Advances in Nonlinear Variational Inequalities
ISSN: 1092-910X
Vol 28 No. 1s (2025)

Neural Differential Equations: A Comprehensive Review and Applications

Anjali Nair[1], Dr. Sandhyatai D. Kadam[2], Dr. Radhika Menon[3], Priti C. Shende [4]
[1,2,3,4]Department of Mathematics, Dr. D. Y. Pa til Institute of Technology, Pimpri, Pune-18

Article History:

Received: *10-08-2024*

Revised: *27-09-2024*

Accepted: *17-10-2024*

Abstract:

The field of neural differential equations has witnessed significant growth in the past decade, with a notable increase in published literature. This approach offers several advantages, including the ability to model complex, nonlinear systems and the potential for extrapolation beyond the observed data. There is a need of systematic overview in implementing the use of neural networks in solving differential equations and their special cases. This work is a discussion on architectures of neural networks used in NDEs, training methodologies and applications across various domains. The current work reviews and categorizes earlier study and gives guidelines to mathematicians, researchers in computer science and engineering. Also, it includes advantages of NDEs and some current challenges and research questions in the field of NDEs.

Introduction: The field of neural differential equations has witnessed significant growth in the past decade, with a notable increase in published literature. This approach offers several advantages, including the ability to model complex, nonlinear systems and the potential for extrapolation beyond the observed data. There is a need of systematic overview in implementing the use of neural networks in solving differential equations and their special cases. This work is a discussion on architectures of neural networks used in NDEs, training methodologies and applications across various domains. The current work reviews and categorizes earlier study and gives guidelines to mathematicians, researchers in computer science and engineering. Also, it includes advantages of NDEs and some current challenges and research questions in the field of NDEs.

Conclusions: The authors have discussed the observations of ODEs, Neural network and the literature review of NDEs. The overall framework of neural networks for solving differential equations is also covered in this paper. This paper also includes general features of neural algorithms for solving differential equations. Neural networks may approximate any continuous system and stabilize the training process with the help of suitable activation functions. It allows us to solve problems more quickly without wasting processing power or memory. The parallel nature of the network helps to minimize the complexity of the problems, as demonstrated by a comparison with the exact solution, along with modeling various dynamical processes. Additionally, one can solve fractional differential equations using Neural network and many more applications based on it.

Keywords: Neural network, Activation functions, Architectures, Training methods, Advantages and applications of NDEs.

1. Introduction:

Differential equations (DEs) have been studied for centuries. Isaac Newton and Gottfried Wilhelm Leibniz laid the groundwork for calculus in the late 17th century, establishing the mathematical foundation for DEs. Both Ordinary differential equations (ODEs) and Partial differential equations (PDEs) are usually used for modeling dynamic systems in economics, physics, biology, engineering systems and many more fields.

In the 1940s and 1950s, Warren McCulloch and Walter Pitts first developed the neural network model that was later referred to as artificial neural networks (ANNs). In the 1940s McCulloch et al [31] introduced the first mathematical model of a neuron, known as the McCulloch-Pitts neuron. This was followed by Rosenblatt's (1958) perceptron model, which laid the groundwork for machine learning. Such models were later introduced by Rumelhart, Hinton, and Williams (1986). The development of backpropagation was introduced during the 1980s and 1990s. More recently, specialized neurons in convolutional neural networks (CNNs) have been created for applications like image recognition (LeCun et al., 1998). In 2002, K. Valasoulis et al [32] successfully solved the ODEs and PDEs by ANN methodology. In 2009, Loannis G Tsoulos et al [32] proposed a novel hybrid method is tried on a series of ODEs, systems of ODEs as well as on PDEs tocreate trial solutions using neural network. Additionally, the introduction of the ReLU activation function is done by Nair and Hinton (2010).

The idea of using neural networks to approximate solutions to differential equations has been explored since the 1990s. Early efforts focused on using neural networks as function approximators for solving PDEs and ODEs numerically. The need for neural differential equations arises from the desire to model and understand complex dynamic systems that are governed by differential equations. Traditional methods for modeling such systems often involve manually specifying the equations or using numerical methods to approximate them. However, these approaches can be limited by their assumptions and require significant domain knowledge. For more information on neural networks one can refer to [18],[38],[39].

NDEs focus on modeling the dynamics of neural networks using DEs. These mathematical models are needed for understanding and simulating the behavior of complex neural systems. NDEs represent a fascinating intersection between traditional differential equations and neural networks. Instead of specifying explicit rules for how a system evolves over time, NDEs learn these dynamics from data using neural networks. The ODEs illustrate the system’s dynamics, while the neural network is employed to determine the unknown parameters of the ODEs using data. The basic idea behind neural ODEs is to use neural networks to learn the parameters of ODEs, which describe the time -evolution of a system. Instead of explicitly defining the equations, neural ODEs learn a “flow “function using a neural network. This flow function describes how the system's state evolves over time. It is parameterized by the neural network, its weights and biases are learned through backpropagation during training, these backpropagation algorithms enabled the training of deeper networks, leading to the resurgence of interest in neural networks. The key difference is that instead of using traditional neural networks that operate on discreate-time data, Neural ODEs use a continuous-time formulation.

