iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** II **Month of publication:** February 2022

DOI: https://doi.org/10.22214/ijraset.2022.40413

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue II Feb 2022- Available at www.ijraset.com

Applications of Artificial Neural Networks to Solve Ordinary Differential Equations

Arunachalam Sundaram
Department of Mathematics, Rizvi College of Arts, Science and Commerce, University of Mumbai, Mumbai, Maharashtra, India.

Abstract: ***Applications of Neural Networks to numerical problems have gained increasing interest. Solving Ordinary Differential Equations can be realized with simple artificial neural network architectures. In this paper, first step is training a Neural Network to satisfy the condition required by a differential equation and then finding a function whose derivative satisfies the Ordinary Differential Equation condition. The method of solving differential equation is implemented in Python Programming using the TensorFlow library.***
Keywords: Neural Network, Differential Equation, Loss function, Train function, Mean Squared Error, TensorFlow.

I. INTRODUCTION

Differential equations play an important role in various fields such as Science and Engineering. Numerical approaches to solve the differential equation have been studied by different researchers. Several methods have been developed to solve the differential equations. Some of the methods to produce a solution in the form of array that contains the value of the solution at a selected group of points and the other methods use basis functions to find the solution in analytic form. However, there is a need to develop more efficient and universal method to solve the differential equations. The connection between Mathematics and physical phenomena is often formulated by differential equations which combine the function and their derivatives to produce a mathematical model of real world problems [10].

With the emerging field of Computer Science and Scientific Computing, Artificial Neural Networks is considered as one of such methods. Algorithms based on ANN have been proposed for solving first order differential equations. Later on the algorithm was developed based on feedforward neural networks for solving ordinary differential equations [7,8]. Lagaris et al. proposed algorithms for solving Partial differential equations on regular and irregular domains. The focus on algorithm is to use output of a single layer neural network to construct numerical solutions that satisfy the boundary value problem. This paper is organized as follows. Section II discusses the related work, Section III focuses on Ordinary Differential Equations, Section IV presents Artificial Neural Networks, Section V discusses about Mathematical Formulation for Neural Network Function, Section VI shows Python Implementation and Results and finally the Conclusion is given in Section VII.

II. RELATED WORK

Issac Elias Lagaris et al., have presented a method to solve Initial Value and Boundary Value Problems using Artificial Neural Networks for solving Ordinary Differential equation and Partial Differential Equations. The idea of solving an Ordinary Differential Equation using a Neural Network was first described by them [1]. Marco Di Giovanni et al., have designed a novel algorithm to find the solutions of a differential equation with non-unique solution using neural networks and tested the accuracy of applying neural networks to the non-linear differential equation [9].

Liam L.H. Lan and Dem's Werth have explored the method of using Neural Networks to find the solution of differential equations and found that the Neural Networks was able to perform better than the typical numerical solutions for differential equation such as for highly oscillating solutions [4]. Toni Schneidereit and Michael Breus have investigated several parameters and constant versus random weight initialization for two solution methods for solving Ordinary Differential Equations using Artificial Neural Networks [3]. Susmita Mall and S. Chakraverty have investigated the solution of Ordinary Differential Equations with initial conditions using Regression based algorithm and compared the different Neural Network architectures corresponding to different regression models [5].

Enze Shi and Chuanju Xu have discussed the possible forms of loss function in an Artificial Neural Networks method for solving differential equations and investigated their efficiency through examples [2]. Forough Arabshahi et al., have presented a novel approach for solving differential equations using Neural Networks [6].

©IJRASET: All Rights are Reserved

III. ORDINARY DIFFERENTIAL EQUATIONS (ODE)

An Ordinary Differential Equation is a relation between a function, its derivatives and the variable upon which they depend. The general form of an Ordinary Differential Equation is given by

$$F(t, y, y', y'', \dots\dots, y^{(n)}) = 0 \quad \dots\dots\dots\dots (1)$$

where n represents the highest order of derivative, y and its derivatives are functions of t.

A linear differential equation of order n can be expressed in the form

$$\sum_{k=0}^{n} F_k(t) y^{(k)}(t) = f(t) \dots\dots (2)$$

in which $F_k(t)$ are known functions.

