RESEARCH ARTICLE **OPEN ACCESS**

Manuscript received October 5, 2024; Revised December 1, 2024; Accepted December 12, 2024; date of publication February 20, 2025
Digital Object Identifier (**DOI**): https://doi.org/10.35882/jeeemi.v7i2.492

How to cite: Tirimula Rao Benala, Anupama Kaushik and Satchidananda Dehuri, “Swarm Intelligence-Based Functional Link Fuzzy Neural Estimator for Software Development Effort Estimation”, Journal of Electronics, Electromedical Engineering, and Medical Informatics, vol. 7, no. 2, pp. 253-269, April 2025.

Swarm Intelligence-Based Functional Link Fuzzy Neural Estimator for Software Development Effort Estimation

Tirimula Rao Benala[1], Anupama Kaushik[2] and Satchidananda Dehuri[3]

[1] Department of Information Technology, JNTU-GV College of Engineering Vizianagaram (Autonomous), Jawaharlal Nehru Technological University Gurajada Vizianagaram Dwarapudi Vizianagaram, Andhra Pradesh-535003, India. Email: btirimula.it@jntugvcev.edu.in
[2] Department of IT, Maharaja Surajmal Institute of Technology, Affiliated to GGSIP University, New Delhi, India. Email: anupama@msit.in
[3] Department of Information and Communication Technology, Fakir Mohan University, Vyasa Vihar, Balasore-756019, Odisha, India. Email: satchi.lapa@gmail.com

Corresponding Author: Tirimula Rao Benala (Email: btirimula.it@jntugvcev.edu.in)

ABSTRACT Accurate Software Development Effort Estimation (SDEE) is pivotal for effective project management, significantly impacting resource allocation and the overall success of software projects. This paper introduces the Swarm Intelligence-Based Functional Link Fuzzy Neural Estimator (SFNE), a novel computational intelligence model designed to enhance estimation accuracy by integrating multiple advanced methodologies. The SFNE framework employs the QUICK algorithm for dataset optimization, effectively minimizing noise and redundancy. A Functional Link Artificial Neural Network (FLANN) captures complex nonlinear relationships within the data, while Interval Type-2 Fuzzy Logic Systems (IT2FLS) address inherent data uncertainties. Additionally, Particle Swarm Optimization (PSO) is applied to fine-tune model parameters, improving prediction precision. Empirical evaluations were conducted using six benchmark datasets from the PROMISE repository. The results demonstrate that the SFNE model significantly outperforms existing models across key metrics, including Mean Magnitude of Relative Error (MMRE), Median Magnitude of Relative Error (MdMRE), and Prediction at 0.25 (PRED(0.25)). Notably, SFNE achieved a predictive accuracy of 99.983% on the DesharnaisL3 dataset and an MMRE of 2.87×10^{-5} on the DesharnaisL1 dataset. These findings underscore the robustness and adaptability of SFNE in addressing the limitations of traditional SDEE methods, particularly in managing data scarcity and uncertainty. The proposed SFNE model establishes a new benchmark for SDEE accuracy and demonstrates substantial potential for practical application in real-world software engineering projects. Future research will explore integrating additional computational intelligence techniques, such as deep learning and reinforcement learning, and developing automated tools to advance SDEE practices further. These advancements contribute to more reliable and efficient software project management, facilitating real-time effort estimation and informed decision-making in the software industry.

KEYWORDS: Software Cost Estimation, Functional Link Artificial Neural Network, Fuzzy Logic System, Interval Type-2FLS, Particle Swarm Optimization, Active Learning Algorithm

I. INTRODUCTION

Fred Brooks (2003) identified software cost estimation as one of the three significant challenges in computer science [1]. Every year, a significantly large number of new applications are produced, and existing applications are modified. Thus, software cost estimation is a significant activity for almost every software company. According to the Standish Group Chaos Report (2015), 19% of software projects fail because of poor software cost estimation practices [2]. In this context, our study aims to devise a suitable SDEE technique. The primary task of software cost estimation is to estimate the total effort required to complete a project, as the cost of effort dominates the project's cost. Estimated software cost serves as the basis for almost every project planning activity. Therefore, inaccurate cost estimation can have serious consequences. Two prime issues in software cost estimation are over and under-estimation. Underestimating the cost of a project leads to the allocation of less staff, the design of short schedules, and the production of low-quality deliverables. In contrast, project cost overestimation can lead to customer cancellation or overallocation of resources, resulting in underproductivity.

A significant challenge in cost estimation is accurately estimating the size of the software artifact to be developed in the planning phase long before the commencement of project development work. Due to the inherent uncertainties in any project, most current techniques used for software cost estimation tend to be inaccurate in the early stages of software development and only improve as the project heads toward the last stage and when most of the risks have been resolved. Most traditional parametric software effort estimation models are based on multiple regression approaches. These models aim to accurately predict the effort by calibrating actual data collected from completed software projects. Examples of popular parametric effort prediction models include the constructive cost model (COCOMO) [3] and software life cycle management (SLIM) [4]. When applied to software engineering data, these models face serious challenges that are typically scarce, incomplete, and imprecise.

In response to these challenges, considerable efforts have been dedicated to developing estimators based on computational intelligence (CI) techniques, which synergize neural networks, evolutionary computation, fuzzy systems (FS), and swarm intelligence (SI). CI techniques are popular because they do not require precise models for evaluating the cost function [5]. Despite the advances in CI, there still needs to be more integration of advanced CI techniques, such as interval type-2 fuzzy logic systems (IT2FLS) and swarm intelligence, in SDEE.

Our study addresses this research gap by developing a novel hybrid estimator, the Swarm Intelligence-Based Functional Link Fuzzy Neural Estimator (SFNE), specifically designed for SDEE. The significant contributions of this study are as follows:

1. Developed a hybrid Swarm Intelligence-Based Functional Link Fuzzy Neural Estimator (SFNE) model to improve the accuracy of software effort estimation.
2. A data reduction technique using the QUICK algorithm was introduced to optimize datasets by eliminating noise and redundancy, enhancing the model's performance.
3. Integrated Functional Link Artificial Neural Network (FLANN) for efficient computational processing, which captures complex nonlinear relationships in the data.
4. Applied Interval Type-2 Fuzzy Logic Systems (IT2FLS) to manage uncertainty and imprecision in software effort datasets, improving the robustness of the model.
5. Utilized Particle Swarm Optimization (PSO) to fine-tune the model's output parameters, ensuring high accuracy and minimizing prediction errors.
6. Validated the SFNE model through extensive experiments on six datasets from the PROMISE repository, demonstrating superior performance compared to other estimation techniques.

