. The theory of IDEs is better understood in the setting where the function G is a product of type $K(t, s)F(\mathbf{y}(s))$ for some arbitrary function $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, and a matrix valued function $K : \mathbb{R}^2 \longrightarrow \mathrm{M}(\mathbb{R}, n)$, where $\mathrm{M}(\mathbb{R}, n)$ indicates the space of square matrices with real coefficients. The function K is called *kernel*. In this article, we therefore consider systems of type

$$\frac{d\mathbf{y}}{dt} = f(t, \mathbf{y}) + \int_{\alpha(t)}^{\beta(t)} K(t, s)F(\mathbf{y}) ds, \tag{1}$$

where K and F are both neural networks that will be learned during training, and K will explicitly depend on the two time variables t and s. In fact, we can generalize this setup and assume that the kernel function takes values in the space of rectangular matrices $\mathrm{M}(\mathbb{R}, m, n)$ where m is the dimension of a latent space, and therefore $F : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ maps $\mathbf{y}$ in the latent space. This allows us to learn dynamics in the latent space. The kernel then maps the latent space back to the original data space. We observe that the variables t and s that K depends on can be intuitively interpreted as "global" and "local" times, respectively. For each time t, the value of the solution $\mathbf{y}(t)$ is determined via an integration in ds, so that for each value of t we have a different integration space $[\alpha(t), \beta(t)]$ where s lies.

We implement an IDE solver based on the work by Gelmi and Jorquera (2014) and Karpel (2018). We use this solver to obtain dynamics, i.e. a solution $\mathbf{y}(t)$ of an IDE as above, for given neural networks K and F. To minimize the loss function L with respect to target data, we use the adjoint state of the IDE in Equation 1, cf. Chen et al. (2018); Stapor, Fröhlich, and Hasenauer (2018), or backpropagate directly through the IDE solver. The adjoint function is defined as $\mathbf{a}(t) := \frac{\partial L}{\partial \mathbf{y}}$ and its dynamics is determined by another system of IDEs, since it is obtained by differentiating the loss function evaluated on the output of the IDE solver. Before considering the dynamical system that determines the evolution of $\mathbf{a}(t)$, it is important to consider the method for solving the IDEs that is employed in this article, as this plays a fundamental role in the implementation and provides a substantial difference with respect to the case of NODEs.

Notice that the functions $\alpha(t)$ and $\beta(t)$ are arbitrary and, therefore, for each $t \in [t_0, t_1]$, the derivative of the function $\mathbf{y}(t)$, $\frac{d\mathbf{y}}{dt}$, depends on values of $\mathbf{y}(t)$ both in the past, the present, and the future of the dynamics. Common choices of α and β are $\alpha(t) = 0$ and $\beta(t) = t$ (Volterra IDEs) or $\alpha(t) = a$ and $\beta(t) = b$ (Fredholm IDEs), see Zappala et al. (2022) Appendix A. Effectively, this means that the RHS of Equation 1 is a functional of $\mathbf{y}$ rather than a function. Therefore, an IDE solver cannot sequentially output the value of $\mathbf{y}(t)$ based on the computation of $\mathbf{y}$ on the previous time point. The idea is therefore to iteratively solve the IDE by producing successive approximations of the solution $\mathbf{y}(t)$ from an initial guess $\mathbf{y}_0(t)$ until convergence of $y_n(t)$ to the solution, within some predetermined tolerance. This allows us to have a full function $\mathbf{y}_n(t)$ at each iteration, and this can then be integrated over arbitrary intervals $[\alpha(t), \beta(t)]$ during the dynamics.

Following the discussion in the previous paragraph, $\mathbf{a}(t) = \frac{\partial L}{\partial \mathbf{y}}$ is the functional derivative of the loss L with respect to $\mathbf{y}(t)$. The loss function L is computed by applying a chosen method (e.g. mean squared error) to the output of the IDE solver. One has the formula

$$\begin{aligned} \frac{d\mathbf{a}(t)}{dt} &= -\int_{t_1}^{t_0} \mathbf{a}(t)^T \frac{\partial f}{\partial \mathbf{y}} dt \\ &- \int_{t_1}^{t_0} \mathbf{a}(t)^T K(t, t) F(\mathbf{y}(t)) dt, \end{aligned} \tag{2}$$

which is derived in the next section. In order to derive the gradients with respect to the parameters θ of the neural networks K and F that define the NIDE we have another equation, namely

$$\begin{aligned} \frac{dL}{d\theta} &= -\int_{t_1}^{t_0} \int_{\alpha(t)}^{\beta(t)} a(t)^T \frac{\partial K(t, s)}{\partial \theta} F(\mathbf{y}(s)) ds dt \\ &- \int_{t_1}^{t_0} \int_{\alpha(t)}^{\beta(t)} a(t)^T K(t, s) \frac{\partial F(s)}{\partial \theta} ds dt. \end{aligned} \tag{3}$$

As discussed in Appendix B Zappala et al. (2022), the adjoint state can be both solved as an ODE where the RHS is obtained via integration of gradients, or it can be solved as an IDE through an iterative procedure.