A general solution of Ordinary Differential Equation such as (1) is a relation between y , t *and* n arbitrary constants which satisfies the equation. The solution may be an implicit relation of the form

$$w(t, y, c_1, c_2, \dots\dots, c_n) = 0 \dots\dots\dots\dots (3)$$

or an explicit function of t of the form

$$y = w(t, c_1, c_2, \dots\dots, c_n) \dots\dots\dots\dots (4)$$

The n arbitrary constants $c_1, c_2, \dots\dots, c_n$ can be determined by prescribing n conditions of the form

$$y^{(n)}(t_0) = \varphi_n \; where \; n = 0,1,2, \dots\dots, n-1 \dots\dots\dots\dots\dots\dots (5)$$

at one point $t = t_0$ which are called initial conditions. The point t_0 is called an initial point. The differential equation (1) together with the initial conditions (5) is called an n^{th} order initial value problem. The first order differential equation is of the form

$$\frac{dy}{dx} = y' = f(x, y) \; with \; initial \; conditions \; y(x_0) = y_0 \dots\dots\dots (6)$$

IV. ARTIFICIAL NEURAL NETWORKS (ANN)

An Artificial Neural Network (ANN) is a mathematical model that tries to simulate the structure and functionalities of biological neural networks. Basic building block of every artificial neural network is artificial neuron which is a simple mathematical model or function. Such a model has three simple sets of rules: multiplication, summation and activation. At the entrance of artificial neuron the inputs are weighted what means that every input value is multiplied with individual weight. In the middle section of artificial neuron is sum function that sums all weighted inputs and bias. At the exit of artificial neuron the sum of previously weighted inputs and bias is passing through activation function that is also called transfer function [11].

Artificial Neural Network (ANN) model involves computations and mathematics, which simulate the human–brain processes. Many of the recently achieved advancements are related to the artificial intelligence research area such as image and voice recognition, robotics using ANN. The ANN models have the specific architecture format, which is inspired by a biological nervous system. Like the structure of the human brain, the ANN models consist of neurons in a complex and nonlinear form. The neurons are connected to each other by weighted links. All the processes in ANN models, such as data collection and analysis, network structure design, number of hidden layers, network simulation, and weights/bias trade-off are computed through learning and training methods. A neural network is simply a collection of Neurons which is also known as activations that are connected through various layers. It attempts to learn the mapping of input data to output data on being provided a training set.

The training of the neural network later facilitates the predictions made by it on a testing data of the same distribution. This mapping is attained by a set of trainable parameters called *weights*, distributed over different layers. The weights are learned by the backpropagation algorithm whose aim is to minimize a loss function. A loss function measures how distant the predictions made by the network are from the actual values. Every layer in a neural network is followed by an activation layer that performs some additional operations on the neurons.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue II Feb 2022- Available at www.ijraset.com

The Loss Function is one of the important components of Neural Networks. Loss is nothing but a prediction error of Neural Net. And the method to calculate the loss is called Loss Function. In simple words, the Loss is used to calculate the gradients. Gradients are used to update the weights of the Neural Net. Tensorflow library has various inbuilt loss functions for different objectives. Mean Squared Error loss is used for regression tasks. This loss is calculated by taking the mean of squared differences between actual and predicted values.

Mathematically speaking, any neural network architecture aims at finding any mathematical function $y = f(x)$ that can map attributes (x) to output (y). The accuracy of mapping of this function differs depending on the distribution of the dataset and the architecture of the network employed. The Universal Approximation Theorem tells us that Neural Networks has a kind of universality. For any function $f(x)$, there is a network that can approximately approach the result. This result holds for any number of inputs and outputs.

V. MATHEMATICAL FORMULATION FOR NEURAL NETWORK FUNCTION

Let us define a function whose derivative satisfies the ordinary differential equation with initial condition:

$$u' = f(u,t), t \in [0,1]; u(0) = u_0 \dots\dots\dots\dots\dots\dots\dots (7)$$

Since neural networks are known as universal approximators, we can consider this property of neural network to use them to approximate the solution of given ordinary differential equation. Let us denote the neural network function as $NN(t) \approx u(t)$.

Differentiating, $NN'(t) = u'(t) = f(u,t) = f(NN(t),t)(using\ equation\ (7))$

If $NN(t)$ is close to the true solution, then its derivative is also close to the true solution.

