Performance Evaluation of Neural Network for Solving Fuzzy Differential Equations Using Root Concatenation

Chitranshu Saxena[1], Dr Sushil Kumar Dhaneliya[2]
[1]*Aptitude Trainer, CGC Chandigarh, Mohali, Chandigarh*
[2]*Assistant Professor, P.K University, Shivpuri, Madhya Prade*sh

***Abstract*—Fuzzy Differential Equations, which incorporate uncertainty through fuzzy logic, are widely used in modeling real-world systems with imprecise data but are challenging to solve using traditional analytical methods. This study introduces a neural network-based solution framework that combines root concatenation with deep learning models to improve accuracy and robustness. The proposed method utilizes root extraction techniques such as Bisection method to generate structured datasets comprising exact, actual, and predicted roots. Convolutional Neural Networks & Long Short-Term Memory architectures are trained using these types of datasets. Contrasted with LSTM's 99.3% accuracy, 98.9% precision, 99.0% recall, and 98.7% F1-score, CNN's experimental findings show that it attains 98.3% accuracy, 97.9% precision, and 98.7% recall. Furthermore, regression metrics such as RMSE (0.0038), MAE (0.0027), and MSE (0.000015) confirm LSTM's superior predictive performance. This approach provides a reliable, data-driven solution for efficiently solving complex fuzzy systems.**

I. INTRODUCTION

Through merging uncertainty and fuzzy sets with ordinary differential equations, a type of mathematical model mentioned as "Fuzzy Differential Equation (FDE)" is being invented. Any place where the data has uncertainty or non-specificity, these equations will be beneficial. The high complexity of fuzzy logic makes FDEs become one of the most difficult tasks. "Neural Networks (NN)" is used to solve FDEs because their capability to cope with the complexity of fuzzy systems has allowed them to gain popularity. NNs are pretty good at dealing with the most difficult problems, especially those that involve the learning of complex data patterns and connections. Fuzzy dynamical systems employ the framework of NNs for the purpose of modeling and describing fuzzy interactions [1].

One of the main advantages of neural networks is that they provide results that are suitable for multiple different cases. On the other hand, multilayer perceptrons can well emulate functions and those which are adaptable to many problems. Fuzzy functions that are not solvable and their derivatives can be calculated through the approach of "Artificial neural networks (ANNs)" and "Bernstein neural network (BNN)" [2].

A hybrid learning machine approach neuro-fuzzy systems also known as "Fuzzy Neural Networks (FNNs)" take the best features of both fuzzy systems and NNs and use them to determine the appropriate values for fuzzy rule and fuzzy set parameters. Pretend for a second that a FNN is comprised of n input units, m hidden units, and s output units [3]. Interconnection weights, biases, and the target vector are all composed of fuzzy numbers, whereas the input vector is comprised of actual values. All of the adjustable parameters, including biases and weights, in a "Fully fuzzy neural networks (FFNN)" are fuzzy numbers. However, more discrete values are used by half fuzzy networks.

Several methods for solving FDEs have been developed thus far due to their widespread application in research and engineering for modeling purposes. In order to solve real-world problems, FDEs typically need to adhere to fuzzy beginning criteria or fuzzy border requirements [4]. To solve the differential equations, a feed-forward neural network is developed and used in the second term. Using just one hidden layer, multilayer perceptrons can approximate any function to an infinite degree of accuracy; this is how this network design comes to be.

Applying fuzzy set theory to the whole set of real numbers, it is found that fuzzy set AAA frequently denotes the most suitable subset. In fuzzy set theory, μA(x) is used to represent the membership function, which specifies the degree of membership of element xxx in set AAA. This function is a continuous mapping on the interval [0, 1]. In this framework, individual items could be allocated special membership degrees in order to depict their extent of association with the fuzzy set [5]. The "A" fuzzy set can be formulated as A {(x, (x)) / x A, (x) [0, 1]}. In cases involving fuzzy numbers, both membership and non-membership should be considered. One major goal in the use of fuzzy set theory in fuzzy logic is to construct a theoretical background for data drawn from linguistic properties.

A fuzzy number is basically a real number that is not exactly zero or one. When speaking about the interval [a, b, c], this is equivalent to real numbers between 0 and 1. Fuzzy sets like this, whose elements are partial, are called 'fuzzy' because of the partialness property of the set elements. The former is represented by a triangle with vertices at two end points, a and c, and one middle point, b. Use the alpha cut to derive the piecewise continuity of the fuzzy number on the interval [0, 1]. The alpha cut is the operation that divides the number into an equal interval in the interval [0, 1] [6].

One of the largest challenges in solving FDEs is dealing with uncertainty effectively and modeling the behavior of fuzzy solutions. To address this, we employ the Root Concatenation Technique, which consists of creating a single input structure from the exact, actual, and predicted roots of the fuzzy solutions. To enhance the quality of approximations and classifications in uncertain circumstances, neural network models can find more sophisticated patterns within the fuzzy system using this hierarchical concatenation of root information [7].

To be more accurate in finding fuzzy solutions, the study tries to utilize root concatenation in a neural network context, implement root extraction techniques like Bisection and Newton-Raphson to regenerate solution info, and, moreover, exploit fuzzy logic to convert traditional differential equations into fuzzy shapes. Also, the paper shows how much better "Convolutional Neural Networks (CNN)", and "Long Short-Term Memory (LSTM)" models are at forecasting fuzzy solutions than the traditional methods.

This study utilizes Neural Networks and the Root Concatenation Technique to perform a comprehensive analysis and assessment with the aim of developing efficient solutions for FDEs. By integrating the ordered root data with neural networks' capabilities in managing complicated fuzzy behaviors, the suggested method seeks to improve the precision, dependability, and usability of FDE solutions. The objective of this endeavor is to make computational structures for dealing with uncertainty stronger and more reliable fuzzy system modeling.

II. LITERATURE REVIEW

Basu et al., (2024) [8] conceived to showcase the proficiency of the trapezoidal-based fuzzy Newton-Raphson technique in dealing with fuzzy algebraic non-linear equations, the article also aimed to prove the effectiveness of the new method by using trapezoidal fuzzy numbers set. By applying this novel procedure, the researchers may solve a great variety of fuzzy non-linear algebraic equations for different fuzzy numbers correctly.