This research presents the development of a novel Swarm Intelligence-Based Functional Link Fuzzy Neural Estimator (SFNE) for Software Development Effort Estimation (SDEE). The SFNE model integrates advanced computational intelligence techniques, including Interval Type-2 Fuzzy Logic Systems (IT2FLS), active learning, and Particle Swarm Optimization (PSO), to enhance the accuracy and reliability of effort prediction. The model addresses critical limitations of traditional estimation approaches by combining data reduction methods and Functional Link Artificial Neural Networks (FLANN). Empirical validation using six benchmark datasets from the PROMISE repository demonstrates the superior performance of SFNE, establishing it as a robust and effective tool for software effort estimation. In the following section, we first discuss the different types of fuzzy systems. Subsequently, we discuss the architecture of FLANN.

A. TYPE 1 FUZZY LOGIC SYSTEM VS. INTERVAL TYPE 2 FUZZY LOGIC SYSTEM

The transition from a crisp set to a fuzzy set becomes necessary when assigning an element's membership as 0 or 1 is challenging. A type-1 fuzzy set (T1FS) is utilized in such scenarios, where the membership grade can be represented as a crisp number within the interval [0, 1]. However, when the uncertainty is so profound that even the membership grade cannot be precisely determined within this interval, a type-2 fuzzy set (T2FS) is employed [6], [7]. Both type-1 and type-2 fuzzy logic systems (FLS) are regarded as state-of-the-art methodologies for managing uncertainty in complex real-world problems. The primary distinction between T2FLS and T1FLS lies in the enhanced degree of design flexibility inherent to T2FLS. The structure of a general type-2 fuzzy logic system (T2FLS), illustrated in FIGURE 1, incorporates a type reduction process and defuzzification, distinguishing it from a type-1 fuzzy logic system (T1FLS). In T1FLS, the output processor directly maps a fuzzy set to a crisp number.

In contrast, T2FLS involves two stages: type reduction, which converts a type-2 fuzzy set into a type-1 fuzzy set, and defuzzification, which then transforms the type-1 fuzzy set into a crisp number. This added complexity renders T2FLS computationally intensive and more challenging to implement than T1FLS. To address these computational demands, interval type-2 fuzzy logic systems (IT2FLS) have been developed. IT2FLS retains the capability to manage uncertainties effectively while reducing computational complexity. This adaptation has led to the widespread adoption of IT2FLS in various applications, as it strikes a balance between handling uncertainty and maintaining computational efficiency [8-10]. Type-1 fuzzy sets denote the degree of membership of a crisp value x' of a base variable x in a fuzzy set A, characterized by a crisp membership function. $\mu_A(x')$ that assumes values within the interval [0, 1]. Such a set can be formally represented by Eq. (1) [7]:

$$A = \{(x, \mu_A(x)) | \forall x \in X\} \tag{1}$$

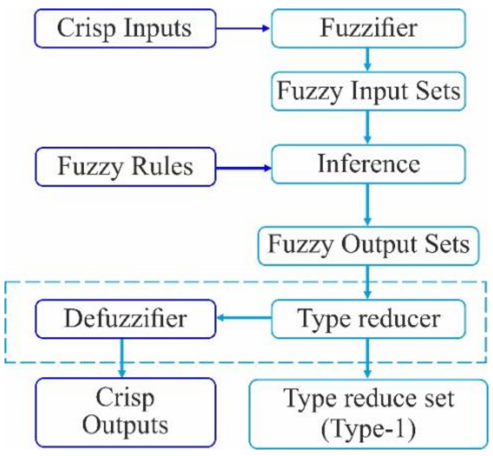


FIGURE 1. General structure of type-2 FLS

Consequently, type-1 fuzzy logic systems (FLS) are limited in handling uncertainties as they require precise identification of membership functions. To address this limitation, Castillo et al. (2007) introduced type-2 fuzzy logic systems (T2FLS) to mitigate the impact of uncertainty within the rule base [11]. This advancement has facilitated the application of T2FLS in various fields, including modeling. Type-2 FLS has since been extensively utilized across domains such as control systems, data mining, system identification, forecasting, computer vision, and pattern recognition.

The representation of a general type-2 and IT2FLS differs from that of type-1 FLS by a tilde symbol. For example, if A

denotes a type-1 fuzzy set, then $\tilde{A}$ denotes interval type-2 fuzzy set or type-2 fuzzy set. A type-2 fuzzy set, denoted by $\tilde{A}$, is characterized by a membership function (MF), $\mu_{\tilde{A}}(x,u)$, where $x \in X$ and $u \in J_x \subseteq [0,1]$, that is, The notation for general type-2 and interval type-2 fuzzy logic systems (IT2FLS) includes a tilde symbol to distinguish them from type-1 fuzzy logic systems (T1FLS). For instance, while A represents a type-1 fuzzy set, $\tilde{A}$ denotes an interval type-2 fuzzy set or a type-2 fuzzy set. A type-2 fuzzy set, denoted by $\tilde{A}$, is defined by a membership function (MF), $\mu_{\tilde{A}}(x,u)$, where $x \in X$ and $u \in J_x \subseteq [0,1]$ such that:

$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u) | \forall x \in X, \forall u \in J_x \subseteq [0,1]\} \qquad (2)$$

Where $0 \le \mu_{\tilde{A}}(x,u) \le 1$.
Eq. (2) [8] defines a type-2 fuzzy set $\tilde{A}$ with a membership function $\mu_{\tilde{A}}(x,u)$ that assigns a membership grade to each pair (x,u), where x is an element from the domain X and u is an element of primary membership J_x, which is a subset of the interval [0,1]. This equation indicates that for every $x \in X$, there is a set J_x of secondary membership values u that lie within [0,1].

The amplitude of a secondary membership function is termed the secondary grade. In Eq. (2) [8], $\mu_{\tilde{A}}(x,u)$ for $x \in X$, $u \in J_x \subseteq [0,1]$ is a secondary grade. When the values of the secondary grade are uniformly equal to 1, it results in an interval type-2 membership function. Thus, $\forall x \in X$ if $\mu_{\tilde{A}}(x,u) = 1$, then $\tilde{A}$ is an interval type-2 fuzzy set.

FIGURE 2 elucidates the definition and representation of a type-2 fuzzy set. The diagram represents the primary membership (x-axis), where x′ is a specific value, and the secondary membership (u-axis), indicating secondary membership values u within [0,1]. The shaded area enclosed by the outer boundary represents the Footprint of Uncertainty (FOU), capturing possible membership values. The upper and lower boundaries of the FOU denote the upper membership function ($\overline{\mu}_{\tilde{A}}$) and lower membership functions ($\underline{\mu}_{\tilde{A}}$), respectively. For each x′, the vertical slice through the FOU represents the secondary membership function $J_{x'}$. The point A_e illustrates a specific instance of primary and secondary membership values.