Adjoint Dynamics

We now consider the dynamics of Equation 1 and derive the corresponding adjoint state, cf. Chen et al. (2018) and Stapor, Fröhlich, and Hasenauer (2018). In particular, we want to show that the adjoint function $\mathbf{a}(t) := \frac{\partial L}{\partial \mathbf{y}}$, where L is the loss function used for training, when f is trivial,

satisfies the IDE

$$\frac{d\mathbf{a}_{aug}}{dt} = [-\mathbf{a}^T \cdot \frac{\partial}{\partial \mathbf{y}}(K(t,t)F(\mathbf{y}(t)))| \tag{4}$$

$$-\mathbf{a}_\theta^T \cdot \int_{\alpha(t)}^{\beta(t)} \frac{\partial}{\partial \theta}(K(t,s)F(\mathbf{y}(s)))ds], \tag{5}$$

where $\mathbf{a}_{aug}$ is the adjoint state of the augmented dynamics, which includes the gradients with respect to the parameters θ of the neural networks K and F, and the square brackets refer to the fact that the dimensions of $\mathbf{a}_{aug}$ are split in two groups of direct summands, which we separate by a vertical bar for clarity. The case when f is nontrivial is obtained from Equation 4 by appropriately adding the computation for the adjoint found in (Chen et al. 2018). This accounts to adding $-\mathbf{a}^T \cdot \frac{\partial f(\mathbf{y},t)}{\partial \mathbf{y}}$ to the first term in the RHS of Equation 4, and concatenating $-\mathbf{a}_\theta^t \cdot f(\mathbf{y},t)$ to the second term, where θ here refers to the parameters of the neural network f. For simplicity we explicitly consider the case with trivial f, since the general case is a combination of this and the results of Chen et al. (2018).

Before showing that Equation 4 describes the dynamics of the adjoint state, we recall the notion of functional derivative, and some known results that can be found in standard quantum field theory textbooks, e.g. Srednicki (2007). An application of such techniques in Bayesian inference has also recently appeared in Kim (2021).

A functional S is a mapping from a space of functions, say $\mathcal{F}$, to the real (or complex) numbers $S : \mathcal{F} \longrightarrow \mathbb{R}$. The notion of variation of a functional is a well known concept from the theory of calculus of variations, and finds its roots in the classical formulation of mechanics, where the equations of motion are found by minimizing the action, which is a functional, whose variation is set to zero. Extending the notion of variation to that of derivative for a functional is not straightforward, and it is formally defined as a limit of test functions (i.e. a distribution). Let us now consider the case of scalar argument functions in $\mathcal{F}$, since the extension to vector functions follows from this. For a functional of type $S(y(x)) = \int G(y(t), d_t y(t), \dots, d_t^k y(t))dt$, where d_t^r is a shorthand for $\frac{d^r}{dt^r}$, the functional derivative is

$$\begin{aligned} &\frac{\delta S(y(t))}{\delta y(t)} \\ &= \int [\frac{\partial G}{\partial y(\tau)} - d_\tau \frac{\partial G}{\partial (d_\tau y(\tau))} + \cdots] \frac{\delta y(\tau)}{\delta y(t)} d\tau \end{aligned} \tag{6}$$

The distribution $\frac{\delta y(\tau)}{\delta y(t)}$ is the Dirac's Delta function $\delta(t-\tau)$, so that one finds

$$\frac{\delta S(y(t))}{\delta y(t)} = \frac{\partial G}{\partial y(t)} - d_\tau \frac{\partial G}{\partial (d_t y(\tau))} + \cdots \tag{7}$$

Observe that, in our case, the integral part of the NIDE is an example of the functional S given above, parametrized by global and local times, and without higher order derivatives with respect to $\mathbf{y}(t)$. Moreover, the function G appearing in S is the composition of F and kernel K, so the functional derivative of the integral part of the NIDE in Equation 1 becomes

$$\frac{\delta}{\delta(\mathbf{y})}[\int_{\alpha(t)}^{\beta(t)} K(t,s)F(\mathbf{y}(s))ds] = \frac{\partial(K(t,t)F(\mathbf{y}(t)))}{\partial y(t)} \tag{8}$$