Singh et al., (2024) [9] discussed a way of solving a fuzzy fractional-based differential equation using power series approximation and applying a fuzzy fractional version of Taylor's theorem. In the method, the fuzzy-based differential equation was characterized in terms of Caputo's concept of the fractional second derivative and the generalized Hukuhara difference.

Mallikarjunaiah et al., (2023) [10] suggested a feedforward ANN model based on deep learning to approximate the solutions of "Singly Perturbed Delay Differential Equations (SPDDEs)", with linear interpolation and uniform data points. Performance is measured using a mean square error-based loss function. The results indicate that the adaptive model efficiently copes with different delay and perturbation factors.

Akdemir et al., (2023) [11] suggested a numerical technique to calculate the inverse of a square matrix with fuzzy trapezoidal or triangular elements by solving fuzzy linear equations. The solution intervals are made finer with the bisection method and interval arithmetic. Results demonstrate the method accurately

approximates the fuzzy inverse closely resembling the identity matrix.

Atyia et al., (2023) [12] exhibited a new approach to the solution of n-th order fuzzy randomized ordinary differential equations' linear fuzzy initial value problems by combining numerical integration with the variational iteration method. The success of the approach was guaranteed by the power of the variational iteration method, a validated and reliable approach. The results indicated that the approach was reliable and effective when used on such complicated mathematical problems.

Akram et al., (2022) [13] investigated an analytical solution to solve a fuzzy system of linear fractional differential equations with Caputo's derivatives of orders in two dimensions. Two methods of resolution having their base in the concept of constraints and a class of extended H-differentiable functions were proposed. Computer simulation of the transport problem verified the correctness of theoretical deductions.

Wang et al., (2021) [14] proposed the "Multilayer Neural Network for Partial Differential Equations (MLNPDE)" as the one to be used for dealing with the elliptic PDEs via the deep learning paradigm. Through the model, the solution is untangled to two expressions such that the first is the one that satisfies the boundary condition. The approach significantly facilitated the standardization of boundary conditions for 2-D elliptic PDE problems.

Abbas et al., (2021) [15] proposed a method to solve interval FDEs of n-th order using the backstepping method. Initially, the n-th-order FDE was converted into a first-order FDE system. Subsequently, the system was solved with the backstepping method in combination with Lyapunov's direct method to establish asymptotic stability.

Semenov et al., (2021) [16] suggested the fuzzy interval bisection method for solving nonlinear fuzzy equations applicable to cyber-physical systems. Using all available fuzzy input information about the equation's coefficients, the proposed method makes it easier to estimate the membership function of the root of the equation.

Gumah et al., (2020) [17] presented a computational method based on the "Reproducing Kernel Hilbert Space (RKHS)" technique to solve "Hybrid Fuzzy Differential Equations (HFDEs)". In order to get approximate-analytical solutions, the Gram-Schmidt process was used to develop a system of orthogonal functions in space W. α-cut representations were used to establish convergence. Numerical examples illustrated the high precision and efficiency of the RKHS technique.

Yang et al., (2020) [18] proposed "Block Neural Network (BNN)" method to solve a variety of differential equations, where training was done using an "Improved Extreme Learning Machine (IELM)" technique. The BNN built approximation functions and their derivatives effectively. The practicality and precision of the approach were confirmed by numerical trials. The newly suggested BNN with IELM outperformed state-of-the-art algorithms while using a smaller number of hidden neurons.

III. RESEARCH METHODOLOGY

Figure 1 depicts the study process that integrates neural networks, numerical methods, and fuzzy logic to answer FDEs.

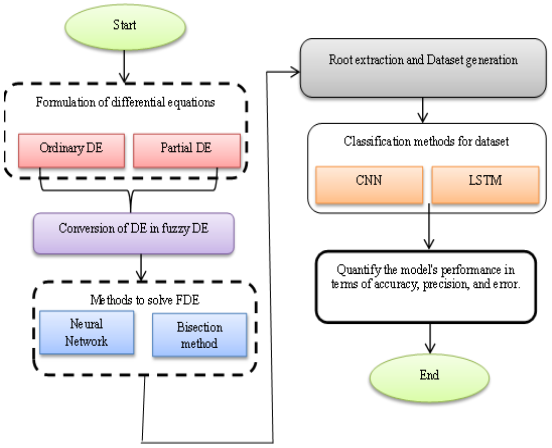

Figure 1: Proposed Methodology

3.1 Formulation of Differential Equation

A differential equation is utilized to ascertain the functional connection among functions and their derivatives. A dependent variable delineates alterations in relation to the independent variable as per its definition [19] . Both ordinary differential equations (ODEs) and partial differential equations (PDEs) are fundamental forms of differential equations.

- Ordinary Differential Equations:

Non-linear equations (NLEs) include derivatives of functions with a single variable. An example is the

Legendre equation, used in quantum physics to describe wave effects. The general form is $\frac{dy}{dx} = f(x, y)$, and higher-order ODEs involve second or higher derivatives, such as

$$\frac{d^2y}{dx^2} + p(x)\frac{dy}{dx} + q(x)y = g(x)$$

ODEs are used in modelling vibrations, electrical circuits, and population dynamics.

- Partial Differential Equations:

PDEs include functions of the partial derivatives of several independent variables, e.g., $\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial v} = 0$. These equations describe physical phenomena such as heat and wave propagation. The solution is a function u(x, t) that satisfies the equation under given conditions.

3.2 Conversion of DE into Fuzzy Differential Equation

Exceeding FDE is a crucial expansion of conventional DE that use fuzzy logic to address uncertainty in computational modeling [20] . Utilizing fuzzy sets, such formulas can manage missing data and ambiguous information, yielding answers that represent possible variability instead of a singular deterministic result. Transformation of the DE into an FDE:

$$\frac{d\hat{y}}{dx} = F(x, \hat{y}, \Delta) \qquad (1)$$

Where:

- $\frac{d\hat{y}}{dx}$ denotes the rate of change of a fuzzy variable $\hat{y}$ in relation to x.
- $F(x, \hat{y}, \Delta)$ is a fuzzy variable that integrates uncertainty denoted by Δ.

A fuzzy number-valued functional F defined on $[a, b]$ is considered (1)&(2)-differentiable of order k (where $k \in \mathbb{N}$) on the interval $[a,b]$ if $F(s)$ is (1)-distinct for all $s = 1, \ldots, k$. Let y be the answer to the FDE of order s. If y is distinguishable, then "$(t) = ((t, r), \overline{y}(t,))$". If y is twice different, then $y(t) = (y(t,r), \overline{y}\ (t,r))$ when s is even, and $y(t) = (\overline{y}\ (t,r), y(t,r))$ when s is odd. In the subsequent section, we compute $\overline{y}(t, r)$ and $(t,)$ utilizing the differential transform technique.