That is $NN(t) \approx f(u,t),\ t \in [0,1]$

Hence, we can use this condition into our loss function. Now, we can calculate the Neural Network derivative $NN'(t)$ at each step for the given derivative function $f(u,t)$. Let us define the loss function which is the mean squared error of the two values.

$$L = \sqrt{\left[\sum_{i=1}^{n} \frac{dNN(t_i)}{dt} - f(u(t_i),t_i)\right]^2} \dots\dots\dots (8)$$

By adding the initial condition term to the cost function

$$L = \sqrt{(NN(0) - u_0)^2} + \left[\sqrt{\left[\sum_{i=1}^{n} \frac{dNN(t_i)}{dt} - f(u(t_i),t_i)\right]^2}\right] \dots\dots\dots (9)$$

The importance of loss function on the training of the neural network, the number of terms in the loss function will impact directly the stability of our training. More terms on this loss function would imply unstable training usually. To avoid this, we can encode the initial condition into the loss in a better way. Let us define a new function and use it directly instead of using neural network.

$$g(t) = u_0 + t\left(NN(t)\right) \dots\dots\dots.. (10)$$

We observe that $g(t)$ satisfies the initial condition since $g(0)$ will lead to $t\left(NN(t)\right) = 0$.

We can train $g(t)$ to satisfy the ordinary differential equation system instead of neural network then it will be a solution to the derivative function.

Let us define the new loss function

$$L = \sqrt{\left[\sum_{i=1}^{n} \frac{dg(t_i)}{dt} - f(u(t_i),t_i)\right]^2} \dots\dots\dots (11)$$

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue II Feb 2022- Available at www.ijraset.com

VI. PYTHON IMPLEMENTATION AND RESULTS

To implement the described method in Python using TensorFlow library, we will use a low level design to avoid a number of possible optimizations provided in the library. Our focus is to implement the solution of ordinary differential equation by using Neural Network. Let us consider the first order differential equation

$$u' = 3x^2; u(0) = 1 \dots\dots\dots\dots\dots\dots\dots (12)$$

To solve this differential equation, integrating both sides of equation (12), we obtain

$$u = x^3 + C \dots\dots\dots\dots (13)$$

By applying the initial condition, the solution is

$$u = x^3 + 1 \dots\dots\dots\dots (14)$$

Instead of solving analytically, let us solve the equation (12) using Neural Network. We will create a Multilayer Perceptron (MLP) Neural Net with two hidden layers, sigmoid activation functions and a gradient descent optimizer algorithm.

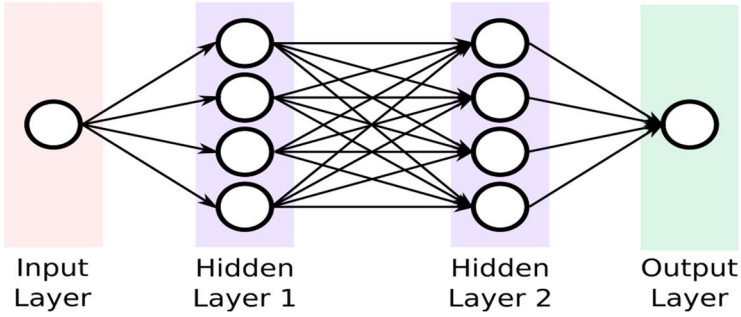


Fig. 1 Neural network architecture with two hidden layers

Note that we do not have any parameters on our loss function. The loss function usually would compare the prediction with actual data. In this case, we do not require data points. In this model, we can calculate the expected value at each point x. We always calculate loss using 10 points in the interval $[0,1]$. The Neural network derivative is expressed by the variable dNN in the code. In Python, we first define the variables. Next, we define the model and loss function. Then we define train function and Python is implemented for plotting the results.