We now consider the adjoint dynamics. We proceed in a fashion similar to Section B.1 in Chen et al. (2018). Recall, from above, that $\mathbf{a}(t) := \frac{\partial L}{\partial \mathbf{y}}$, where L is the loss function evaluated on the output of the IDE solver. Then, we consider the equality $\frac{dL}{d\mathbf{y}(t)} = \frac{dL}{d\mathbf{y}(t+\epsilon)} \frac{d\mathbf{y}(t+\epsilon)}{d\mathbf{y}(t)}$ (cf. with previous discussion on functional derivatives), which corresponds to $\mathbf{a}(t) = \mathbf{a}(t+\epsilon)^T \cdot \frac{d\mathbf{y}(t+\epsilon)}{d\mathbf{y}(t)}$, using the definition of $\mathbf{a}(t)$. To derive $\frac{d\mathbf{y}(t+\epsilon)}{d\mathbf{y}(t)}$, let us first write

$$\mathbf{y}(t+\epsilon) = \mathbf{y}(t) + \int_t^{t+\epsilon} \int_{\alpha(\tau)}^{\beta(\tau)} K(\tau,s)F(\mathbf{y}(s))ds d\tau, \tag{9}$$

which is obtained by integrating the IDE from t to $t+\epsilon$. For small values of ϵ, Equation 9 becomes

$$\mathbf{y}(t+\epsilon) = \mathbf{y}(t) + \epsilon \cdot \int_{\alpha(t)}^{\beta(t)} K(t,s)F(\mathbf{y}(s))ds, \tag{10}$$

from which we obtain

$$\begin{aligned} \frac{d\mathbf{y}(t+\epsilon)}{d\mathbf{y}(t)} &= \frac{\delta}{\delta \mathbf{y}(t)}[\mathbf{y}(t) + \epsilon \cdot \int_{\alpha(t)}^{\beta(t)} K(t,s)F(\mathbf{y}(s))ds] \\ &= 1 + \epsilon \cdot \frac{\partial(K(t,t)F(\mathbf{y}(t)))}{\partial \mathbf{y}(t)}, \end{aligned} \tag{11}$$

where in the last equality we have used Equation 8. As a consequence, for small ϵ, we can write the equality

$$a(t) = a(t+\epsilon) + \epsilon a(t+\epsilon)^T \cdot \frac{\partial(K(t,t)F(\mathbf{y}(t)))}{\partial \mathbf{y}(t)}, \tag{12}$$

up to terms that vanish to zero with the same order of ϵ^2.

To complete, we now compute the derivative of $\mathbf{a}(t)$ w.r.t. time. We have

$$\frac{d\mathbf{a}}{dt} = \lim_{\epsilon \to 0^+} \frac{\mathbf{a}(t+\epsilon) - a(t)}{\epsilon} \tag{13}$$

$$= \lim_{\epsilon \to 0^+} -\frac{1}{\epsilon}\mathbf{a}(t+\epsilon)^T [\epsilon \frac{\partial(K(t,t)F(\mathbf{y}(t)))}{\partial y(t)}] \tag{14}$$

$$= -a(t)^T \cdot \frac{\partial(K(t,t)F(\mathbf{y}(t)))}{\partial y(t)}. \tag{15}$$

In order to update the parameters of the neural networks of the NIDE, we need to consider the augmented system (cf. Chen et al. (2018) and Stapor, Fröhlich, and Hasenauer (2018)). The augmented state $\mathbf{a}_{\text{aug}}(t)$ is obtained by considering the augmented IDE, where $\mathbf{y}_{\text{aug}} = [\mathbf{y}(t)|\theta]$ is obtained by concatenating the parameters of F and K to $\mathbf{y}(t)$. The temporal derivative θ is trivial, since the parameters are time independent, and the augmented IDE reads

$$\frac{d\mathbf{y}_{\text{aug}}}{dt} = [\int_{\alpha(t)}^{\beta(t)} K(t,s)F(\mathbf{y}(s))ds|\mathbf{0}]. \tag{16}$$