3.3 Methods to Solve FDE

3.3.1 Neural Networks

When dealing with system uncertainties, a neural network is a useful tool for approximating FDE solutions. In this method, the fuzzy solution $\hat{y}(x)$is modeled using a feedforward neural network. The network approximates the fuzzy function and its derivatives, treating the uncertainty through fuzzy parameters or α-cut representations. The learning process involves minimizing the residual error of the FDE across selected training points.

Given an FDE:

$$\frac{d\hat{y}}{dx} = F(x, \hat{y}(x), \Delta),$$

the neural network approximates $\hat{y}(x)$ as $\widehat{y_{NN}}(x,w)$, where w are the network weights. The residual error is defined as:

$$R(x) = \frac{d\hat{y}_{NN}}{dx} - F(x, \hat{y}_{NN}(x, w), \Delta)$$

The training aims to minimize the loss function:

$$\text{Loss}(w) = \frac{1}{N}\sum_{i=1}^{N} ||R(x_i)||^2$$

Neural networks provide results with excellent generalization, including differentiability.

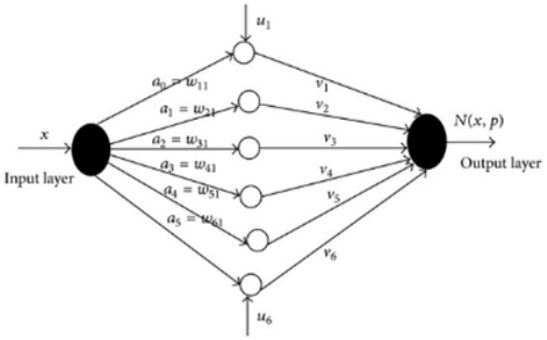

Figure 3: NN Architecture for FDE

Example 2: Consider the following fuzzy IVP:

$$\begin{cases} x' = 3t^2x(t), & t \in [0,1 \\ x(0) = \left(0.5\sqrt{r} + 0.2\sqrt{1-r} + 0.5\right) \end{cases} \qquad (2)$$

Example 3: Take into account the following fuzzy IVP:

$$\begin{cases} x'(t) = -x(t) + t + 1, & t \in [0,1] \\ x(0) = (0.96 + 0.04r, 1.01 - 0.01r) \end{cases} \qquad (3)$$

3.3.2 Bisection Method

Finding the solutions to the equation f(x) = 0 is possible using the Bisection Method if f(x) is continuous on the interval [a, b]. In order for an interval to have a root, the signs of f (a) and f (b) must be different. The method iteratively halves the interval, selecting the half that satisfies the sign

condition. The formula for the root approximation is $x_k = \frac{a+b}{2}$ and the number of iterations needed is given by $| (b-a)/2^k | \leq \epsilon$.

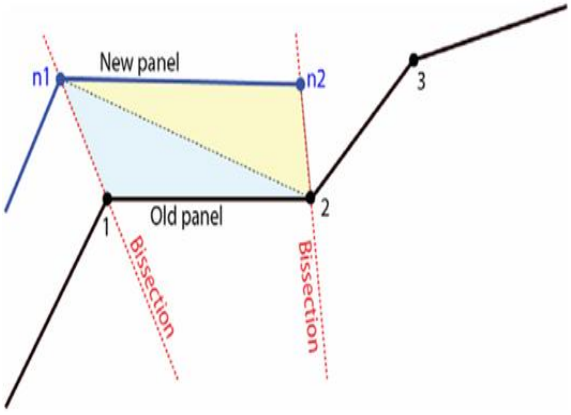

Figure 4: Bisection method [11]

Example 4: Preliminary estimations are

$\mu_1^{(0)} = \mu(a_1^{(0)}, b_1^{(0)}, c_1^{(0)})$ (4)

And $\mu_2^{(0)} = \mu(a_2^{(0)}, b_2^{(0)}, c_2^{(0)}$ (5)

tolerance; maximal iteration count n.

Example 5: Find the solution to a fuzzy non-linear equation

$\mu_x^3 + \mu_x^2 - 1 = 0.$ (6)

3.4 Root extraction and dataset generation

A rigorous systematic process for constructing datasets and obtaining roots adds fundamental value to the evaluation of numerical and intelligent techniques that solve FDEs. The FDE's root solutions provide system points which are significant, demarcating states of equilibrium or critical points of thresholds upon which scientists are able to infer outcomes of system behavior from results of analysis. Evaluation progresses toward constructing a dataset from roots that will be employed for further analysis purposes.

The data set consists of different root-related subcategories, which are essential elements for performance assessment of the deployed methods. The dataset represented by it includes a specific set of components:

- Exact Roots: Bisection generate roots that are taken to be exact roots.
- Actual Roots: These are the analytically obtained or known roots of the differential equation if available.
- Predicted Roots: These roots are approximately using smart models like neural networks that are trained to estimate the solutions of FDEs.

The development of this dataset enables a systematic and quantitative comparison of different solution approaches. In addition, the dataset will serve as a critical input to train and test neural network models in subsequent phases of the research.

3.5 Classification Techniques for dataset

3.5.1 Conventional Neural Network

Deep learning models called CNNs can be used to extract spatial properties from grid-like input, such photographs. CNNs use weights, biases, and nonlinear activation functions to model complex patterns [21]. For solving Fuzzy Differential Equations (FDEs), CNNs classify relationships between exact, actual, and predicted roots, enabling fuzzy system approximation. The mathematical model for a CNN layer is:

$$z = W \cdot x + b, \qquad y = \sigma * (z),$$

The activation function is denoted by σ (·), the input is x, the bias is b, and W stands for weights. As a loss function, the mean squared error is typically used:

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

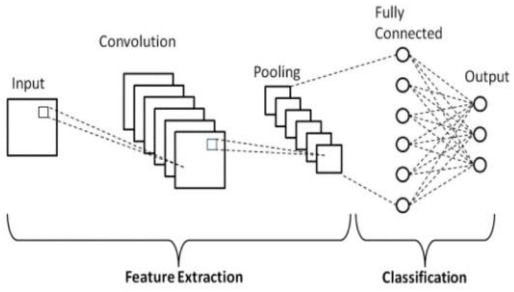

Figure 6: CNN model architecture

Long Short-Term Memory

LSTM in a RNN manages data sequences of differing durations, represented as $x = (x_1, x_2 \ldots x_n)$, by integrating fresh information into a singular memory unit. The retention of fresh knowledge, the forgetting of previous information, and the accessibility of current information to the user are regulated by gates. The equations are employed to update the memory (c_t) and the state that is hidden (h_t) at each time step (t) [22]:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W.[h_{t-1}, x_t] \quad (7)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \quad (8)$$

$$h_t = O_t \odot \tanh(c_t) \quad (9)$$

3.6 Performance Metrics

This section evaluates the effectiveness of the suggested deep learning model. The proposed technique employs a model to address FDE. The Gate activations are denoted by i, f, and o. The LSTM neural network employs additive memory updates, distinguishing it from the conventional recurrent neural network by separating the cell memory from the hidden state, which interacts with the external environment during prediction.

model's efficacy was evaluated using the essential variables: $A_{accuracy}$, $F1_{score}$, $P_{precision}$, and R_{recall} are the relevant variables:

$$\text{Accuracy} = \frac{TN+TP}{TP+FN+FP+TN} \quad (10)$$

Where, TN = True Negative, (TP) = True Positive, (FP) = False Positive and (FN) = False Negative.

$$\text{Precision} = \frac{NA}{NA+FP} \quad (11)$$

$$\text{Recall} = \frac{NA}{FN+NA} \quad (12)$$

$$\text{F1 score} = \frac{2\times precision \times recall}{recall+precision} \quad (13)$$

$$\text{MSE} = \frac{1}{n} = \sum_{i=1}^{n}(x_i - \hat{x_i})^2 \quad (14)$$

The variables n and xi represent the dataset's total number of features and the actual yield for the i^{th} sample, and $\hat{x_i}$ represents the predicted yield.

IV. RESULT AND ANALYSIS

4.1 Solution of FDE using Neural Network

Neural networks offer a flexible, data-driven approach to solving FDEs under uncertainty. Using deep learning models like CNNs and LSTMs, they handle nonlinear, ambiguous problems more effectively than traditional methods. This is useful in modeling uncertain population dynamics, fuzzy control in robotics, and environmental predictions with vague input data.

Example 2: Consider the following fuzzy IVP:

$$\begin{cases} x' = 3t^2 x(t), & t \in [0,1 \\ x(0) = \left(0.5\sqrt{r} + 0.2\sqrt{1-r} + 0.5\right) \end{cases} \quad (15)$$

The exact solution for t=1 is: $x(1,r) = 0.5\sqrt{r}e, \left(0.2\sqrt{1-r} + 0.5\right)e)$. Fig. 4.4 shows the exact solution and the approximated solution for t=1.

Figs. 12&15 show the convergence behaviours for computed values of the weight parameters $\underline{w}_{i1}, \underline{w}_{i2}$; bias $\underline{u}$ and the weights of output layer $\underline{v}$ for different numbers of iterations.

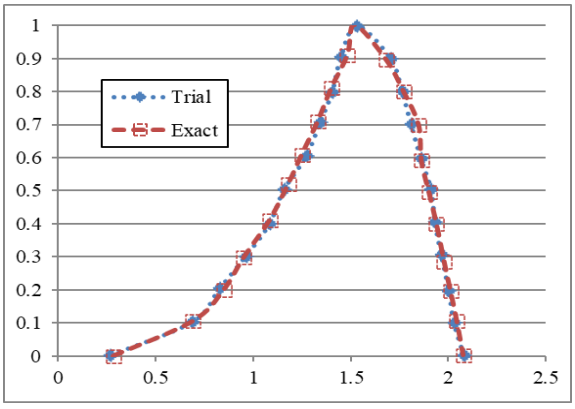

Figure 4.1: Comparison between the trial solution and exact solution for Example 5.2.

The graph compares the trial (predicted) and exact solutions for Example 2 over the domain of the independent variable. Both curves closely overlap, peaking near 0.98 at x=1.5 and symmetrically decreasing to 0 by x=2, demonstrating excellent agreement between the two solutions.

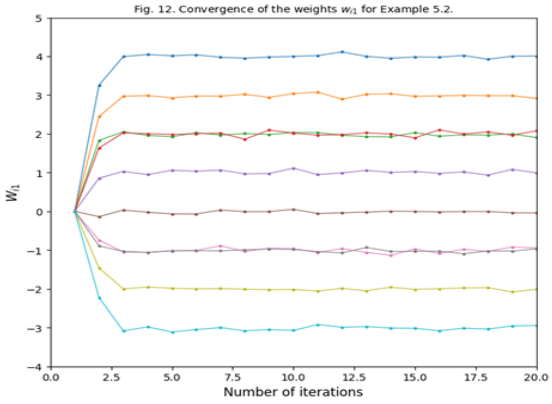

Figure 4.2: Convergence of the weights wi1 for example 5.2

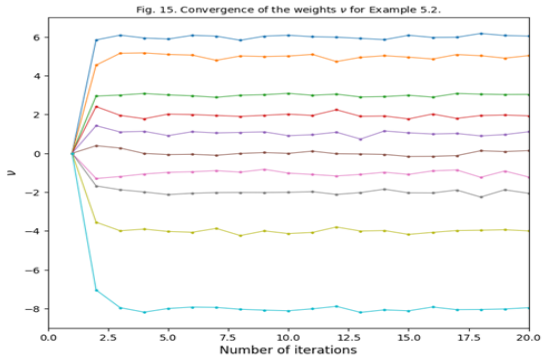

Figure 4.3: Convergence of the weight v for 5.2

Figures 4.5 and 4.6 show the trajectories of w_{i1} and v for Example 2 across iterations. Both weights initially fluctuate but stabilize quickly—w_{i1} between −3.5 and 4.5, and v after a drop near −8.