1.1Taxonomy of Literature

Issac Elias Lagaris et al [13], had introduced a technique for addressing initial and boundary value problems using ANN to solve ODEs and PDEs. Patrick Kidger [17] provides a comprehensive survey of the intersection between dynamical systems and deep learning. NDEs are particularly effective for generative problems, dynamical systems, and time series data. Zhilu Lai et al [2] propose a novel method for structural identification using Neural ODEs, incorporating domain knowledge like structural dynamics. This approach forms Physics-informed Neural ODEs, which are designed to discover or approximate governing equations. Suyong Kim et al [3] introduced new methods for

calculating derivatives to boost the stability of stiff ODE systems while minimizing computational complexity. This method combined with equation scaling and the use of deeper neural networks with rectified activation functions, facilitates the learning of neural ODEs in stiff systems. Hanshu YAN et al [4] studied the robustness of neural ODEs. It reveals that neural ODE-based models exhibit greater robustness compared to CNN models. Additionally, they explored enhancing model robustness by smoothing the loss surface, achieved by controlling the spectral norm of the Jacobian matrix of the loss function. Ricky Tian Qi Chen et al [5] explored the use of black-box ODE solvers and developed new models for time-series modeling, supervised learning, and density estimation, discussing their scope and limitations. Helen Marie McGlone [9] examined deep feedforward neural networks to solve ordinary and partial differential equations by solving a variety of initial and boundary value problems. Arunachalam Sundaram [14] utilized ANN to solve first order ODE by implementing Python and investigate possible forms of loss functions for ANN method efficiency through several numerical examples. Neha Yadav et al [18] presented various neural network methods based on multilayer perceptron, radial basis functions, multiquadric functions and finite element techniques for solving differential equations. Enze Shi et al [21] explored the approximation capabilities of neural networks in function fitting, with a focus on approximating Müntz polynomials to better understand the training behavior of the proposed neural networks. V. I. Gorikhovskii et al [24] examined the impact of different neural network topologies, optimizers, activation functions, and fast access links. This paper provides a summary on basics of NDEs in brief

This paper is a discussion on architectures of neural networks used in NDEs, training methodologies and applications across various domains.In section I and II, we have discussed the fundamentals of the DEs and neural network approaches respectively to be used while solving DEs.In the subsequent section, we have discussed different types of activation function according to the number ofneurons in hidden layer,several training techniques and optimization algorithms for effective training of NDEs are summarized. The penultimate section is devoted to the application of NDEs.

2. Overview of Differential Equations

The differential equations are mainly classified in ODEs and PDEs, ODEs involve a function that depends on a single independent variable. A partial derivative with respect to two or more independent variables and an unknown function of those variables are combined to form PDE. They are used to formulate and resolve issues involving several variables with uncertain functions. ODEs and PDEs are utilized to model a wide range of dynamical systems. Figure1 shows the categorization of dynamic systems. The concept of DEs as a mathematical model is a powerful tool to interpret physical observations, such as heat transfer or fluid movement [20]. Neural networks are used to solve ODEs and PDEs with initial and boundary value problems. The trial solution of the differential equation consists of two components. The 1st component is designed to satisfy the initial or boundary conditions, while the 2nd component involves neural networks with adjustable weights.

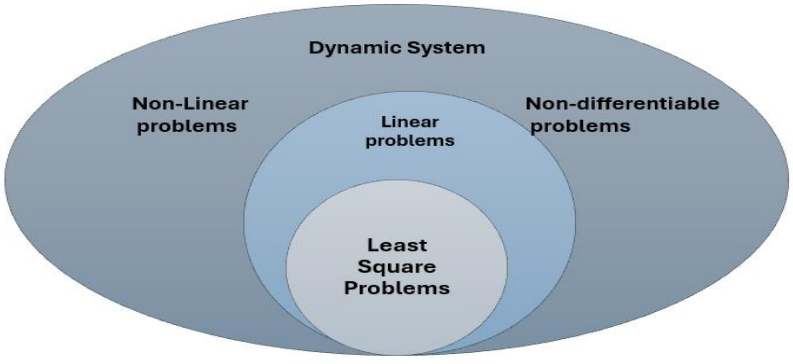

Fig 1: Categorization of dynamic systems

For more detailed study on DE one can refer to [35],[36].[37],[40].

3. Fundamentals of Neural Networks

A network diagram is usually used for demonstrating various kinds of non-linear models, all of them are grouped together under the phrase "neural network." This type of technology is modeled after the brain's biological neurons. The main concept is that each unit, or neuron, receives inputs and processes them using a weighted sum and a bias and the output is generated by passing the result through the activation functions. Neural networks learn by adjusting the weights and biases, typically through various training methods like forward and backward propagation. This involves calculating the gradient of the loss function and updating the parameters in the direction that reduces the loss. Training a neural network involves the process of minimizing the loss function. A custom loss function is defined to penalize variations from satisfying the ODE and the initial condition. The loss function is defined as a weighted sum of the ODE loss and initial condition loss.

$$\text{Loss} = \text{Loss}_B + \text{Loss}_E$$

$Loss_B$ = Boundary loss function

$Loss_E$ = Initial condition loss function.

The **loss** represents a prediction error of the neural network. The processes of calculating this error are called loss function [13].