When considering the augmented adjoint function $\mathbf{a}_{\text{aug}}(t)$, the time derivative of the dimensions of $\mathbf{a}_{\text{aug}}(t)$ corresponding to $\int_{\alpha(t)}^{\beta(t)} K(t,s)F(\mathbf{y}(s))ds$ are obtained as above. Then, given $\mathbf{a}_\theta(t) := \frac{\partial L}{\partial \theta}$ we need to obtain the dynamics associated to $\mathbf{a}_\theta(t)$. We can proceed as in the case of Equation 15 with the difference that the term involving $\frac{\delta}{\delta\theta}$ in this case simply gives

$$\begin{aligned} &\frac{\delta}{\delta\theta}\int_{\alpha(t)}^{\beta(t)} K(t,s)F(\mathbf{y}(s))ds \\ &= \int_{\alpha(t)}^{\beta(t)} \frac{\partial}{\partial\theta}(K(t,s)F(\mathbf{y}(s)))ds. \end{aligned} \tag{17}$$

The latter then can be used to compute the time derivative $\mathbf{a}_\theta(t)$ completing the derivation of Equation 4.

Experiments

In this section we perform several experiments with NIDEs on data that was analytically generated by dynamical systems corresponding to IDEs, and data of real-world systems that present non-local characteristics. Specifically, we consider brain dynamics data from whole-cortex activity recordings in freely behaving mice, and fMRI brain activity recordings of people on ketamine. We focus on comparing our model with NODEs, as this is another model that is continuous in time, and with LSTMs, which are discrete-time deep learning methods that model non-locality by including previous states of the data.

In addition, we investigate the interpretability of NIDEs in two ways: 1) By decomposing the dynamics into instantaneous and non-instantaneous components, and 2) by inspecting the latent space provided by the integrand function of the integral operator F.

We analyze, in detail, the capability of NIDE to extrapolate with respect to time, and to generalize to unseen initial conditions. Furthermore, in order to directly compare the expressivity of NIDE to NODE, we consider fitting models to data sampled from increasingly complex dynamics. We show that when the dynamics are generated by an IDE with sufficiently complex non-local properties, NODE is not capable of properly fitting the data, but NIDE is. Details about the architecture of the model used in each task are provided in Table 2 Zappala et al. (2022).

Testing the Expressivity of NIDE

We first compare the expressivity of NIDE to NODE by fitting data of dynamics generated by IDE and ODE systems. Here, we use the same number of parameters for both NIDE and NODE, with NIDE having its parameters split between function F and kernel K. While both NIDE and NODE perform well on the ODE generated curve, only NIDE can accurately fit the IDE generated data (Figure 1C). These experiments suggest that, with the same number of parameters, NIDE is a more expressive model than NODE, capable of fitting complex non-local dynamics.

To further explore this, we perform a more systematic experiment in which we gradually increase the data complexity, while fitting both NIDE and NODE models. Specifically, we sample points from a self-intersecting spiral that was generated using an IDE and fit models while increasing the range from which we sample the dynamics. We observe that shorter, and thus simpler dynamics, can easily be fitted by both models while longer, and thus more complex dynamics, can only be properly fitted by NIDE (Figure 1B, Table 1).

Time Extrapolation

To assess the ability of NIDE to generalize to future time points (i.e. extrapolate), we generate 1000 $4D$ dynamics over a fixed interval using an IDE with uniformly sampled initial conditions. During training, we randomly mask up to 50% of the time points at the end of the dynamics, which we predict during test. The experiments show that NIDE has lower MSE (Table 3 of Zappala et al. (2022)) and higher R^2 for the masked points, thereby demonstrating that NIDE extrapolates better than NODE on data sampled from an IDE system. An example of such extrapolation is shown in Figure 2.

Next, to inspect the extent to which NIDE can generalize to new initial conditions, we consider the model trained on the $4D$ curves dataset and evaluate it on curves from new initial conditions that have not been seen during training. We find that NIDE yields lower MSE for predicted dynamics from initial conditions than NODE, as shown in Table 5 in Zappala et al. (2022) per extrapolated time point.

Decomposition into Markovian and Non-Markovian Dynamics

NIDEs learn dynamics through the derivative of the function $\mathbf{y}(t)$ considered above, as the superposition of two components. The first one is common to the setting of NODEs, where $\frac{d\mathbf{y}}{dt}$ is given by a neural network f, evaluated on pairs $(t, \mathbf{y}(t))$: $f(t, \mathbf{y})$. We refer to this summand as the *Markovian* component, as the contribution of f at the instant t depends only on the current time step of the solver. On the contrary, the integral part of the NIDE will be referred to as the *non-Markovian* component, as at each time t this depends on the full interval $[\alpha(t), \beta(t)]$.