Example 3: Take into account the following fuzzy IVP:

$$\begin{cases} x'(t) = -x(t) + t + 1, & t \in [0,1] \\ x(0) = (0.96 + 0.04r, 1.01 - 0.01r) \end{cases} \quad (16)$$

where $r \in [0,1]$. We can write the parametric form of the problem as follows

$$\begin{cases} \underline{x}'(t) = F[t, \underline{x}(t,r), \overline{x}(t,r)], & \underline{x}(0) = 0.96 + 0.04r \\ \overline{x}'(t) = F[t, \overline{x}(t,r), \underline{x}(t,r)], & \overline{x}(0) = 1.01 - 0.01r \end{cases}$$

In which $t \in [0,1]$and F,G satisfy (9), the neural nform of the trial solutions is as follows:

$$\begin{cases} \underline{x}(t,r) = (0.96 + 0.04r) + t\underline{N}(t,r,\underline{p}) \\ \overline{x}(t,r) = (1.01 - 0.01r) + t\overline{N}(t,r,\overline{p}) \end{cases} \quad (17)$$

regarding the interval [0,1]. The initial criteria are satisfied by the trial solutions in equation (23). We can now use the quasi-Newton BFGS approach to solve the unconstrained optimization problem by substituting (17) with (22) and (23). For t=0.1, both the exact and approximate solutions are displayed in Fig.16.

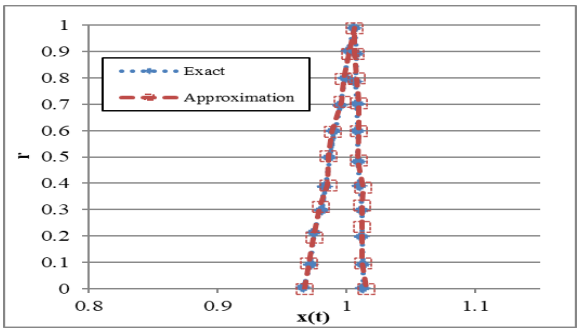

Figure 4.4 : Solution to Example 5.3, both exactly and as close as possible.

Figure 4.7 compares the approximate and exact solutions $x_a(0.1,r)$ and $x_T(0.1,r)$ for $r \in [0,1]$, showing minimal error and strong agreement. The results confirm the method's stability and precision across varying r values.

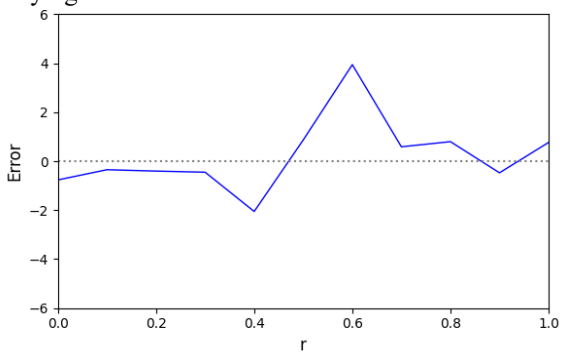

Figure 4.5: $\underline{E}(1,r)$ for example 3

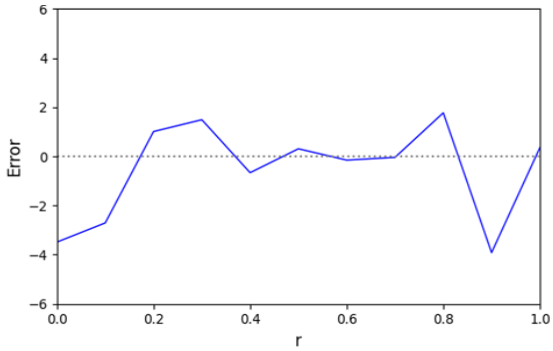

Figure 4.6: $\overline{E}(1,r)$ for example 3

Figures 4.8 and 4.9 display the error E (0.1, r) for Example 5.3 over $r \in [0, 1]$, with values fluctuating

between −5 and +4. The moderate oscillations indicate minor local deviations, confirming the neural network's efficient and accurate approximation of the FDE solution.

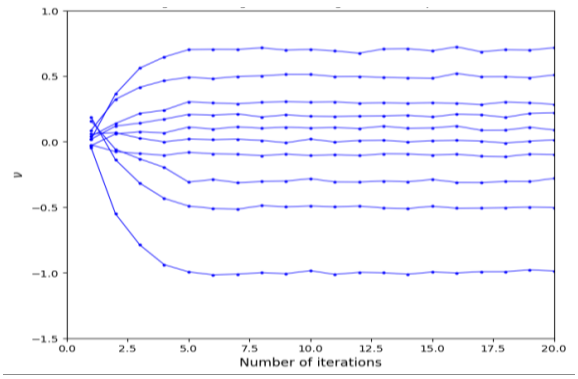

Figure 4.7: Convergence of weight ν for example 3

Graph 4.10 shows the behavior of weight values ν over 20 iterations, starting near zero and stabilizing between −1.0 and 0.75 after 5 –10 iterations. Minimal variation indicates successful convergence of the learning process.

4.2 Solution of FDE using Bisection Method

The Bisection Method solves FDEs by iteratively having an interval containing the root of approximate solutions. It refines the root value at each step until the desired accuracy is reached, effectively handling uncertainties in fuzzy systems. This method is illustrated with examples demonstrating its accuracy in solving complex FDEs.

Example 4: Preliminary estimations are

$\mu_1^{(0)} = \mu(a_1^{(0)}, b_1^{(0)}, c_1^{(0)})$ (18)

And $\mu_2^{(0)} = \mu(a_2^{(0)}, b_2^{(0)}, c_2^{(0)})$ (19)

tolerance; maximal iteration count n.

Solution:

$\mu_1^{(0)} = \mu(a_1^{(0)}, b_1^{(0)}, c_1^{(0)})$ or message of failure.

Step 1: Set i=1 FA $= \bar{f}(r(b_1^{(0)}))$

Step 2: While t less than equal n do step 3-6

Step 3: Set $\mu_x^{(n)} = \frac{\mu_1^{(n-1)} + \mu_2^{(n-1)}}{2}$ (Compute $\mu_x^{(0)}$)

FP $= \bar{f}(r(b_n^{(n)})$

Step 4: if FP=0 or $\frac{|\mu(b^n) - \mu(b^{n-1})|}{2}$ is less then TOL then output: $\mu_x^{(n)}$ (the procedure is successful)

Step 5: Set i=i+1,Step 6: if FA.FP>0 then setting $\mu_1^{(0)} = \mu_x^{(n-1)}$ update $\mu_x^{(0)}$

FA=FP

Else $\mu_2^{(0)} = \mu_x^{(n-1)}$.