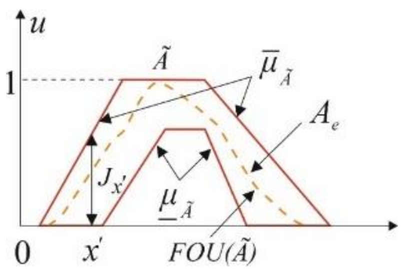


FIGURE 2. IT2 FL and its associated quantities.

B. ARCHITECTURE OF FLANN

Initially proposed by Pao, FLANN is a novel single-layer neural network with a faster convergence rate; it is a computationally efficient neural network model compared with MLP [12]. The typical structure of FLANN is shown in FIGURE 3. The nonlinearity in FLANN is introduced by orthogonal functional expansions (i.e., basis functions). The commonly used basis functions are Chebyshev polynomial, Legendre polynomial, and power polynomial. Software effort estimation is a functional approximation optimization problem. The goal of FLANN can be defined as selecting a basis function to learn the effort estimation function $f(X)$ by approximating the function$f_W(X)$. The interpolation of the function $f(X)$ is achieved by FLANN. W is the set of weights to be optimized to obtain the best approximate of $f(X)$.

In this study, we selected the Chebyshev polynomial functional expansion as the basis function because of its low error estimation characteristics. The basis function nonlinearly transforms the input space (low dimensions) into feature space at high dimensions. The feature space was multiplied by the weight vector, resulting in normalized output in the range [-0.5, +0.5]. The summation was given as input to the sigmoid function for predicting the cost function. The optimal cost function was obtained by iteratively updating the weight vector.

The PSO learning algorithm modified the weight vector.

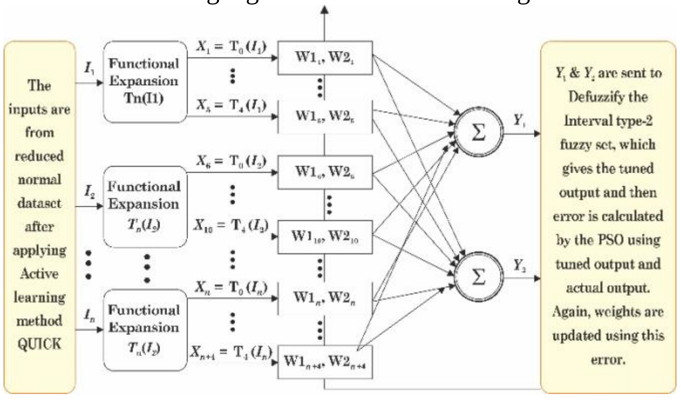

The inputs are from reduced normal dataset after applying Active learning method QUICK
I_1
Functional Expansion Tn(I1)
I_2
Functional Expansion $T_n(I_2)$
I_n
Functional Expansion $T_n(I_2)$
$X_1 = T_0(I_1)$
$X_5 = T_4(I_1)$
$X_6 = T_0(I_2)$
$X_{10} = T_4(I_2)$
$X_n = T_0(I_n)$
$X_{n+1} = T_4(I_n)$
W1$_1$, W2$_1$
W1$_5$, W2$_5$
W1$_6$, W2$_6$
W1$_{10}$, W2$_{10}$
W1$_n$, W2$_n$
W1$_{n+4}$, W2$_{n+4}$
Σ
Σ
Y_1
Y_2
Y_1 & Y_2 are sent to Defuzzify the Interval type-2 fuzzy set, which gives the tuned output and then error is calculated by the PSO using tuned output and actual output. Again, weights are updated using this error.

FIGURE 3. FLANN

C. ACTIVE LEARNING FOR DATASET REDUCTION

Active learning was introduced by Simon in 1974 [13]. The key idea behind active learning is to improve the performance of the FLANN by choosing valuable samples from the software effort estimation dataset. QUICK, an active learning algorithm proposed by Ekrem Kocaguneli et al. (2012), has been utilized in this study. It identifies the essential content of the dataset fed to the SFNE model to improve estimation accuracy [14]. The QUICK method has two principal components-synonym pruning and outlier pruning.

Initially, the dataset was represented by a 2D matrix. The rows represent project instances, and the columns describe the features or attributes. The dataset undergoes transposition in synonym pruning, and the similarity measure between attributes is calculated based on Euclidean distance. After obtaining the distance matrix, similar values in each row rank are assigned by incrementing by 1. Attributes having similar neighbors represented by the popularity index, that is, the most famous attributes, are eliminated. Next, the process of outlier pruning is initiated. The obtained matrix is transposed to restore its original form. Now, the rows represent project instances, and the columns represent features. The matrix contains only selected features from the previous phase. The distance matrix is generated using an Euclidean distance measure. The rows are sorted based on the distance. The k-closet neighbors of another instance are defined to be popular. The system retains the popular ones while removing the unpopular project instances. Thus, we obtain the most useful data samples and feed them to the next stage.

D. PARTICLE SWARM OPTIMIZATION LEARNING ALGORITHM

PSO is a population-based multi-agent stochastic algorithm proposed by Eberhart and Kennedy in 1995 [15]. In PSO, the particle represents a potential solution. A set of possible solutions is called a "Swarm." Each particle's state information is represented by its position and velocity in the search space. Randomly generated particles (known as trial solutions) are selected to fly through a D-dimensional search space toward the optimal solution over some iterations by utilizing its best position and global best particle state in each iteration.

In the D-dimensional design space, the position and velocity vectors of i^{th} particle for the d^{th} dimension are assumed to be $\vec{x}_{id}$ and $\vec{v}_{id}$, respectively. At any sampling instance t, the velocity and position can be represented by Eq. (3) and Eq. (4) [24]:

$$\vec{v}_{id}(t+1) = \overrightarrow{w_i} \otimes \underline{\vec{v}_{id}(t)} + \overrightarrow{c_1} \otimes \overrightarrow{r_1}\left(\overrightarrow{pbest_{id}}(t) - \vec{x}_{id}(t)\right) + \overrightarrow{c_2} \otimes \overrightarrow{r_2} \otimes \left(\overrightarrow{gbest} - \vec{x}_{id}(t)\right) \quad (3)$$

$$\vec{x}_{id}(t+1) = \vec{x}_{id}(t) + \vec{v}_{id}(t+1) \quad (4)$$

The symbol $\otimes$ denotes point-by-point vector multiplication, w is the inertia weight; c_1 and c_2 They are chosen uniformly and randomly in some interval [0, 2], and they are the acceleration coefficients influencing the maximum size of the step a particle can take in one iteration. State space exploitation is introduced by vectors random numbers r_1 and r_2. They are the distributed random numbers in the range [0, 1]. To ensure optimal convergence, the adaptive inertia weight strategy is adopted. The inertia weight decreases as the generation increases to balance the exploration and exploitation trade-off in the search space.