To test this construction, we generate $2D$ curves through an IDE with nontrivial analytical functions f, K, F, with different initial conditions. Since f, K and F are known, we can recover the decomposition into Markovian and non-Markovian for each of the curves in the dataset. This gives us a ground truth.

We train a NIDE model on the full data without providing any information regarding the underlying decomposition. Then, we compute the Markovian and non-Markovian decomposition of the trained models, and compare these to the ground truth. We find that NIDE can accurately reconstruct the ground truth components for both training and validation data. We find $R^2 = 0.991 \pm 0.01$ (mean$\pm$std) for the whole fitting, $R^2 = 0.957 \pm 0.03$ for the instantaneous part, and $R^2 = 0.943 \pm 0.05$ for the integral part. An example of a reconstructed decomposition is shown in Figure 4 of Zappala et al. (2022).

	Number of time points				
	25	50	100	125	150
NODE	5.0E-04± 1.0E-04	2.4E-03± 7.0E-04	1.35E-01±1.6E-03	1.7E-02±2.1E-03	2.34E-01±4.2E-02
NIDE	**2.82E-05± 1.01E-05**	**4.02E-04± 2.00E-04**	**7.5E-03± 2.10E-03**	**6.4E-03±2.60E-03**	**1.94E-02±1.34E-02**

Table 1: Average MSE for NIDE (ours) and NODE on fitting data sampled from an IDE generated self-intersecting spiral (lower is better).

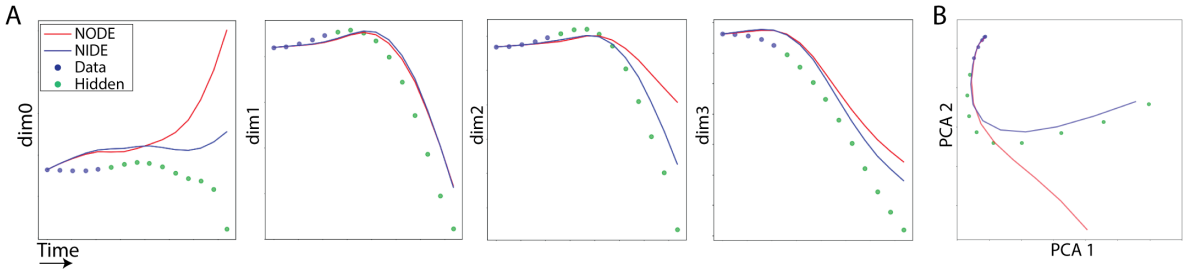

Figure 2: Comparison between NODE (red) and NIDE (ours, blue) in a time extrapolation task. Shown is one of the $4D$ curves. Panel A shows each of the 4 dimensions as a function of time, while panel B shows the 2D PCA projection of the dynamics. Data points that were masked during training and were then extrapolated are indicated as "hidden" (green). NIDE outperforms NODE in a time extrapolation task where, during training, up to 50% of the points are masked and then predicted during inference.

We stress that this is indeed a highly nontrivial result, as there is no a priori reason to assume that a given dynamics needs to be learned following a prescribed superposition of Markovian and non-Markovian components. However, if NIDE is exposed to a sufficiently large sample of dynamics generated by a given system, it is able to accurately reconstruct the local and non-local components.

We note that, for this experiment, we did not compare NIDE to other models since, to the best of our knowledge, no prior attempts have been made to decompose dynamics into their Markovian and non-Markovian components. For example, a NODE would only learn a Markovian component.

Modeling Brain Dynamics Using NIDEs

Spatiotemporal neural activity patterns can be modeled as a dynamical system (Vyas et al. 2020), for example using ODEs (Linden et al. 2021). However, ODEs have shown only partial success in modeling brain dynamics, arguably due to the significant non-local component of neuronal activity (Linden et al. 2021; Breakspear 2017). Previous attempts at considering the non-local behavior of brain dynamics include the Amari model (Amari 1977). The latter was a theoretical model based on IDEs, and NIDE provides a practical and concrete realization of it. More specifically, we apply NIDE to wide-field calcium brain imaging in freely behaving mice as well as fMRI recordings in humans on ketamine.

Wide-field calcium imaging is a recently developed technology that uses calcium indicators as a proxy for neural activity, which allows recording of brain dynamics at high spatial and temporal resolution in mice (Cardin, Crair, and Higley 2020). Details about data collection and preprocessing are found in Zappala et al. (2022).