Example 5: Solve a fuzzy non-linear equation

$\mu_x^3 + \mu_x^2 - 1 = 0.$ (20)

Solution: Suppose that $\bar{f}(\mu_x) = \mu_x^3 + \mu_x^2 - 1$

If we use the interval $[\mu_1^{(0)}.\mu_2^{(0)}]$=[μ (0,0.5,1) μ (0.5,1,1.5)], then equation (1.1) generates the sequence $\{\mu_x^{(n)}\}$ of approximate solutions as follows:

Iteration 1: $\bar{f}(\mu_1^{(0)})=\mu^3(0,0.5,1) + \mu^2(0,0.5,1) - 1$

$=\mu(-1, -0.375, 1)<0$

$\bar{f}(\mu_2^{(0)})=\mu^3(0.5,1,1.5) + \mu^2(0.5,1,1.5) - 1$

$=\mu(-0.625,1,4.625)>0$

Hence root lies among $[\mu_1^{(0)}, \mu_2^{(0)}]$.

Then, we get $\mu_3^{(1)} = \frac{\mu_1^{(0)} + \mu_2^{(0)}}{2} = \mu(0.25,0.75,1.25)$,

Where $\bar{f}(\mu_3^{(0)}) < 0$.So, root lies between $[\mu_2^{(0)}, \mu_3^{(1)}]$.In the same way we obtain the approximate solutions of $\bar{f}(\mu_x) = \mu_x^3 + \mu_x^2 - 1 = 0$. From the solution it is clear that $\mu_{17}^{(15)} = \mu(0.254867549,0.754867549,1.254867549)$ is the desired approximate root of the given equation. The optimum solution of "Fuzzy non-linear equation (FNLE)" is shown in Fig.1.3.

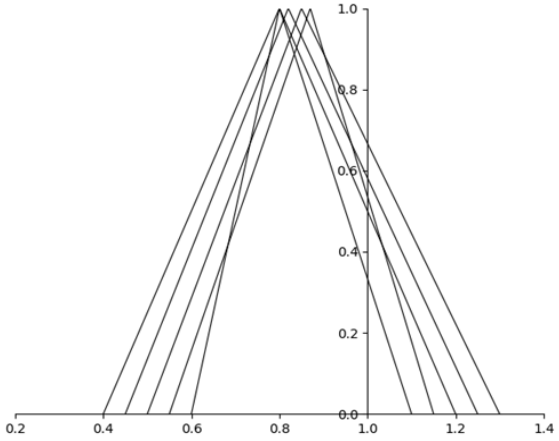

Figure 4.8: The best solution to a fuzzy non-linear problem visually representation.

Root extraction and dataset generation

To ensure accurate performance evaluation, the dataset includes three types of roots: Exact Roots, Actual Roots, and Predicted Roots, generated by machine learning models. This structure allows

consistent comparison across various parameter instances and helps validate model accuracy.

Table 4.1: Comparison of exact, analytical, and predicted roots for different values of r for dataset generation.

r	Exact Root (Bisection/NR)	Actual Root (Analytical)	Predicted Root (NN)
0.0	2.03872	2.03870	2.03868
0.1	2.10664	2.10660	2.10655
0.2	2.17461	2.17460	2.17458
0.3	2.24260	2.24259	2.24255
0.4	2.31053	2.31052	2.31050
0.5	2.37852	2.37850	2.37847
0.6	2.44648	2.44645	2.44642
0.7	2.51439	2.51437	2.51434
0.8	2.58246	2.58245	2.58242
0.9	2.65032	2.65030	2.65027
1.0	2.71828	2.71828	2.71825

Table 4.4 compares Exact, Actual, and Predicted Roots for various r values in [0, 1], highlighting the accuracy of numerical and neural network approaches. Exact Roots use numerical methods, Actual Roots are ground truth, and Predicted Roots come from trained neural networks. Small differences among them show the effectiveness of neural networks in solving FDE.

4.4. Experimental Results

4.6.1 CNN model

Shown in Figure 4.13 is the CNN performance after 50 epochs. Training and validation accuracy rapidly increase and stabilize at 1.0 in subplot (a), whereas loss decreases with small spikes in validation in subplot (b), suggesting good training, little overfitting, and good generalization. Figure 4.14's confusion matrix provides a detailed analysis of the CNN model's performance in class distinction, with respect to true positives, false positives, true negatives, and false negatives.

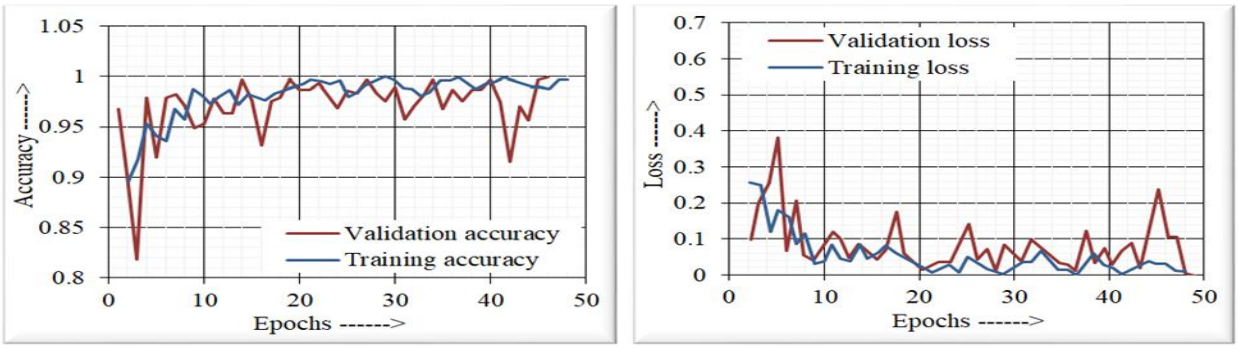

Figure 4.9: CNN a) Training/Validation accuracy b) Training/Validation loss

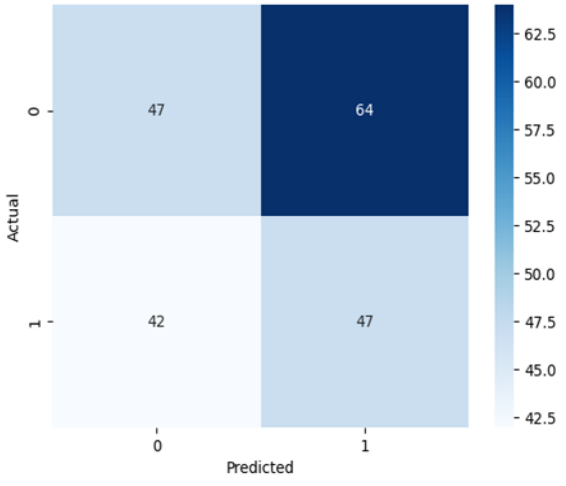

Figure 4.10: CNN model confusion metrics.