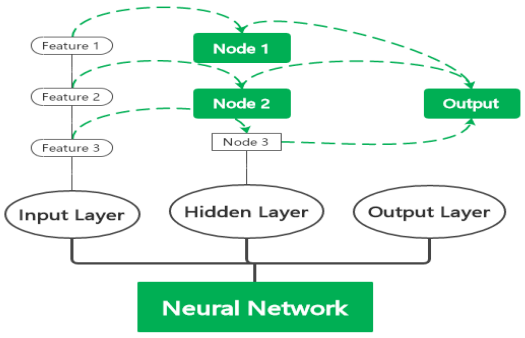

Fig 2: Representation of Neural Network

4. Neural Ordinary Differential Equations

Neural Ordinary Differential Equations (NODEs) extend the concept of layer-to-layer propagation to continuous depth models. This approach allows us to train models using ODEs by treating forward propagation in a neural network as equivalent to a one-step discretization of an ODE. A neural network works as a mathematical model for stimulating the structure and functioning of biological neural networks. An artificial neuron is the basic unit of any ANN which is a simple mathematical model or function. The three fundamental rules of this model are multiplication, summation, and activation. At the input stage, each value is multiplied by an individual weight. In the middle section, a summation function adds all the weighted inputs and a bias term. At the output stage, the sum of the weighted inputs and bias passes through an activation function, also known as a transfer function. A neural network is essentially a collection of these neurons, also referred to as activations, connected through various layers. It aims to learn the mapping from input data to output data using a provided training set.

4.1 **Mathematical Formulation for Neural Differential Equations:**

ODEs describe how the rate at which function changes is dependent on the function itself and its derivatives. General form of ODE with initial condition is given as.

$$\frac{dz}{du} = f(u,z), \qquad u \in [u_0, u_n] \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

with initial condition $z(u_0) = z_0$

Let us denote neural network functions as, $NN(u) \approx z(u)$, then

$NN'(u) \approx z'(u) = f(u,z)$ ……...by using (1)

If $NN(u) \approx z(u)$, $NN'(u) \approx z'(u)$ is near to true solution. Here we define the Loss function for equation (1) as follows.

$$L_\varphi(u) = \left\| NN_\varphi{}'(u) - NN'(u) \right\|^2 + s\left\| Z_\varphi(u_0) - Z(u_0) \right\|^2$$

Where, $\varphi =$ network parameters

s = constant coefficient

$Z_\varphi =$ predicted solution

$N_\varphi{}' =$ derivative of the predicted solution

$\left\| NN_\varphi{}'(u) - NN'(u) \right\|^2 =$ ODE loss, it measures the extent to which the predicted solution diverges from meeting the criteria of the ODE definition.

$\left\| Z_\varphi(u_0) - Z(u_0) \right\|^2 =$ initial condition loss, measure the extent to which the predicted solution deviates from satisfying the initial condition.

For example,

$\frac{dz}{du} = uz$, with $u_0 = 0, z_0 = 1$

The analytic solution of given ODE is $z = e^{\frac{u^2}{2}}$

Loss function is defined as, $L_\varphi(u) = \left\| z'_\varphi - uz \right\|^2 + s\left\| z_\varphi(0) - 1 \right\|^2$.

Where, $\left\| z'_\varphi - uz \right\|^2 =$ ODE loss

$\left\| z_\varphi(0) - 1 \right\|^2$= initial condition loss.

The first layer of neural network is basically input layer, which has nodes it indicates features for network. The second layer is a hidden layer that can have any number of neurons. All the features in the input layer must be passed through all the neurons in the hidden layer. Finally, we get the output from the output layer. Figure3 explains the steps in solving ODEs using Neural network.

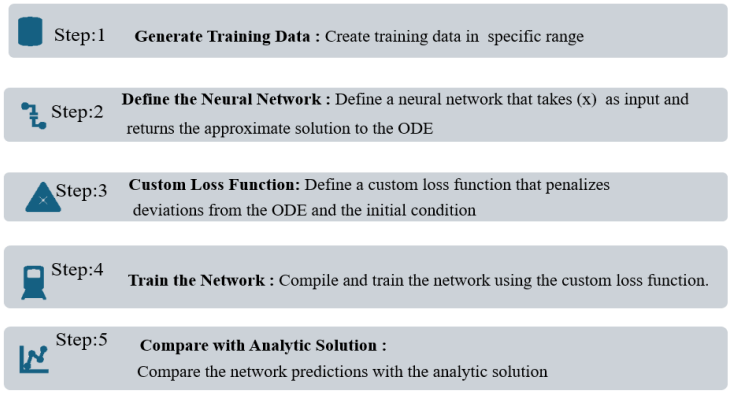

Fig 3: Steps in solving ODEs using Neural network

4.2 Activation Functions

Activation functions are vital components in neural networks as they introduce non-linearity, allowing neural networks to learn complex patterns and relationships. In hidden layer, the number of neurons decides the types of activation functions to be implemented [15]. The main activation functions are the following,

- **Sigmoid Function (Logistic Function):**

Output values lie in 0 and 1. Historically, because of vanishing gradient problem this function became less popular in hidden layers which were generally employed in binary classification problems.

$$\sigma(t) = \frac{1}{1+e^{-t}}$$

- **Hyperbolic Tangent Function (Tanh):** Output values vary from -1 and 1, like the sigmoid function but centered at 0. Often used in hidden layers of neural networks [14].

$$tanh(t) = \frac{e^{t}-e^{-t}}{e^{t}+e^{-t}}$$

- **Rectified Linear Unit:** Outputs 0 for negative inputs, and linearly increases for positive inputs. Simple and computationally efficient. The most used activation function in hidden layers due to its sparsity and effectiveness in mitigating the vanishing gradient problem is the ReLU [14].

$$ReLU(t) = max(0, t)$$

- **Leaky ReLU:** It tackles the dying ReLU issue by permitting a small, non-zero gradient when the input is negative. This guarantees that neurons can continue to learn even with negative inputs.

$$Leaky.ReLU(t) = \begin{cases} t, t > 0 \\ \alpha(e^{t} - 1), \quad otherwise \end{cases}$$

Where α is typically set to 1.