We compare NIDE to NODE in a time extrapolation task of calcium imaging recordings in mice passively exposed to a visual stimulus (Lohani et al. 2020). Extrapolation performance is shown as the mean R^2 between prediction and ground truth for several recording segments (Table 4 of Zappala et al. (2022)). We find that NIDE outperforms NODE in this task. An example of an extrapolation is provided in Figure 3. Qualitatively, we observe that NIDE learns a better model because it predicts excitation in the visual cortex, which is present in the original data, while NODE predicts an inhibitory pattern.

We also test the performance of NIDE on predicting dynamics of new initial conditions (see Figure 5 in Zappala et al. (2022)). In this task, the whole sequence is predicted from a given initial condition. Here, too, NIDE outperforms NODE.

The function F (the integrand of the integral operator) produces a latent space embedding of the data. To test the resulting latent representation we compare the output of F to PCA and UMAP (McInnes, Healy, and Melville 2018) on the calcium imaging data. As shown in Zappala et al. (2022), Figure 7, the latent space of F presents a higher degree of geometric structure and resembles more accurately the topology of the embedded manifold with respect to time, which we quantify with k-NN regression using k=3 neighbors (Figure 7 of Zappala et al. (2022)).

Next, we apply our method of Markovian and non-Markovian decomposition to the calcium imaging data. This allows us to quantify the extent to which the brain dynamics are dictated by non-local components (Figure 8 in Zappala et al. (2022)). Interestingly, the decomposition shows

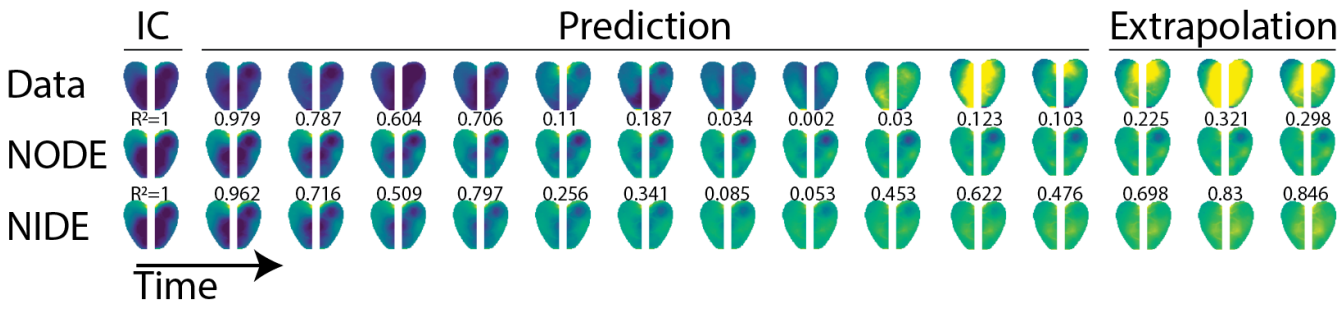

Figure 3: Time extrapolation performance of NODE and NIDE (ours) on wide-field calcium brain imaging data. NODE and NIDE were trained on brain dynamics in which time points (at the end of sequences) were masked. Time extrapolation was then assessed by predicting the masked (future) time points. NIDE (ours) outperforms NODE in this task. Shown is a dynamics with NODE and NIDE predictions and corresponding R^2 values with the ground truth. IC refers to the initial condition, which is provided to both models.

that the Markovian part of the signal mainly represents inhibition of neural activity in the visual cortex, while the non-Markovian part represents excitation, suggesting that the visual cortex is driven by non-local dynamics. Neural inhibition after the excitation is a known property of the excitatory-inhibitory networks in the brain, ensuring that any increase in neural activity will be followed by suppression (Bhatia, Moza, and Bhalla 2019). Its Markovian dynamics suggest that excitation is primarily determined locally in time, thus by the current state of the brain.

Functional magnetic resonance imaging (fMRI) is the most common technique for studying whole-brain activity in humans. In fMRI, the blood-oxygen-level dependent (BOLD) signal is used as a proxy for brain activity (Akiki and Abdallah 2019). Major depression is a worldwide leading cause of disability with poorly understood neurobiology and high treatment resistance. Ketamine, a serendipitously discovered rapid acting antidepressant, offers a unique opportunity to unravel the brain networks underlying major depression and to establish novel treatment targets (Abdallah et al. 2018). In this experiment, fMRI scans were acquired repeatedly during infusion of normal saline followed by intravenous infusion of a subanesthetic dose of ketamine in healthy subjects. The aim is to determine the ketamine induced brain dynamics during infusion, which are believed to reverse critical aspects of the psychopathology of major depression and lead to sustained relief of symptoms (Abdallah et al. 2017, 2021).