E. DESIGN OF INTERVAL TYPE-2 FUZZY SETS (IT2FSS) FOR ATTRIBUTES

This section shows how IT2FSs for an attribute of the COCOMO81 dataset were designed. FIGURE 4 depicts the algorithm (APPENDIX A: Algorithm 1). COCOMO'81 has 17 attributes; the last attribute, i.e., the 17th attribute, is the output attribute. The MATLAB FIS editor represents all the attributes, as shown in FIGURE 4(a) and 4(b). The membership function can be of any type; we chose the Gaussian membership function with five linguistic variables (very low, low, moderate, high, and very high). The standard deviation is updated, and membership values are calculated and represented as in Algorithm 2 (APPENDIX A).

Illustration of Algorithm 1: Algorithm 1 delineates a systematic procedure for transforming type-1 fuzzy sets into type-2 ones within a Functional Link Artificial Neural Network (FLANN) framework. The algorithm proceeds through several key steps to ensure a robust and precise conversion process.

Initially, the algorithm computes the membership values associated with type-1 fuzzy sets for each data instance. Let N denote the number of data instances and M the number of attributes. For each data instance $i\ (where\ i = 1, \dots, N)$ and each attribute $j\ (where\ j = 1, \dots, M-1)$, the algorithm performs the following computations:

1. **Calculation of Mean membership values:** The mean of the type-1 membership values across all linguistic variables is calculated as in Eq. (5) [7]:

$$\mu_2(i,j) = \frac{1}{k}\sum_{l=1}^{k} \mu_1(i,j,l) \quad (5)$$

Where $\mu_1(i,j,l)$ represents the membership value for the l-th linguistic variable (with $l = 1, \dots, 5$) of the j-th attribute in the i-th data instance, $k = 5$ is the total number of linguistic variables considered.

2. Calculation of Standard Deviation of Membership Values: The standard deviation of the type-1 membership values is determined to quantify their variability as shown in Eq. (6) [7]:

$$\sigma_2(i,j) = \sqrt{\frac{1}{k}\sum_{l=1}^{k} \mu_1(i,j,l) - \mu_2(i,j)^2} \quad (6)$$

These statistical measures, $\mu_2(i,j)$ and $\sigma_2(i,j)$, effectively capture the central tendency and dispersion of the membership values for each attribute, thereby facilitating the construction of type-2 fuzzy sets.

Following the computation of the mean and standard deviation, each attribute in the dataset is segmented according to the derived type-2 fuzzy parameters. Although the algorithm permits flexibility in determining the number of segments, this number must match the output nodes of the FLANN's final layer. In the present implementation, two segments are employed, facilitating defuzzification through which type-2 fuzzy sets are converted into crisp output values for effective decision-making. This methodological approach preserves the integrity of the fuzzy logic system by ensuring that each dataset value is accurately represented through its mean membership and associated variability.

For example, if the mean value of FLANN [1, 1] = 0.3456, then the first segment ranges from 0 to (0.3456/2), and segment 2 spans from (0.3456/2) to 0.3456. Thus, we get segments that are divided into equal intervals of type-2 mean called interval type-2. According to step 4.4 of the algorithm, each value is divided into two segments. The lower and upper membership values are obtained with the help of R, α, and β for both segments using the formulas in Algorithm 2. Both segments' upper and lower means are calculated as mentioned in step 4.5. Membership values are obtained using step 4.6. Finally, T-norm is performed to obtain interval type-2 fuzzy set as cited in step 4.7.

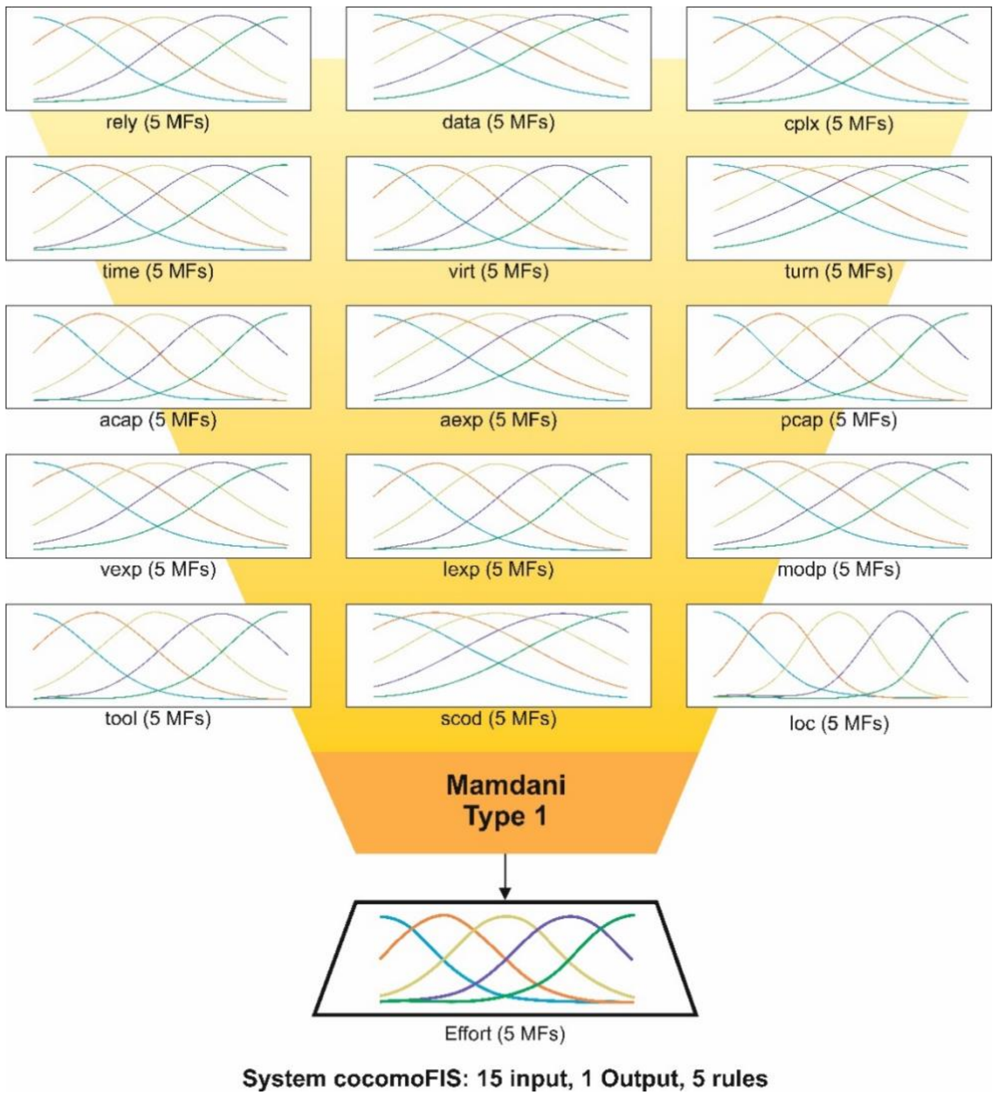
The rest of the paper is structured as follows: Section II outlines the related work, Section III presents the methodology used in this study, Section IV summarizes the results and findings- compares our model's performance with other prediction models such as ANN, FLANN with gradient descent, and RBF. Section V presents the discussion. Section VI describes the threats to validity, and Section VII concludes the article.