4.3 Neural Network Architecture:

Usage of neural network architectures in NODEs enhances their ability to model complex, continuous dynamics efficiently and flexibly. It consists of input, hidden, and output neurons layers. The significant changes in the activation values of the output neurons define the dynamic behavior. The various other neural network architectures are available, depending on the specific properties and requirements of different applications. Here are some common architectures used in NDEs.

- **Feedforward Neural Networks (FNNs)**: FNNs are composed of layers of neurons that are interconnected with neurons of the next layer. In the context of nonlinear dynamical equations, FNNs

can be utilized to model the system's dynamics, where the network's output represents the time derivative of the state variable.

- **Recurrent Neural Networks (RNNs):** RNNs are constructed to capture temporal dependencies in sequential data using feedback loops. In nonlinear dynamical equations, RNNs can model systems with memory or time-dependent dynamics, where the hidden state represents the system's state at each time step.
- **Long Short-Term Memory Networks (LSTMs):** LSTMs are a type of RNN precisely designed to manage long term dependencies in sequential data. In nonlinear dynamical equations, LSTMs can model systems with complex, non-local dependencies, making them suitable for tasks involving long-term memory or context information.
- **Gated Recurrent Units (GRUs):** GRUs are similar to LSTMs but feature a simplified gating mechanism, making them more computationally efficient. In nonlinear dynamical equations, GRUs can model temporal dependencies in system dynamics, particularly when computational resources are limited.
- **Convolutional Neural Networks (CNNs):** CNNs are specialized in capturing spatial dependencies in data through convolutional operations. In the context of nonlinear dynamical equations, CNNs can model systems with spatially varying dynamics or process spatially structured input data, like images or sensor measurements distributed across space.
- **Graph Neural Networks (GNNs):** In the context of nonlinear dynamical equations, GNNs can model dynamical systems represented as graphs, where nodes are system components and edges are interactions. GNNs effectively capture dependencies between components and model their evolution over time.
- **Hybrid Architectures:** Hybrid architectures integrate different types of neural networks to capture both temporal and spatial dependencies in system dynamics. For instance, combining an RNN with a CNN allows modeling systems with both temporal and spatial variations.

The choice of architecture is vital and should be customized to fit the quality of the system being modeled and the specialized requirements of the task at hand.

4.4 Techniques & Algorithms for Training Neural Differential Equations (NDEs):

Training neural differential equations (NDEs) involves optimizing the parameters of the neural networks that parameterize the differential equations to approximate the underlying dynamics of the system. Several training techniques and optimization algorithms can be employed to effectively train NDEs. These techniques and algorithms can be combined and adapted because of specific features of the NDEs and the requirements of the training task to achieve efficient and effective training of neural differential equations. Here are some commonly used methods:

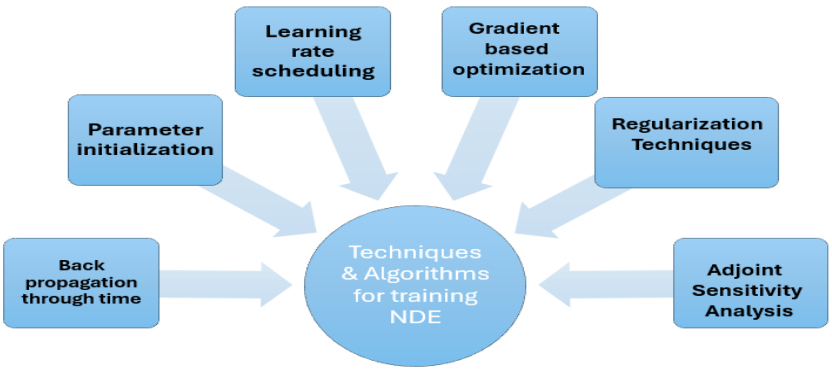

Fig 4 Training Techniques and Algorithms

5 Advantages of Neural Differential Equations

By using differential equation solvers, the optimization process of neural networks can be improved. This approach offers advantages over conventional neural network solutions, such as higher storage efficiency storage efficiency and adaptive computation [11]. The neural network-based solution for Des has several benefits. They provide a differentiable, closed analytic form that can be utilized in further calculations, unlike many other techniques that offer discrete or limited differentiability solutions. This method also boosts excellent generalization properties and maintains manageable computational complexity even with an increased number of samplings points. Additionally, they are versatile and are applicable to systems having orthogonal box boundaries or irregular, arbitrarily shaped boundaries. Neural network models can tackle complex differential equation problems in realtime, making them valuable in various scientific and engineering applications. Furthermore, this approach is applicable implemented on similar architectures, enhancing computational efficiency.