We find that the decomposition into Markovian and non-Markovian components, learned by NIDE, provides a space with better separation between ketamine and control conditions, as indicated by a higher k-NN classification accuracy with k=3 neighbors (Zappala et al. (2022) Figure 6, Table 6). This suggests that NIDE recovers dynamics relevant to the underlying biology. Interestingly, the separation between the two conditions is greater in the instantaneous part, compared to the integral part, suggesting that ketamine affects the Markovian component of brain dynamics, possibly by inducing a brain state that is more localized in time.

Scope and Limitations

The objective of this work is to introduce a new machine learning framework for IDEs. This is applicable to several domains arising in biology, physics, and engineering where data is sampled from systems that follow IDE-like dynamics.

While our implementation of integration makes use of *torchquad* (Gómez, Toftevaag, and Meoni 2021) and it is therefore fully supported by GPUs, hence relatively efficient, improvements can be made by implementing more advanced and more efficient integration methods.

Conclusions

We presented a novel method for modeling dynamical systems with non-instantaneous behavior. Our method, termed Neural Integro-Differential Equations (NIDE), models complex non-local dynamics based on the theory of IDEs, which are commonplace in physics and biology. To train our model, we have presented a differentiable IDE solver implemented in PyTorch. We have performed extensive experimentation on tasks such as time extrapolation and predicting dynamics of unseen initial conditions on both toy and real world data and show that NIDE outperforms other methods. To showcase real-world applications, we have used NIDE to model two brain activity recording datasets: 1) Wide-field calcium brain imaging in freely behaving mice, and 2) fMRI recordings of people on ketamine. For the calcium imaging dataset, NIDE more accurately predicts future brain states as well as unseen dynamics. In addition, a latent space learned by NIDE, via the integrand of the integral operator, provides an embedding that more accurately reflects time dynamics compared to other unsupervised embedding methods. For the fMRI data, NIDE learns a decomposition that improves the ability to classify patients' conditions. IDEs are a well-established mathematical framework and have found numerous applications in mathematics, physics, and biology. We have taken IDEs into the world of deep learning, and since many real-world dynamical systems display non-instantaneous behavior, we anticipate that there will be many applications for NIDEs.

References

Abdallah, C. G.; Ahn, K.-H.; Averill, L. A.; Nemati, S.; Averill, C. L.; Fouda, S.; Ranganathan, M.; Morgan, P. T.; D'Souza, D. C.; Mathalon, D. H.; et al. 2021. A robust and reproducible connectome fingerprint of ketamine is highly associated with the connectomic signature of antidepressants. *Neuropsychopharmacology*, 46(2): 478–485.

Abdallah, C. G.; Averill, L. A.; Collins, K. A.; Geha, P.; Schwartz, J.; Averill, C.; DeWilde, K. E.; Wong, E.; Anticevic, A.; Tang, C. Y.; et al. 2017. Ketamine treatment and global brain connectivity in major depression. *Neuropsychopharmacology*, 42(6): 1210–1219.

Abdallah, C. G.; Sanacora, G.; Duman, R. S.; and Krystal, J. H. 2018. The neurobiology of depression, ketamine and rapid-acting antidepressants: Is it glutamate inhibition or activation? *Pharmacology & therapeutics*, 190: 148–158.

Akiki, T. J.; and Abdallah, C. G. 2019. Determining the hierarchical architecture of the human brain using subject-level clustering of functional networks. *Scientific reports*, 9(1): 1–15.

Amari, S.-i. 1977. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological cybernetics*, 27(2): 77–87.

Bhatia, A.; Moza, S.; and Bhalla, U. S. 2019. Precise excitation-inhibition balance controls gain and timing in the hippocampus. *Elife*, 8: e43415.

Breakspear, M. 2017. Dynamic models of large-scale brain activity. *Nature neuroscience*, 20(3): 340–352.

Cardin, J. A.; Crair, M. C.; and Higley, M. J. 2020. Mesoscopic imaging: shining a wide light on large-scale neural dynamics. *Neuron*, 108(1): 33–43.

Chen, R. T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Chen, R. T. Q.; Amos, B.; and Nickel, M. 2021. Learning Neural Event Functions for Ordinary Differential Equations. *International Conference on Learning Representations*.