4.6.2 LSTM Model

The training and validation performance of an LSTM model across 50 epochs is shown in figure 4.16. In the accuracy figure 4.16 (a), both training and validation accuracy rapidly increase, surpassing 0.95 within the first 30 epochs and stabilizing around 0.99 by the end of training, indicating excellent classification performance. In the loss figure (b), both training and validation loss decrease sharply from initial values around 0.3–0.4 to values below 0.05 after 50 epochs. This consistent performance across metrics demonstrates that the LSTM model generalizes well, achieving high reliability without significant overfitting or underfitting. Figure show the confusion metrics of LSTM model.

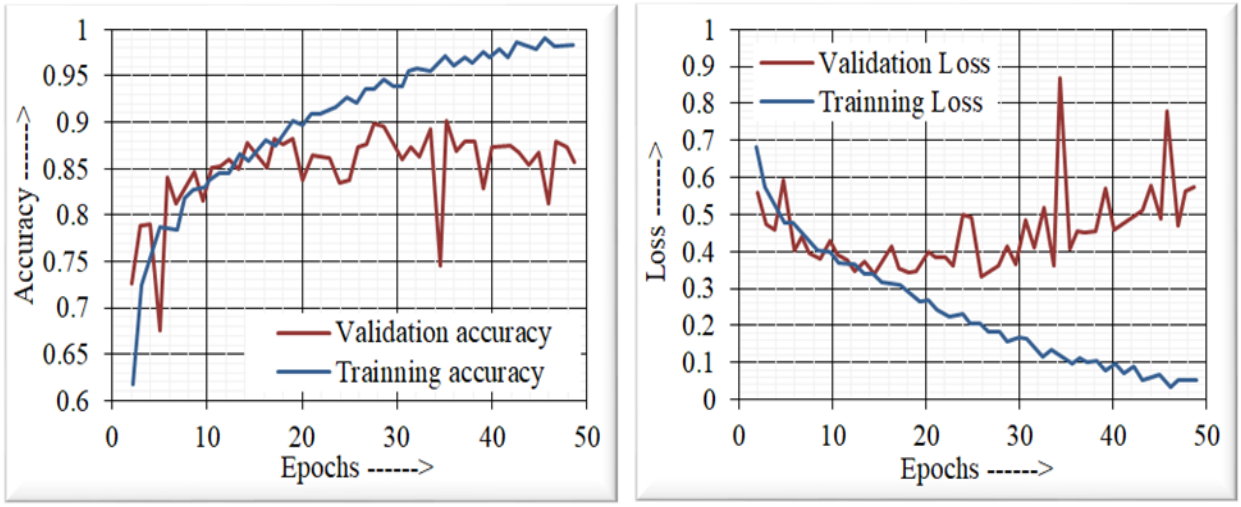

Figure 4.11: LSTM a) Accuracy during Training and Validation b) Loss during Training/Validation

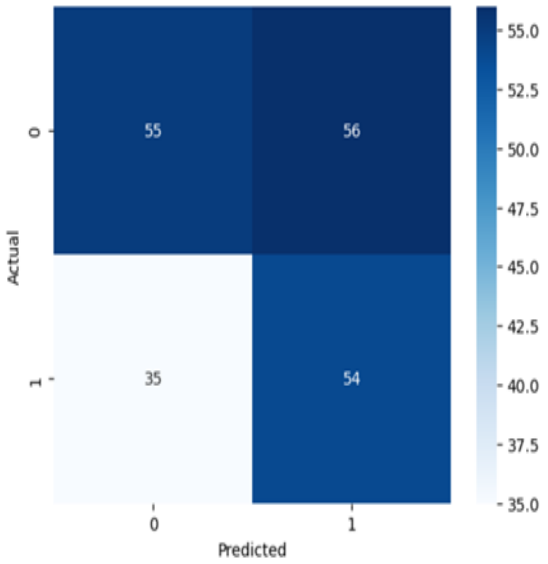

Figure 4.12: LSTM model confusion matrix

4.5 Comparative Analysis

The comparative analysis section provides a thorough comparison of the performance of the CNN and LSTM models on different metrics.

Table 4.2: Comparison of proposed CNN and LSTM model

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
CNN	98.3	97.9	98.7	98
LSTM	99.3	98.9	99	98.7

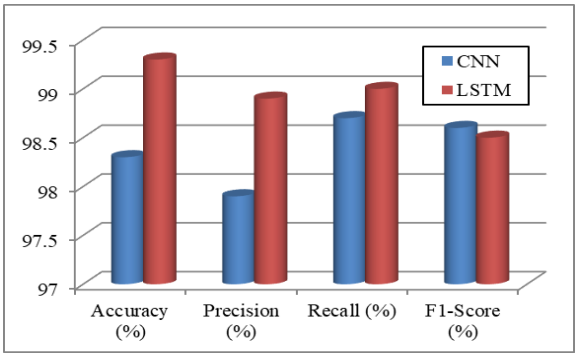

Figure 4.13: CNN model v/s LSTM for various performance metrics

Figure 4.19 compare CNN and LSTM performance on FDE prediction. LSTM outperforms CNN with 99.3% accuracy versus 98.3%. It also achieves higher precision (98.9%), recall (99%), and F1-score (98.7%), indicating superior overall performance.

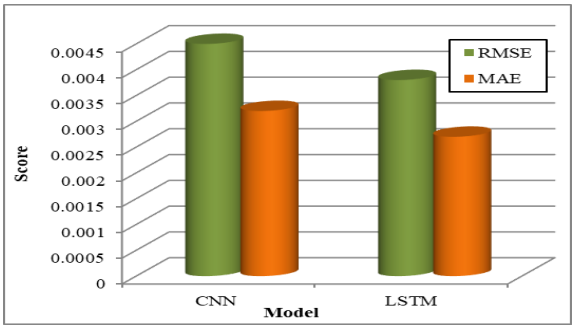

Figure 4.14: Graphical representation of RMSE, MAE, and MSE values

Table 4.8 and Figure 4.20 compare CNN and LSTM using regression metrics RMSE, MAE, and MSE. LSTM outperforms CNN with lower values across all metrics: RMSE (0.0038 vs. 0.0045), MAE (0.0027 vs. 0.0032), and MSE (0.000015 vs. 0.00002), indicating more accurate and reliable predictions.