6 Application of Neural Differential Equations:

- NDEs are designed to simulate continuous-time dynamics, they offer a variable framework for capturing complicated and nonlinear relationships in dynamic systems.
- Integrating neural networks with ODEs enables the incorporation of neural networks into solving ODEs.
- NDEs are ideal for modeling dynamic systems and analyzing time-series data. They help capture the evolution of variables over time and provide insights into the underlying mechanisms driving observed behaviors.
- NDEs have been employed in generative modeling tasks, allowing for the generation of data samples that follow continuous-time dynamics. This is useful in scenarios where the data evolves smoothly over time.
- NDEs can be used for parameter estimation and system identification in scenarios where the underlying system is described by differential equations. This is valuable for understanding and characterizing real-world processes.
- Researchers are exploring hybrid methodologies that merge neural networks with traditional numerical methods for solving differential equations. This synergy can lead to more efficient and accurate solutions for certain classes of problems.
- NDEs enhance the field of machine learning for scientific discovery by developing models that learn from data while adhering to the fundamental laws of physical systems.
- NDEs provide a bridge between traditional differential equations and deep learning, allowing for the incorporation of neural networks into models that require a continuous-time representation.
- NDEs provide a bridge between traditional differential equations and deep learning, allowing for the incorporation of neural networks into models that require a continuous-time representation.

7. Challenges

The field of neural differential equations (NDEs) has made significant progress in recent years, but several challenges and open research questions remain. Addressing the challenges and open research questions will advance the field of NDEs and enable the expansion of more accurate, interpretable, and consistent models for modeling complex dynamical systems in various domains. Following are the challenges:

1. **Scalability**: One of the major challenges in neural differential equations (NDEs) is scalability, especially for large-scale systems with high-dimensional state spaces or long-time horizons. Training neural networks to approximate the dynamics of such systems can be computationally intensive and

requires sophisticated optimization techniques. Developing scalable algorithms and architectures that can efficiently handle large-scale NDEs remains an open research question.

2. **Interpretability and Explainability**: Neural networks employed in Neural Differential Equations (NDEs) are frequently viewed as black-box models, which complicates the task of interpreting and elucidating the dynamics they learn. Understanding the underlying mechanisms and relationships captured by NDEs is crucial for building trust in their predictions and enabling meaningful insights into the behavior of dynamical systems. Developing methods for interpreting and explaining the learned dynamics of NDEs is an important research direction for improving their transparency and interpretability.

3. **Generalization**: Generalization refers to the ability of NDEs to accurately model and predict the behavior of unseen or out-of-distribution data.Ensuring that NDEs generalize well across different scenarios, system configurations, and operating conditions is essential for their practical applicability.Addressing issues such as overfitting, robustness to noise and uncertainties, and transfer learning is critical for improving the generalization performance of NDEs.

4. **Incorporating Domain Knowledge**: Integrating domain-specific knowledge into Neural Differential Equations (NDEs) can enhance their effectiveness, clarity, and relevance to particular problem areas. Leveraging domain-specific insights, constraints, and priors can guide the design of more effective architectures, training methodologies, and regularization techniques for NDEs.Developing frameworks for integrating domain knowledge into the modeling and training process of NDEs is an active area of research.

5. **Handling Nonlinearity and Complexity**:Many real-world systems exhibit nonlinear and complex dynamics, posing challenges for modeling and prediction using NDEs.Designing neural network architectures and training algorithms that can capture and approximate the nonlinear and complex interactions present in such systems is essential. Exploring novel architectures, such as deep neural networks, attention mechanisms, and graph neural networks, tailored to handle nonlinear and complex dynamics is an important research direction.

6. **Uncertainty Quantification**: Quantifying uncertainty in the predictions of NDEs is crucial for assessing their reliability and making informed decisions in uncertain or high-stakes applications. Developing probabilistic frameworks and uncertainty estimation techniques for NDEs that account for aleatoric and epistemic uncertainties is an active area of research. Integrating uncertainty quantification techniques into the training and inference stages of NDEs can enhance their robustness and reliability in practical applications.

8. Development of NODEs Over a Decade

Here we are enlisted the list of research papers on NDEs till 2023.

Sr. No	Title	Authers	Year	Citations
1	NICE: Non-linear Independent Components Estimation	Laurent Dinh, David Krueger, Yoshua Bengio	2014	2007
2	Variational Inference with Normalizing Flows	Danilo Jimenez Rezende, S. Mohamed	2015	3601