Problems	First Order Differential Equation	Initial Condition	Solution
1	$u' = 3x^2$	$u(0) = 1$	$u = x^3 + 1$
2	$u' = 2x$	$u(0) = 1$	$u = x^2 + 1$
3	$u' = 4x^3$	$u(0) = 1$	$u = x^4 + 1$
4	$u' = 5x^4$	$u(0) = 1$	$u = x^5 + 1$
5	$u' = 3x^2$	$u(0) = 2$	$u = x^3 + 2$
6	$u' = 4x^3$	$u(0) = 3$	$u = x^4 + 3$
7	$u' = 4x + 3$	$u(0) = 0$	$u = 2x^2 + 3x$

Table 1 List of the Problems Solved using Neural Network

We have applied ANN for various first order differential equations with initial conditions. Table1 shows the list of problems solved using neural network. The solution for all the problems is obtained by implementing Python and the graphs are shown below in Fig. 2 to Fig. 8.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

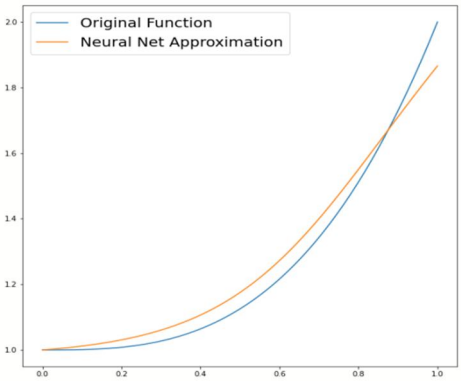


Fig. 2 Graph of $u = x^3 + 1$

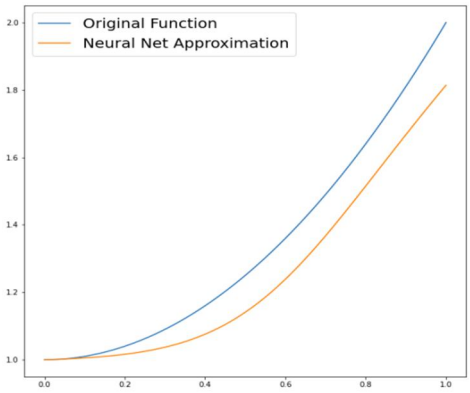


Fig. 3 Graph of $u = x^2 + 1$

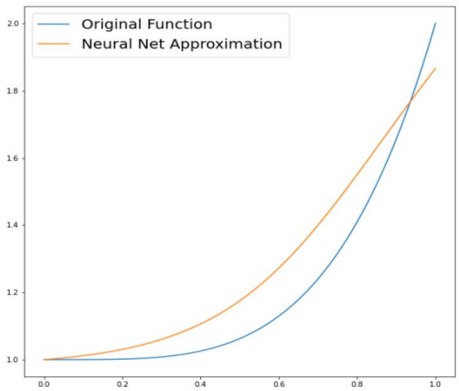


Fig. 4 Graph of $u = x^4 + 1$

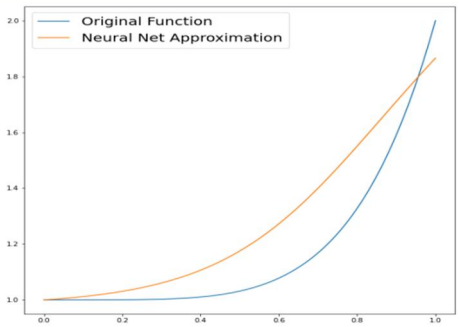


Fig. 5 Graph of $u = x^5 + 1$

©IJRASET: All Rights are Reserved

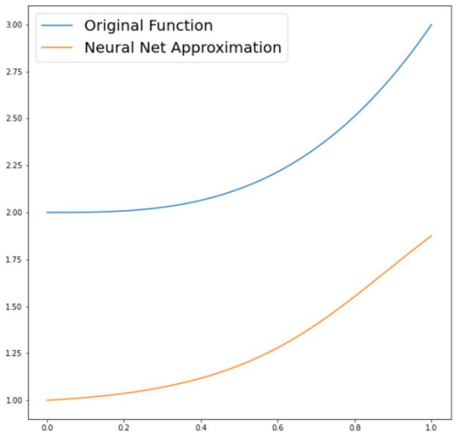


Fig. 6 Graph of $u = x^3 + 2$

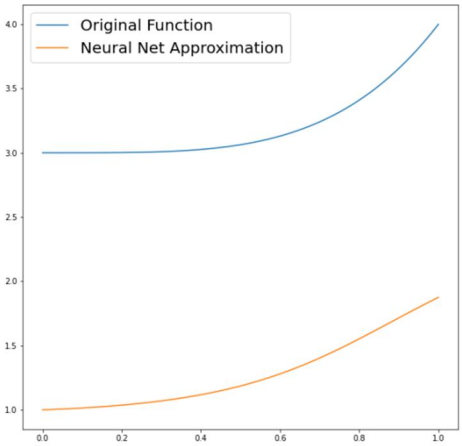


Fig. 7 Graph of $u = x^4 + 3$

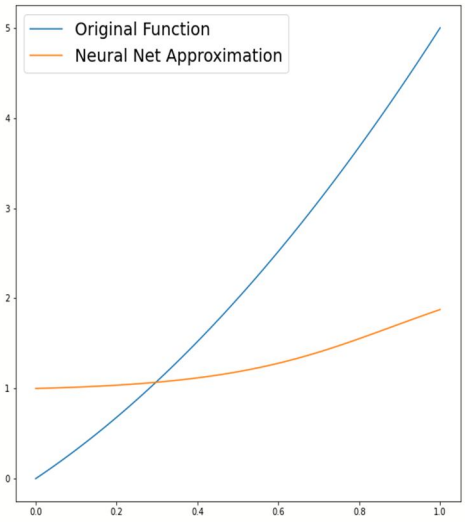


Fig. 8 Graph of $u = 2x^3 + 3x$

We observe that for problems 1 to 4 from Table 1, the initial conditions $u(0) = 1$ are same for all four problems and the graphs are shown in Fig. 2 to Fig. 5. Problems 5 to 7 from Table 1, the initial conditions are changed and the graphs are shown in Fig. 6 to Fig. 8. The graphs show that the original function and Neural network approximation of the solution of first order differential equations.