II. RELATED WORK

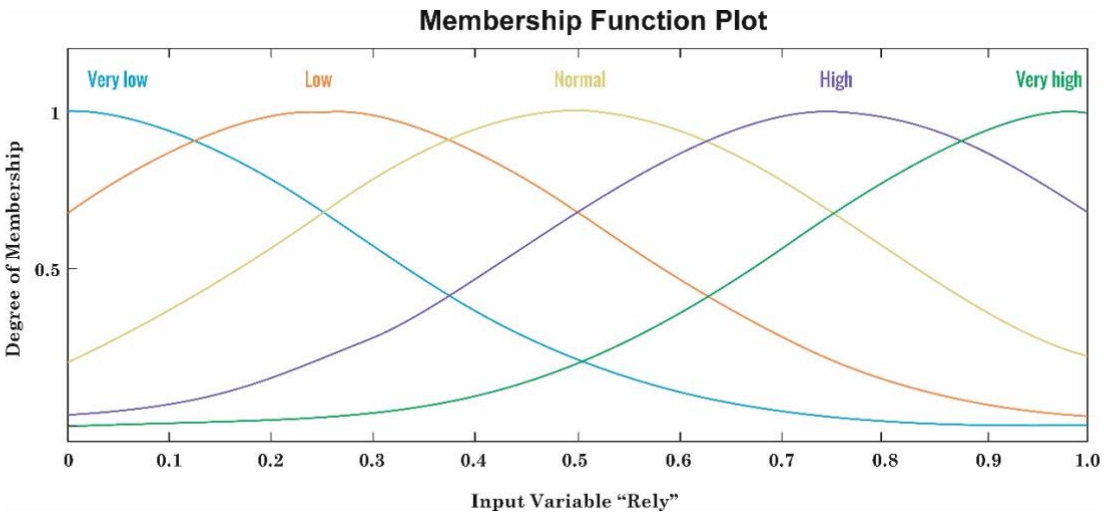
The application of computational intelligence (CI) in software development effort estimation (SDEE) has been extensively studied. This section provides a brief review of notable research in this area. Muzaffar et al. (2010) examined factors within fuzzy logic systems that influence the accuracy of software development effort predictions [16]. Their findings indicate that a fuzzy logic-based prediction system's architecture, parameters, and training algorithms critically impact accuracy. Ahmed and Muzaffar (2009) proposed a type-2 fuzzy logic framework to handle the imprecision and uncertainty inherent in effort prediction data [17].

In 2004, Xu and Khoshgoftaar introduced an innovative fuzzy identification cost estimation technique that automatically generates fuzzy membership functions and rules to process linguistic data [18]. In 2009, Azzeh, Neagu, and Cowling combined fuzzy set theory with grey relational analysis to enhance software project similarity measurement and attribute weighting in effort-based analysis (EBA) [19]. Sheta utilized the Takagi-Sugeno method to develop fuzzy models for two nonlinear processes in 2006 [20], while Lee *et al.* proposed a fuzzy size estimation procedure for goal-driven use case models based on Use Case Points (UCP) in 2011 [21]. In 2012, Ekrem *et al.* explored active learning-based effort estimation, identifying essential components of SDEE datasets and recommending suitable estimation methods for different datasets.

Benala *et al.* reported using Functional Link Artificial Neural Networks (FLANN) for SDEE in 2009, conducting empirical validation using the COCOMO'81 dataset from the PROMISE repository. Their promising results highlight FLANN's simple architecture and computational efficiency compared to multilayer perceptrons (MLP) [22]. In 2012, Benala *et al.* proposed three approaches for SDEE using FLANN, introducing a preprocessing step called optimally reduced datasets (ORDs) [23]. These ORDs reduced the dataset to small, representative subsets, which were then processed by FLANN and tested on eight polynomial expansions. Their methods demonstrated superior performance over conventional FLANN, support vector machine regression (SVR), radial basis function (RBF), and classification and regression trees (CART). They also explored genetic algorithm optimization for FLANN and fuzzy clustering combined with FLANN for SDEE. In 2013, Benala et al. utilized particle swarm optimization (PSO) to optimize FLANN feature weights. This resulted in the PSO-FLANN framework, which showed promising outcomes [24]. 2014, they proposed a new model that integrates active learning and PSO in FLANNs to improve software effort estimation [25]. In 2024, Manchala, P., & Bisi, M. present a novel model using feature selection and parameter optimization to enhance the accuracy of software development effort estimation (SDEE) through a two-stage optimization technique integrating improved social network search algorithms with an adaptive neuro-fuzzy inference system (ANFIS) [26].



(a)



(b)

FIGURE 4. Membership function of COCOMO81 for one attribute: (a) FIS for COCOMO81; (b) membership function for Input Variable Rely

III. METHODOLOGY

The Methodology section delineates a systematic framework for developing and assessing the Swarm Intelligence-Based Functional Link Fuzzy Neural Estimator (SFNE) tailored for Software Development Effort Estimation (SDEE). This section is structured into four primary subsections to facilitate clarity and ensure technical coherence. Initially, the **Dataset Description** subsection provides a comprehensive overview of the six benchmark datasets utilized in this study, sourced from the PROMISE repository. These datasets encompass a variety of software project attributes, including effort metrics, development environments, and project scales, accompanied by statistical analyses to underscore the complexities inherent in accurate effort estimation across diverse domains. Subsequently, the **High-Level Algorithm of SFNE** subsection expounds on the proposed estimator's architectural framework and integral components. It elucidates the integration of Functional Link Artificial Neural Networks (FLANN), Interval Type-2 Fuzzy Logic Systems (IT2FLS), and Particle Swarm Optimization (PSO), highlighting the model's capability to manage uncertainty and nonlinear relationships through its layered structure. The **Experimental Procedure** subsection outlines the stepwise implementation process of SFNE, encompassing data preprocessing via the QUICK algorithm and the application of leave-one-out cross-validation (LOOCV) for model evaluation. Finally, the **Performance Evaluation Metrics** subsection defines the criteria employed to measure SFNE's efficacy, robustness, and adaptability, including metrics such as MMRE, MdMRE, and PRED(0.25). This methodical approach ensures a logical progression, technical rigor, and reproducibility, substantiating SFNE's viability as a robust tool for SDEE.