3	Improved Variational Inference with Inverse Autoregressive Flow	Diederik P. Kingma, Tim Salimans, M. Welling	2016	1675
4	Density estimation using Real NVP	Laurent Dinh, Jascha Narain Sohl-Dickstein, Samy Bengio	2016	3218
5	Reversible Architectures for Arbitrarily Deep Residual Neural Networks	B. Chang, Lili Meng, E. Haber, Lars Ruthotto, David Begert, E. Holtham	2017	242
6	Stable architectures for deep neural networks	E. Haber, Lars Ruthotto	2017	645
7	A Proposal on Machine Learning via Dynamical Systems	Weinan E	2017	668
8	Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical	Yiping Lu, Aoxiao Zhong, Quanzheng Li, Bin Dong	2017	449
9	Training Glow with Constant Memory Cost	Xuechen Li	2018	17
10	Sylvester Normalizing Flows for Variational Inference	Rianne van den Berg, Leonard Hasenclever, Jakub M. Tomczak, M. Welling	2018	236
11	Neural Importance Sampling	T. Müller, B. McWilliams, Fabrice Rousselle, M. Gross, Jan Novák	2018	307
12	Deep Neural Networks Motivated by Partial Differential Equations	Lars Ruthotto, E. Haber	2018	428
13	Invertible Residual Networks	Jens Behrmann, D. Duvenaud, J. Jacobsen	2018	550
14	FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models	Will Grathwohl, Ricky T. Q. Chen, J. Bettencourt, I. Sutskever, D. Duvenaud	2018	746
15	Neural Ordinary Differential Equations	T. Chen, Yulia Rubanova, J. Bettencourt, D. Duvenaud	2018	4011
16	Neural Networks with Cheap Differential Operators	Ricky T. Q. Chen, D. Duvenaud	2019	33
17	Invertible Convolutional Flow	Mahdi Karami, D. Schuurmans, Jascha Narain Sohl-Dickstein, Laurent Dinh, Daniel Duckworth	2019	42
18	Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows	R. Cornish, Anthony L. Caterini, George Deligiannidis, A. Doucet	2019	114
19	Cubic-Spline Flows	Conor Durkan, Artur Bekasov, Iain Murray, G. Papamakarios	2019	53

20	On Robustness of Neural Ordinary Differential Equations	Hanshu Yan, Jiawei Du, V. Tan, Jiashi Feng	2019	121
21	PointFlow: 3D Point Cloud Generation with Continuous Normalizing Flows	Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge J. Belongie, Bharath Hariharan	2019	561
22	Modelling Dynamical Systems Using Neural Ordinary Differential Equations	D. Karlsson, Olle Svanström	2019	10
23	Latent ODEs for Irregularly Sampled Time Series	Yulia Rubanova, Ricky T. Q. Chen, D. Duvenaud	2019	228
24	Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design	Jonathan Ho, Xi Chen, A. Srinivas, Yan Duan, P. Abbeel	2019	409
25	Block Neural Autoregressive Flow	Nicola De Cao, Ivan Titov, Wilker Aziz	2019	112
26	Unconstrained Monotonic Neural Networks	Antoine Wehenkel, Gilles Louppe	2019	131
27	Hamiltonian Neural Networks	S. Greydanus, Misko Dzamba, J. Yosinski	2019	741
28	Emerging Convolutions for Generative Normalizing Flows	Emiel Hoogeboom, Rianne van den Berg, Max Welling	2019	93
29	GRU-ODE-Bayes: Continuous modeling of sporadically observed time series	E. Brouwer, J. Simm, Adam Arany, Y. Moreau	2019	245
30	Residual Flows for Invertible Generative Modeling	Ricky T. Q. Chen, Jens Behrmann, D. Duvenaud, J. Jacobsen	2019	338
31	Latent Ordinary Differential Equations for Irregularly Sampled Time Series	Yulia Rubanova, T. Chen, D. Duvenaud	2019	476
32	Augmented Neural ODEs	Emilien Dupont, A. Doucet, Y. Teh	2019	536
33	OT-Flow: Fast and Accurate Continuous Normalizing Flows via Optimal Transport	Derek Onken, Samy Wu Fung, Xingjian Li, Lars Ruthotto	2020	119
34	The Convolution Exponential and Generalized Sylvester Flows	Emiel Hoogeboom, Victor Garcia Satorras, Jakub M. Tomczak, M. Welling	2020	27

35	Woodbury Transformations for Deep Generative Flows	You Lu, Bert Huang	2020	16
36	Dissecting Neural ODEs	Stefano Massaroli, Michael Poli, Jinkyoo Park, A. Yamashita, H. Asama	2020	169
37	Scalable Gradients for Stochastic Differential Equations	Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, D. Duvenaud	2020	258
38	A Temporal Kernel Approach for Deep Learning with Continuous-time Information	Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, Kannan Achan	2021	4
39	Do Residual Neural Networks discretize Neural Ordinary Differential Equations.	Michael E. Sander, Pierre Ablin, Gabriel Peyré	2022	19
40	A Survey on Solving and Discovering Differential Equations Using Deep Neural Networks	Hyeonjung (Tari)Jung, Jayant Gupta, Bharat Jayaprakash, Matthew Eagon, Harish Panneer Selvam, Carl Molnar, William Northrop, Shashi Shekhar	2023	3

8.1 Connected Graph of Above Listed Papers.

Connected Papers will generate a visual graph of similar papers based on co-citation and bibliographic coupling. Each node (bubble) represents a paper, and the connections between nodes indicate the relationship between the papers. Here, we listed some papers regarding neural differential equations and their development works in a connected graph. The main paper we considered as [5] in big node in connected graph. Remaining nodes gives year wise developed paper works based on the original paper. A detailed list of papers along with title, authors and their citations are enclosed in the below graph.