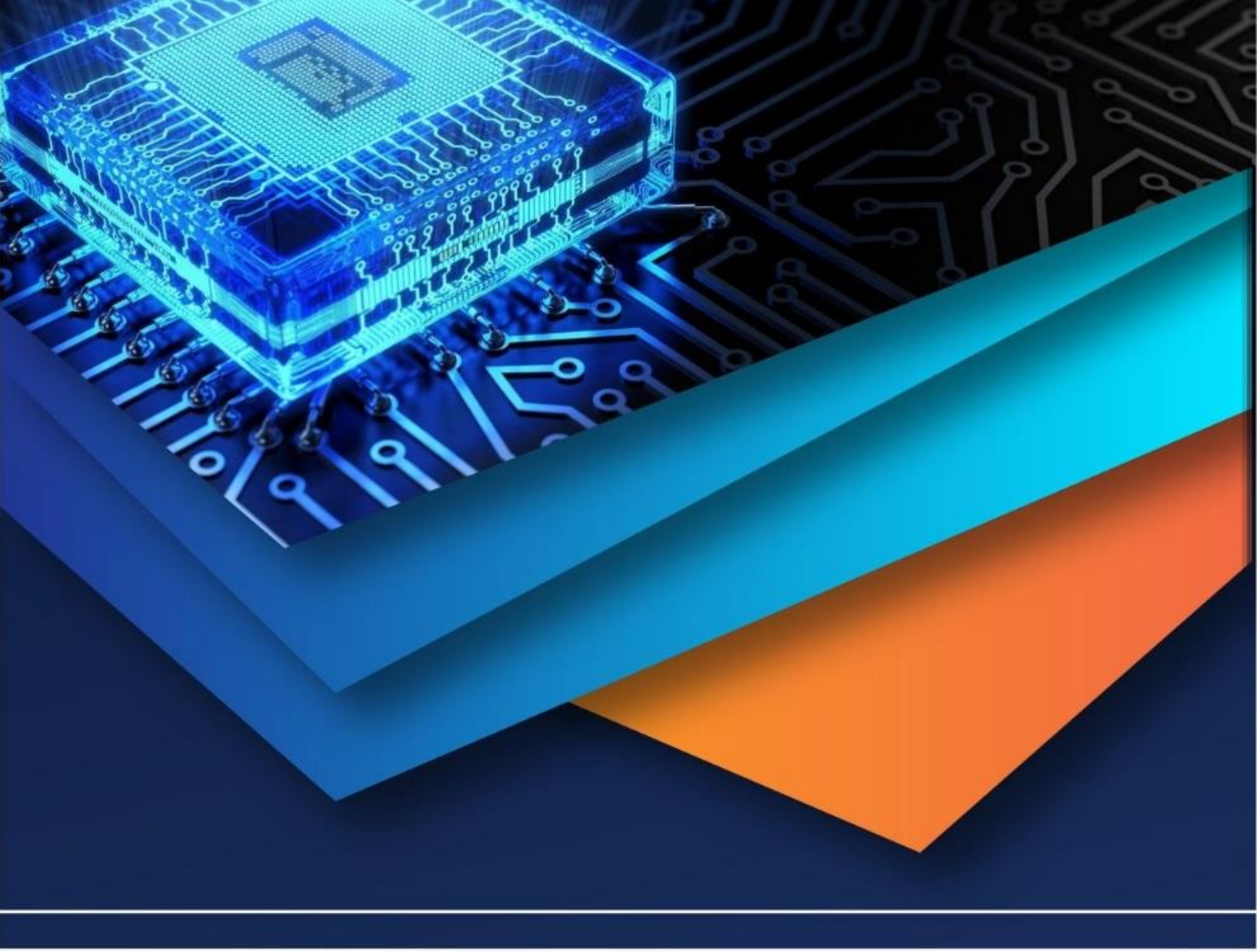
VII. CONCLUSION

In this paper, ANN is applied to find the solution of first order ordinary differential equation. We have implemented Python programming to find the solution. In this approach, we found that ANN was able to perform better than numerical solvers for differential equation. The proposed method has been explored by solving the differential equation to obtain the accuracy. We have discussed possible forms of loss functions in an ANN method for solving differential equation and investigated their efficiency through a number of numerical examples.

REFERENCES

[1] Isaac Elias Lagaris, Aristidis Likas and Dimitrios I. Fotiadis, Artificial Neural Networks for Solving Ordinary and Partial Differential Equations, IEEE Transactios On Neural networks, Vol. 9, No. 5, September 1998, pp. 987 – 1000.
[2] Enzi Shi and Chuanju Xu, A Comparative Investigation of Neural Networks in Solving Differential Equations, Journal of Algorithms and Computational Technology, Vol. 15, 2021, pp. 1 – 15.
[3] Toni Schneidereit and Michael Dreub, Solving Ordinary Differential Equations Using Artificial Neural Networks – A Study on the Solution Variance, Proceedings of ALGORITMY 2020, pp. 21 – 30 .
[4] Liam L.H Lau and Denis Werth, ODEN: A framework to Solve Ordinary Differential Equations using Artificial Neural Networks, arXiv:2005, 14090V2[physics.comp-ph] june 2020, pp. 1 – 10