V. CONCLUSION

The neural network performance evaluation of root concatenation to solve FDEs proves their excellent ability to tackle uncertainty and nonlinearity inherent in fuzzy systems. Both the training and validation phases show that CNN and LSTM models are competent at estimating solutions for FDEs, thanks to their high accuracy and low loss. Contrasted with LSTM's 99.3% accuracy, 98.9% precision, 99.0% recall, and 98.7% F1-score, CNN's experimental findings show that it attains 98.3% accuracy, 97.9% precision, and 98.7% recall. Furthermore, regression metrics such as RMSE (0.0038), MAE (0.0027), and MSE (0.000015) confirm LSTM's superior predictive performance. The close conformity of predicted, exact, and actual roots further testifies to the robustness of neural networks. In summary, the root concatenation method coupled with deep learning presents an effective, data-driven alternative to classical numerical methods for solving intricate fuzzy differential equations.

REFERENCES

[1] Sahni, Manoj, Meghna Parikh, and Ritu Sahni. "Sumudu transform for solving ordinary differential equation in a fuzzy environment." Journal of Interdisciplinary Mathematics 24, no. 6 (2021): 1565-1577.

[2] Suhhiem, Mazin H. "Artificial Neural network for solving fuzzy differential equations under generalized H-derivation." International Journal 5, no. 1 (2017): 1-9.

[3] Abbasbandy, Saeid, Mahmood Otadi, and Maryam Mosleh. "Numerical solution of a system of fuzzy polynomials by fuzzy neural network." Information Sciences 178, no. 8 (2008): 1948-1960.

[4] Mazandarani, Mehran, and Li Xiu. "A review on fuzzy differential equations." IEEE access 9 (2021): 62195-62211.

[5] Stefanini, Luciano, Laerte Sorini, and Maria Letizia Guerra. "Parametric representation of fuzzy numbers and application to fuzzy calculus." Fuzzy Sets and Systems 157, no. 18 (2006): 2423-2455.

[6] Mosleh, Maryam, and Mahmood Otadi. "Simulation and evaluation of fuzzy differential equations by fuzzy neural network." Applied Soft Computing 12, no. 9 (2012): 2817-2827.

[7] Nieto, J. J., Alireza Khastan, and K. Ivaz. "Numerical solution of fuzzy differential equations under generalized differentiability." Nonlinear Analysis: Hybrid Systems 3, no. 4 (2009): 700-707.

[8] Basu, Sanjoy, and G. S. Mahapatra. "Numerical solution of fuzzy non-linear equations under generalized trapezoidal fuzziness by Newton–Raphson method." Franklin Open 9 (2024): 100168.

[9] Singh, Payal, Kamal Hossain Gazi, Mostafijur Rahaman, Soheil Salahshour, and Sankar Prasad Mondal. "A fuzzy fractional power series approximation and Taylor expansion for solving fuzzy fractional differential equation." Decision Analytics Journal 10 (2024): 100402.

[10] Mallikarjunaiah, S. M. "A deep learning feed-forward neural network framework for the solutions to singularly perturbed delay differential equations." Applied Soft Computing 148 (2023): 110863.

[11] Akdemir, Hande Günay. "Approximate Fuzzy Inverse Matrix Calculation Method using Scenario-based Inverses and Bisection." Fundamental Journal of Mathematics and Applications 6, no. 1 (2023): 42-50.

[12] Atyia, Osama M., Fadhel S. Fadhel, and Mizal H. Alobaidi. "Using Variational Iteration Method for Solving Linear Fuzzy Random Ordinary Differential Equations." Mathematical Modelling of Engineering Problems 10, no. 4 (2023).

[13] Akram, Muhammad, Tayyaba Ihsan, and Tofigh Allahviranloo. "Solving Pythagorean fuzzy fractional differential equations using Laplace transform." Granular Computing 8, no. 3 (2023): 551-575.

[14] Wang, Zheng, Alphonse Houssou Hounye, Jiaoju Wang, Cong Cao, and Muzhou Hou. "Adaptive multilayer neural network for solving

elliptic partial differential equations with different boundary conditions." Digital Signal Processing 118 (2021): 103223.

[15] Abbas, Muna S., and Fadhel S. Fadhel. "Solution of n-th order interval fuzzy differential IAL equations using the backstepping method." International Journal of Nonlinear Analysis and Applications 12, no. Special Issue (2021): 1965-1985.

[16] Semenov, Konstantin, and Anastasiia Tselishcheva. "Solving Fuzzy Equations Based on Fuzzy Interval Bisection Method for Intelligent Data Processing in Cyber-Physical Systems." In International Conference System Analysis In Engineering And Control, pp. 300-310. Cham: Springer International Publishing, 2021.

[17] Gumah, G., M. F. M. Naser, M. Al-Smadi, S. K. Q. Al-Omari, and Dumitru Baleanu. "Numerical solutions of hybrid fuzzy differential equations in a Hilbert space." Applied Numerical Mathematics 151 (2020): 402-412.

[18] Yang, Yunlei, Muzhou Hou, Jianshu Luo, and Zhongchu Tian. "Numerical solution of several kinds of differential equations using block neural network method with improved extreme learning machine algorithm." Journal of Intelligent & Fuzzy Systems 38, no. 3 (2020): 3445-3461.

[19] Denis, Byakatonda. "An overview of numerical and analytical methods for solving ordinary differential equations." arXiv preprint arXiv:2012.07558 (2020).

[20] S. L. Chang and L. A. Zadeh, On fuzzy mapping and control, IEEE Transactions on Systems, Man and Cybernetics, 2 (1972) 30-34.

[21] Gonzalez, Claudia I., and Cesar Torres. "Fuzzy convolutional neural network model applied to classification problems." Journal of Intelligent & Fuzzy Systems (2024): JIFS-219369.

[22] Cheng, Jianpeng, Li Dong, and Mirella Lapata. "Long short-term memory-networks for machine reading." arXiv preprint arXiv:1601.06733 (2016).