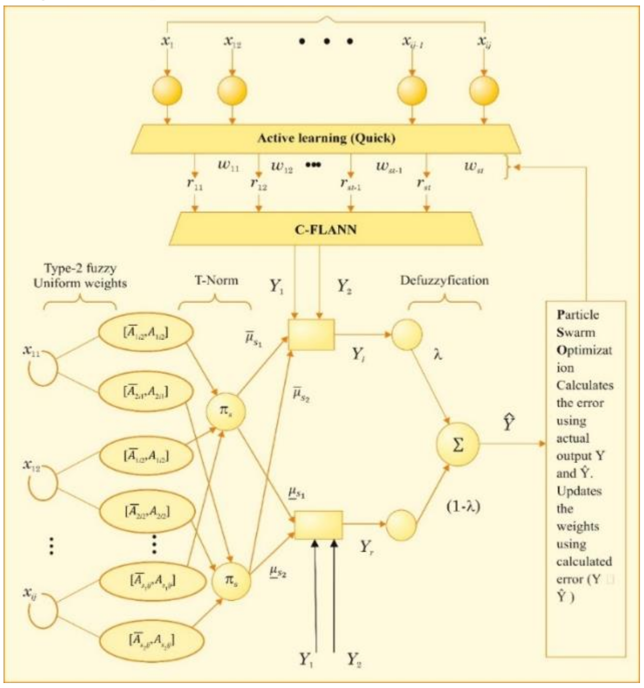


FIGURE 5. Layered structure of SFNE using the Mamdani Fuzzy Reasoning System

A. DATASET DESCRIPTION

The evaluation utilized six datasets from the PROMISE repository, which is publicly accessible at the PROMISE repository. These datasets are categorized based on their sources and characteristics:

Desharnais: Contains Canadian software projects.
COCOMO81: Projects developed in the United States.
NASA93: Projects developed in the United States.
China: Chinese software projects.
Maxwell: Finnish banking software projects.
Albrecht: IBM projects from the 1970s.

In 2011, Dejaeger *et al.* classified these datasets into four categories: size features, development features, environment features, and project data [29]. Table 1 provides the descriptive statistics of these datasets, illustrating the variability and skewness of effort values, posing challenges for accurate estimation.

TABLE 1
Descriptive statistics of the datasets

	Features	Number of Projects	Effort Data					
			Unit	Min	Max	Mean	Median	Skew
Desharnais	12	77	Hours	546	23,940	5046	3647	2.0
Nasa	3	18	Months	5	138.3	49.47	26.5	0.57
Cocomo	17	63	Months	6	11,400	686	98	4.4
China	18	499	Hours	26	54,620	3921	1829	3.92
Maxwell	27	62	Hours	583	63,694	8223.2	5189.5	3.26
Albrecht	7	24	Months	1	105	22	12	2.2

B. HIGH-LEVEL ALGORITHM OF SFNE

SFNE is an integrated model that combines two computational models: FLANN and interval type-2 fuzzy logic system. Thus, it has the advantage of both models. Further, it incorporates the active learning technique, namely QUICK, as a preprocessing step. This technique helps improve the prediction accuracy by feeding essential inputs to the system. Adaptive PSO updates the weight in FLANN. Hence, it always converges to global optima, unlike FLANN with backpropagation, which may trap in local minima.

SFNE is typically a 5-layer network, as shown in FIGURE 5. The first layer takes the input from the dataset and transmits it to the next layer. Layer 2 initiates the interval type-2 fuzzy logic system (IT2FLS). Our method incorporates a variant of Wang and Mendel (1991) approach for generating fuzzy rules from standard data [27]. The type-2 Gaussian membership function with uncertain mean is considered for the antecedent and consequent variables, and the type-2 Gaussian with uncertain standard deviation is considered the membership function for the input in this work. An IT2FLS deals with the lower and upper membership functions. Nodes in Layer 3 receive degrees of associated rules from nodes in Layer 2. In the proposed model, two rules are generated, each having two values (lower and upper). Thus, the total output of this layer is four. Nodes in Layer 4 are called consequent nodes. The outputs obtained from the third layer and two local outputs of FLANN are considered inputs to this layer. The final layer is the output processing layer, which comprises two back-to-back components. The first element is a type reducer. We adopted the Karnik and Mendel algorithm [28] for type reduction. The defuzzification component computes the final crisp output. Algorithm 2 represents the methodology (as illustrated in APPENDIX A).

C. EXPERIMENTAL PROCEDURE FOR SFNE USING SAMPLE EXAMPLE

The simple SFNE comprises a hypothetical dataset containing four rows and four columns, as shown in Table 2. The step-by-step procedure for applying the SFNE model to the given dataset follows. APPENDIX B depicts the illustration for building interval type-2 FL.

TABLE 2
Sample Dataset (D)

	1	2	3	4
1	0.8800	1.1600	0.7000	1
2	0.8800	1.1600	0.8500	1
3	1	1.1600	0.8500	1
4	0.7500	1.1600	0.7000	1

Step 1) The dataset is normalized using min-max normalization.

TABLE 3
Normalized Dataset

	1	2	3	4
1	0.2000	1.0000	0	0
2	0.2000	1.0000	0.3846	1.0000
3	0	1.0000	0	0
4	0.1579	0	0.1579	0

Step 2) The sample SFNE system adopts the leave-one-out-cross-validation (LOOCV) technique as a sampling method. It performs four iterations of model building and evaluation. The results of the first two iterations are displayed in Tables 4 and 5. This process is continued for four iterations as four data points are available in the dataset.

TABLE 4
First Iteration (after applying LOOCV)

		1	2	3	4	
Train	1	0.2000	1.0000	0	0	
	2	0.2000	1.0000	0.3846	1.0000	
	3	0	1.0000	0	0	
	4	0.1579	0	0.1579	0	Test dataset

TABLE 5
Second Iteration (after applying LOOCV)

		1	2	3	4	
	1	0.2000	1.0000	0	0	Test dataset
Train	2	0.2000	1.0000	0.3846	1.0000	
	3	0	1.0000	0	0	
	4	0.1579	0	0.1579	0	

Step 3) The preprocessing step adapted in our work involves the active learning method QUICK to obtain useful samples. Table 5 displays the output of this step. The system forwards the results obtained to the next stage, FLANN, to predict the output.

TABLE 6
Matrix after application of quick method

	1	2	3
1	1.0000	0	0
2	1.0000	0.3846	1.0000
3	1.0000	0	0

D. BUILDING INTERVAL TYPE-1 FL

Step 4) Type-1 fuzzy logic system is applied to the normalized dataset parallel to steps 2 and 3 to convert each attribute to the fuzzy number.

Step 5) The mean and standard deviation for each attribute in Table 3 are obtained using MATLAB functions "mean" and "std." Table 6 displays the obtained results. As each attribute has a single value, Tables 3 and 6 are identical.