[5] Susmita Mall and S.Chakraverty, Comparison of Artificial Neural Network Architecture in Solving Ordinary Differential Equations, Advances in Artificial Neural Systems, Hindawi Publishing corporation, Vol. 2013, pp. 1 -12.
[6] Forough Arabshahi, Sameer Singh and Animashree Anandkumar, Towards Solving Differential equations through Neural Programming, ICML workshop Neural Abstract Machines and Program Induction, V2, 2018, pp.1 – 4 .
[7] Lee H and Kang IS Neural Algorithm for Solving Differential equations, Journal of Computational Physics, 1990; Vol. 91, pp. 110 – 131 .
[8] Meade AJ and Fernandez AA, The Numerical Solution of linear Ordinary Differential Equations by feedforward neural networks, Mathematical and Computer Modelling, 1994, Vol. 19, 1994, pp. 1- 25.
[9] Malek A. and Shekari Beidokhti R., Numerical solution for higher order differential equations using a hybrid neural network optimization method, Applied mathematics and Computation, Vol. 183, 2006, pp.260 – 271 .
[10] Michoski C., Milosavljevic M., Oliver T.,et al., Solving Differential Equations using Deep neural networks, Neurocomputing 2020; Vol. 399, pp. 193 – 212.
[11] J.M. Zurada, Introduction to Artificial Neural Network, West Publishing, 1994.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)