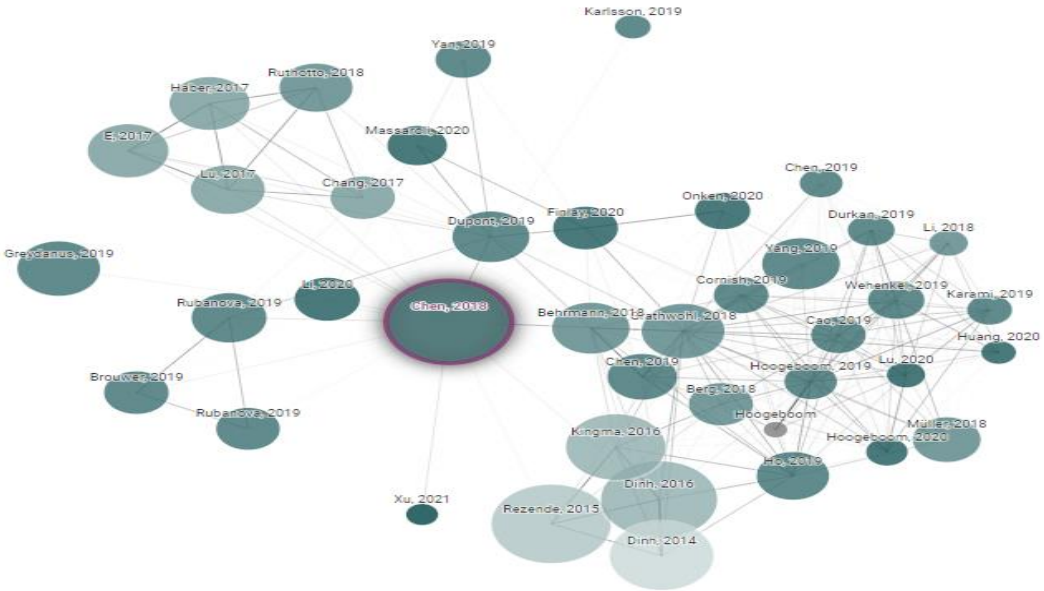

Fig 5 Connected graph

Conclusion

The authors have discussed the observations of ODEs, Neural network and the literature review of NDEs. The overall framework of neural networks for solving differential equations is also covered in this paper. This paper also includes general features of neural algorithms for solving differential equations. Neural networks may approximate any continuous system and stabilize the training process with the help of suitable activation functions. It allows us to solve problems more quickly without wasting processing power or memory. The parallel nature of the network helps to minimize the complexity of the problems, as demonstrated by a comparison with the exact solution, along with modeling various dynamical processes. Additionally, one can solve fractional differential equations using Neural network and many more applications based on it.

References:

[1]. Alexander Norcliffe', Cristian Bodnar, Ben Day, Jacob Moss & Pietro Lio,'Neural ODE Processes', https://doi.org/10.48550/arXiv.2103.12413, Aug 2021.

[2]. Zhilu Lai a,b, Charilaos Mylonas a ,Satish Nagarajaiahc and Eleni Chatzi a,b,'Structural identification with physics-informed neural ordinary differential equations', https://doi.org/10.1016/j.jsv.2021.116196 ,Volume 508, September 2021.

[3]. Suyong Kim, Weiqi Ji, Sili Deng, Yingbo Ma,2 and Christopher Rackauckas,'Stiff Neural Ordinary Differential Equations', https://arxiv.org/abs/2103.15341, Sep 2021.

[4]. Hanshu Yan, Jiawei DU, Vincent Y. F. Tan & Jiashi Feng, 'On Robustness Of Neural Ordinary Differential Equations', https://arxiv.org/abs/1910.05513, Mar 2022.

[5]. Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, David Duvenaud, **'Neural Ordinary Differential Equations'** , https://arxiv.org/abs/1806.07366, Dec 2019.

[6]. Viktor Oganesyan, Alexandra Volokhova and Dmitry Vetrov, 'Stochasticity in Neural ODEs: An Empirical Study', https://arxiv.org/abs/2002.09779, Jun 2020.

[7]. Katharina Ott Bosch, Michael Tiemann Bosch, Renningen Philipp Hennig,'Uncertainty and Structure in Neural Ordinary Differential Equations', https://arxiv.org/abs/2305.13290, May 2023.

[8]. R. Devipriya, S. Selvi 'Modelling and Solving Differential Equations using Neural Networks' A Study, Vol. 10: No. June 2020.

[9]. Helen Marie McGlone, 'Neural Network Analysis in Higgs Search using ttH, H → bb and Tag Database Development for ATLAS', https://theses.gla.ac.uk/id/eprint/71226, 2009.

[10]. Maria Laura Piscopo, Michael Spannowsky, and Philip Waite, 'Solving differential equations with neural networks: Applications to the calculation of cosmological phase transitions', https://doi.org/10.1103/PhysRevD.100.016002, Feb 2019.

[11]. Pramod Kumar Parida, 'Artificial Neural Network Based Numerical Solution of Ordinary Differential Equations', https://core.ac.uk/download/pdf/53188453.pdf, 2012.

[12]. Haoxuan Li, 'The advance of neural ordinary differential ordinary differential equations', http://dx.doi.org/10.54254/2755-2721/6/20230709, June 2023.

[13]. Isaac Elias Lagaris, Aristidis Likas, Member, IEEE, and Dimitrios I. Fotiadis'Artificial Neural Networks for Solving Ordinary and Partial Differential Equations', https://doi.org/10.1109/72.712178, September 1998.

[14]. Arunachalam Sundaram, 'Applications of Artificial Neural Networks to Solve Ordinary Differential Equations', https://doi.org/10.22214/ijraset.2022.40413, Feb 2022.