TABLE 7
Matrix representing the mean and standard deviation of each attribute of table 2

	1	2	3	4
1	0.2000	1.0000	0	0
2	0.2000	1.0000	0.3846	1.0000
3	0	1.0000	0	0
4	0.1579	0	0.1579	0

Step 5.1 The rules are created using the Mamdani FIS. The Wang and Mendel approach (1991) is used to define the input and output variable fuzzy sets and initiate the FIS. In our work, the fuzzy sets for the input and output variables are created using a Gaussian membership function. Suppose the domain interval of each attribute is $[x^-, x^+]$, i.e., the most probable range of the attributes. In our work, the interval spans from [0, 1] as both input and output variables are normalized in the range [0, 1]. The domain interval is divided into 2N+1 regions, with N=2, i.e., and each attribute has five fuzzy regions, as shown in FIGURE 4(b).

Step 5.3 The function $gaussmf\left(matrix_{some_{x(i,j)}}, [\sigma, c]\right)$ is an in-built function in MATLAB that helps find membership values for all variables in the normalized dataset, each comprising five different membership values based on number of linguistic variables. Table 8 indicates that the membership values for all attributes of example Table 7 are very low, thus obtaining type-1 FL membership values.

TABLE 8
Matrix representing lower membership values of each attribute

	1	2	3	4
1	0.7980	0.0112	1.0000	1.0000
2	0.7980	0.0112	0.7601	1.0000
3	0	1.0000	0	0
4	0.1579	0	0.1579	0

E. PERFORMANCE EVALUATION METRICS

This section describes the metrics employed to evaluate the performance of software development effort estimation (SDEE) models. Performance metrics are essential for assessing the effectiveness of these models. For a comprehensive evaluation, five widely recognized metrics were selected: MMRE, MdMRE, PRED (0.25), SA, and Delta.

The Mean Magnitude of Relative Error (MMRE) is a critical metric for evaluating the accuracy of predictive models in software estimation. It is calculated by first determining the Magnitude of Relative Error (MRE) for each prediction, as defined by Eq. (7) [34]:

$$MRE_i = \left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{7}$$

where y_i is the actual value and $\hat{y}_i$ is the predicted value for the i-th instance. Subsequently, MMRE is obtained by averaging all MRE values across the dataset, as shown in Eq. (8) [34]:

$$MMRE = \sum_{i=1}^{n} MRE_i / n \tag{8}$$

This approach normalizes the prediction errors, enabling a comprehensive assessment of the model's performance by averaging the relative discrepancies between predicted and actual values.

The Median Magnitude of Relative Error (MdMRE) serves as a robust global error metric that is less susceptible to the influence of outliers compared to mean-based measures. MdMRE is defined as the median value of all individual Magnitude of Relative Errors (MREs) within a dataset. Mathematically, it is expressed in Eq. (9) [34]:

$$MdMRE = median(MRE_i) \tag{9}$$

Where $MRE_i = \left|\frac{y_i - \hat{y}_i}{y_i}\right|$. By utilizing the median, MdMRE effectively captures the central tendency of the prediction errors, providing a more representative measure of a model's typical performance. This characteristic makes MdMRE particularly valuable for evaluating models in scenarios where outliers may distort average error assessments, thereby offering a clearer insight into the model's reliability and accuracy.

The PRED(x) metric quantifies the percentage of predictions that fall within a specified tolerance level, x%, of the actual values. Mathematically, PRED(x) is defined in Eq. (10) [34]:

$$PRED(x) = \frac{100}{N} \times \sum_{i=1}^{N} D_i \tag{10}$$

where D_i is an indicator function defined by Eq. (11) [34]:

$$D_i = \begin{cases} 1\ if MMRE < \dfrac{x}{100} \\ 0 \quad otherwise \end{cases}$$

When $x = 25$, the PRED metric is defined as
PRED (0.25). (11)

For example, when $x = 25$, the metric is referred to as PRED (0.25). This metric clearly measures the model's accuracy by indicating the proportion of predictions within 25% of the actual values. PRED(x) is particularly useful for assessing the reliability of predictive models, as it highlights the consistency of predictions within an acceptable error margin, thereby offering valuable insights into the model's practical applicability and performance.

Traditional measures based on Magnitude of Relative Error (MRE), such as MMRE and PRED, exhibit several limitations, including an asymmetric error distribution that often biases results toward certain prediction models [33],[34]. Consequently, these MRE-based metrics may underestimate errors. To overcome this issue, researchers have adopted the Mean Absolute Error (MAE), which is calculated by first determining the Absolute Error (AE) for each prediction in Eq. (12) [33]:

$$AE_i = |y_i - \hat{y}_i| \tag{12}$$

Where y_i and $\hat{y}_i$ are actual and predicted efforts, respectively. The MAE is then obtained by averaging all absolute errors as shown in Eq. (13) [33]:

$$MAE = \frac{\sum_{i=1}^{N} AE_i}{N} \tag{13}$$

Unlike MRE-derived metrics, MAE is not susceptible to asymmetric distributions and thus provides a more balanced view of prediction accuracy. However, MAE alone can be difficult to interpret because it lacks standardization. To address this challenge, Shepperd and MacDonell [33] introduced the concept of Standardized Accuracy (SA), defined in Eq. (14) [33]:

$$SA = 1 - \frac{MAE}{\overline{MAE_{P_0}}} \tag{14}$$

where MAE is the mean absolute error, and $\overline{MAE}_{p_o}$ denotes the mean absolute error obtained from a large number (typically 1000) of random guesses. SA thus indicates how effectively the model outperforms random guessing: values near zero suggest low reliability, whereas negative values are deemed unacceptable. In addition, the Effect Size (**Δ**) provides further insight into the magnitude of improvement over random guessing. Delta is used to assess the effect size improvement over random guessing and is depicted in Eq. (15) [33]:

$$\Delta = \frac{MAE - \overline{\mathrm{MAE}_{P_0}}}{s_{P_0}} \tag{15}$$

Where s_{P_0} is the sample standard deviation from the random-guessing strategy. Categorized as small (0.2), medium (0.5), or large (0.8), a $\Delta \geq 0.5$ is typically considered favorable [33], [34]. This standardized framework—encompassing MAE, SA, and Δ—enables robust, unbiased assessment of software effort estimation models, thereby ensuring more reliable and interpretable conclusions regarding their predictive performance.

IV. RESULT

The following sections detail cost estimation models and experimental results.

A. COST ESTIMATION MODELS

This study explores the feasibility of various models based on FLANN. The experimental setup includes C-FLANN, P-FLANN, and L-FLANN models. For simplicity, C-FLANN is referred to as FLANN in this paper. Other popular models, such as ANNs and RBFs, are also included for a comprehensive evaluation.