Fu, W.; and Hirsa, A. 2021. An unsupervised deep learning approach to solving partial integro-differential equations. *Quantitative Finance*, 1–14.

Gelmi, C. A.; and Jorquera, H. 2014. IDSOLVER: A general purpose solver for nth-order integro-differential equations. *Computer Physics Communications*, 185(1): 392–397.

Gómez, P.; Toftevaag, H. H.; and Meoni, G. 2021. torchquad: Numerical Integration in Arbitrary Dimensions with PyTorch. *Journal of Open Source Software*, 6(64): 3439.

Goswami, S.; Yin, M.; Yu, Y.; and Karniadakis, G. E. 2022. A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials. *Computer Methods in Applied Mechanics and Engineering*, 391: 114587.

Grigoriev, Y. N.; Ibragimov, N. H.; Kovalev, V. F.; and Meleshko, S. V. 2010. *Symmetries of integro-differential equations: with applications in mechanics and plasma physics*, volume 806. Springer.

Karpel, J. T. 2018. IDESolver: a general purpose integro-differential equation solver. *Journal of Open Source Software*, 3(21): 542.

Kim, H. 2021. Fast Bayesian Inference for Gaussian Cox Processes via Path Integral Formulation. *Advances in Neural Information Processing Systems*, 34: 26130–26142.

Kovachki, N.; Li, Z.; Liu, B.; Azizzadenesheli, K.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2021. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*.

Lakshmikantham, V. 1995. *Theory of integro-differential equations*, volume 1. CRC press.

Linden, N. J.; Tabuena, D. R.; Steinmetz, N. A.; Moody, W. J.; Brunton, S. L.; and Brunton, B. W. 2021. Go with the FLOW: visualizing spatiotemporal dynamics in optical widefield calcium imaging. *Journal of the Royal Society Interface*, 18(181): 20210523.

Lohani, S.; Moberly, A. H.; Benisty, H.; Landa, B.; Jing, M.; Li, Y.; Higley, M. J.; and Cardin, J. A. 2020. Dual color mesoscopic imaging reveals spatiotemporally heterogeneous coordination of cholinergic and neocortical activity. *BioRxiv*.

Lu, L.; Jin, P.; Pang, G.; Zhang, Z.; and Karniadakis, G. E. 2021a. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3): 218–229.

Lu, L.; Meng, X.; Mao, Z.; and Karniadakis, G. E. 2021b. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1): 208–228.

Martin, R.; Chappell, D.; Chuzhanova, N.; and Crofts, J. J. 2018. A numerical simulation of neural fields on curved geometries. *Journal of computational neuroscience*, 45(2): 133–145.

McInnes, L.; Healy, J.; and Melville, J. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Medlock, J.; and Kot, M. 2003. Spreading disease: integro-differential equations old and new. *Mathematical Biosciences*, 184(2): 201–222.

Smith, H. L. 2011. *An introduction to delay differential equations with applications to the life sciences*, volume 57. Springer New York.

Srednicki, M. 2007. *Quantum field theory*. Cambridge University Press.

Stapor, P.; Fröhlich, F.; and Hasenauer, J. 2018. Optimization and uncertainty analysis of ODE models using 2nd order adjoint sensitivity analysis. *Bioinformatics*, 34(13): i151–i159.

Stech, H. W.; Rankin, S. M.; Herdman, T. L.; et al. 1981. *Integral and functional differential equations*, volume 67. CRC Press.

Trenogin, V. 2020. *Integro-differential equation*. Encyclopedia of Mathematics. The European Mathematical Society, Springer.

Volterra, V. 1913. *Leçons on integral équations and intégro-différential equations : Leçons given àt the Faculty of Sciences in Rome in 1910*. Gauthier-Villars.

Volterra, V. 1932. Theory of functionals and of integral and integro-differential equations. *Bull. Amer. Math. Soc*, 38(1): 623.

Vyas, S.; Golub, M. D.; Sussillo, D.; and Shenoy, K. V. 2020. Computation through neural population dynamics. *Annual Review of Neuroscience*, 43: 249–275.

Wazwaz, A.-M. 2011. *Linear and nonlinear integral equations*, volume 639. Springer.

Wilson, H. R.; and Cowan, J. D. 1972. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical journal*, 12(1): 1–24.

Zappala, E.; Fonseca, A. H. d. O.; Moberly, A. H.; Higley, M. J.; Abdallah, C.; Cardin, J.; and van Dijk, D. 2022. Neural integro-differential equations. *arXiv preprint arXiv:2206.14282*.