[15]. Pau Baldillou Salse, 'An Introduction to Neural Ordinary Differential Equations',2024.

[16]. Saeed Althubiti, Manoj Kumar, Pranay Goswami & Kranti Kumar,'Artificial neural network for solving the nonlinear singular fractional differential equations', https://doi.org/10.1080/27690911.2023.2187389, March 2023.

[17]. Patrick Kidger 'On Neural Differential Equations', A thesis submitted for the degree of Doctor of Philosophy Trinity 2021.

[18]. Neha Yadav, Anupam Yadav Manoj Kumar, 'An Introduction to Neural Network Methods for Differential Equations', http://dx.doi.org/10.1007/978-94-017-9816-7, March 2015.

[19]. Fangxue Zhang, Xinping Xiao, Mingyun Gao, 'An extended neural ordinary differential equation network with grey system and its applications', https://doi.org/10.1016/j.neucom.2024.127343, April 2024.

[20]. Hyeonjung (Tari)Jung, Jayant Gupta, Bharat Jayaprakash, Matthew Eagon, Harish Panneer Selvam, Carl Molnar, William Northrop, Shashi Shekhar 'A Survey on Solving and Discovering Differential Equations Using Deep Neural Networks' https://doi.org/10.48550/arXiv.2304.13807, April 2023.

[21]. Enze Shi1 and Chuanju Xu,'A comparative investigation of neural networks in solving differential equations; http://dx.doi.org/10.1177/1748302621998605, March 2021.

[22]. Sebastian Herzog, Florentin Wörgötter, 'Application of neural ordinary differential equations to the prediction of multi-agent Systems' https://doi.org/10.1007/s10015-021-00719-6, 2022.

[23]. Jan Blechschmidt | Oliver G. Ernst Three, 'Ways to Solve Partial Differential Equations with Neural Networks — A Review', https://doi.org/10.48550/arXiv.2102.11802, Feb 2021.

[24]. V I Gorikhovskii, T O Evdokimova and V.A Poletansky 'Neural networks in solving differential equations',10.1088/1742-6596/2308/1/012008, 2022.

[25]. Alexander Norcliffe, Cristian Bodnar, Ben Day, Nikola Simidjievski, Pietro Liò 'On Second Order Behavior in Augmented Neural ODEs', https://openreview.net/pdf?id=XpmaGtI04ki, July 2024.

[26]. Patrick Kidger, James Morrill, James Foster and Terry Lyons,'Neural Controlled Differential Equations for Irregular Time Series', https://doi.org/10.48550/arXiv.2005.08926, Nov 2020.

[27]. Xingzi Xu, Ali Hasan†, Khalil Elkhalil, Jie Ding, and Vahid Tarokh,'Characteristic Neural Ordinary Differential Equations', https://doi.org/10.48550/arXiv.2111.13207, Nov 2021.

[28]. YongKyung Oh, Dongyoung Lim, Sungil Kim,'Invertible Solution of Neural Differential Equations for Analysis of Irregularly-Sampled Time Series', https://doi.org/10.48550/arXiv.2401.04979, Jan 2024.

[29]. Gavin D. Portwood, Peetak P. Mitra, Mateus Dias Ribeiro, Tan Minh Nguyen, Balasubramanya T. Nadiga, Juan A. Saenz, Michael Chertkov, Animesh Garg, Anima Anandkumar, Andreas Dengel, Richard Baraniuk, David P. Schmidt,'Turbulence forecasting via Neural ODE,' https://doi.org/10.48550/arXiv.1911.05180, Nov 2019.

[30]. Michael E. Sander, Pierre Ablin, Gabriel Peyré,'Do Residual Neural Networks discretize Neural Ordinary Differential Equations.' https://doi.org/10.48550/arXiv.2205.14612, May 2022.

[31]. Warren s. Mcculloch and Walter pitts 'A logical calculus of the ideas immanent in nervous activity' https://doi.org/10.1007/BF02478259, Dec 1943.

[32]. Loannis G Tsoulos, Dimitris G, E. Glavas,'Solving differential equations with constructed neural networks', http://dx.doi.org/10.1016/j.neucom.2008.12.004, June 2009.

[33]. B. Ph. van Milligen, V. Tribaldos, and J. A. Jiménez,'Neural Network Differential Equation and Plasma Equilibrium Solver', https://doi.org/10.1103/PhysRevLett.75.3594, November 1995.

[34]. K. Valasoulis, Dimitrious I Fotiadis, Isaac Lagaris, Aristidis likas, 'Solving differential equations with neural networks: implementation on a DSP platform, http://dx.doi.org/10.1109/ICDSP.2002.1028323, Feb 2002.

[35]. Shair Ahmad Antonio Ambrosetti,'A Textbook on Ordinary Differential Equations', 10.1007/978-3-319-16408-3, Springer Volume 88, 2015.

[36]. Jeffrey R. Chasnov, Differential Equations, 2009.

[37]. Gabriel Nagy, Ordinary Differential Equations, January 2021.

[38]. Simon Haykin, Neural Networks and Learning Machines, 1999.

[39]. Kevin Gurney University of Sheffield, An Introduction to Neural networks, 1997.

[40]. E. C. Zachmanoglou and Dale W. Thoe, Introduction to Partial Differential Equations with Applications, 1986