Radial Basis Function Networks (RBF) are designed using parameters such as the global width (σ) and a measure of closeness (δ) between the interpolation matrix and its approximation. The model determines the minimum number of basis functions required to achieve the desired approximation accuracy and identifies the centers of these basis functions. Weights to the output nodes are then computed using the pseudo-inverse method [30].

In ANN models, configuration parameters such as the number of hidden layers, hidden nodes, and transfer functions are critical in determining prediction accuracy. A single hidden layer is often recommended to avoid over-parameterization. The training phase involves using historical data to train ANN models with the specified configurations, and the model structure that results in the lowest MMRE is chosen for further analysis [31].

B. EXPERIMENTAL RESULTS

This section reports the results of applying all the techniques discussed in sections 3 and 4.3 in the testing stage. The best-performing technique is documented in bold. Tables 9–13 summarize all the methods applied to the PROMISE repository dataset. FIGURE 6 shows the real values (solid line) and the predicted values generated by the proposed SFNE model, as it has the best value of standardized accuracy (Sa) and Delta (dashed line) for the COCOMO81 dataset. Upon analysis of the empirical validation results, SFNE shows remarkable accuracy on all performance measures. In general, all results for SFNE are relatively good. The best results for all the performance measures across all the techniques are as follows:

(1) According to the performance indicator SA, it is 99.983% for the DesharnaisL3 dataset using the SFNE technique.
(2) According to the performance indicator Delta, it is 2.8498 for the COCOMO81 dataset using the SFNE technique.
(3) According to the performance indicator MMRE, it is 2.87E-05 for the DesharnaisL1 dataset using the SFNE technique.
(4) According to the performance indicator MdMRE, it is 7.1123E-09 for the DesharnaisL1 dataset using the SFNE technique.
(5) According to the PRED (0.25) performance indicator, all the methods seem very close to each other except RBF.

The results (1)–(4) are exhibited by SFNE. (1) emphasizes the fact that SFNE is not a random guess model and is meaningful; (2) solidifies the fact that the SFNE model is not by chance, and the value 2.8498 suggests that there is a large effect improvement over a random guess model; and (3) and (4) have been selected based on the goal as minimization of MMRE and MdMRE.

The worst results for all the performance measures across all the techniques are as follows:

(1) According to the performance indicator SA, it is 41.734 % for the Albrecht dataset using the RBF technique.
(2) According to the performance indicator Delta, it is 0.1432 for the China dataset using the RBF technique.
(3) According to the performance indicator MMRE, it is 0.34391 for the Albrecht dataset using the RBF technique.
(4) According to the performance indicator MdMRE, it is 0.030064 for the Albrecht dataset using the RBF technique.
(5) According to the PRED (0.25) performance indicator, all the methods seem very close to each other except RBF.

The experimental results suggest that SFNE outperforms all other models in most cases, and RBF performs worst in almost all the scenarios. PSO-FLANN exhibits performance comparable to SFNE in some instances but performs better than SFNE in others.

TABLE 9
Significance Testing - Standardized Accuracy (SA)

	PSO-FLANN	ACTIVE-FLANN-PSO	FLANN-FUZZY-PSO	SFNE	BP-ANN	BP-FLANN	RBF
Albrecht	92.664	82.084	54.038	92.913	90.917	91.512	41.734
China	99.811	99.674	97.702	99.81	97.8	99.629	94.347
Cocomo81	99.913	99.505	99.872	99.917	99.811	99.89	94.165
Cocomo81e	99.519	98.8672	98.414	99.632	99.009	99.49	32.466
Cocomo81o	95.869	91.558	87.307	95.09	85.031	93.114	69.704
Cocomo81s	99.138	99.343	97.401	99.796	98.145	98.759	48.469
Desharnais	99.894	99.822	99.667	99.748	99.81	99.893	90.256
DesharnaisL1	99.958	99.956	99.86	99.968	99.896	99.929	86.282
DesharnaisL2	99.955	99.95	99.869	99.97	99.924	99.922	74.612
DesharnaisL3	99.973	99.964	99.882	99.983	99.886	99.858	53.74
Maxwell	99.953	99.936	99.865	99.915	99.935	99.946	45.93
Nasa93	99.151	98.444	95.544	95.757	98.444	99.081	74.798
Nasa93_center_1	98.608	96.472	96.008	99.92	97.586	96.426	67.409
Nasa93_center_2	97.617	94.624	92.974	98.167	97.739	94.501	54.667
Nasa93_center_5	99.851	99.573	99.328	99.717	99.618	99.786	20.238

TABLE 10
Effect Size – (COHEN'S D) Glass Delta

	PSO-FLANN	ACTIVE-FLANN-PSO	FLANN-FUZZY-PSO	SFNE	BP-ANN	BP-FLANN	RBF
Albrecht	1.5834	1.4027	0.92339	1.5877	1.5536	1.1536	1.224
China	0.16072	0.1605	0.15733	0.15748	0.16072	0.1604	0.1432
Cocomo81	2.8496	2.8485	2.849	2.8498	2.8467	2.838	2.6857
Cocomo81e	2.5287	2.5121	2.5006	2.5316	2.5157	2.528	0.82493
Cocomo81o	0.84341	0.80548	0.76808	0.83656	0.74806	0.8191	0.6264
Cocomo81s	2.6896	2.6267	2.6425	2.6952	2.7075	2.6793	1.315
Desharnais	0.27793	0.27773	0.2773	0.27753	0.2777	0.2779	0.2577
DesharnaisL1	0.39191	0.39195	0.39152	0.3919	0.39166	0.3917	0.2952
DesharnaisL2	0.72081	0.72092	0.72019	0.72078	0.72059	0.7205	0.2119
DesharnaisL3	1.2581	1.2583	1.257	1.462	1.257	1.2567	1.258
Maxwell	1.4337	1.4335	1.4324	1.4332	1.4334	1.4336	1.2184
Nasa93	0.71417	0.70907	0.68819	0.71366	0.70907	0.7136	0.6137
Nasa93_center_1	0.91131	0.89158	0.88729	0.92344	0.90187	0.8911	0.6911
Nasa93_center_2	1.2326	1.2395	1.1739	1.9148	1.2341	1.1932	1.1522
Nasa93_center_5	1.9041	1.8988	1.8941	1.9015	1.8997	1.9029	1.4838

TABLE 11
MMRE Values

	PSO-FLANN	ACTIVE-FLANN-PSO	FLANN-FUZZY-PSO	SFNE	BP-ANN	BP-FLANN	RBF
Albrecht	0.0147	0.0359	0.0921	0.0142	0.0182	0.0651	0.34391
China	5.27E-05	9.11E-05	6.41E-04	6.14E-04	5.32E-05	1.03E-04	0.49378
Cocomo81	0.0023	0.0034	0.0029	0.0022	0.005	0.0131	0.15448
Cocomo81e	0.0017	0.004	0.0056	0.0013	0.0035	0.0018	0.23851
Cocomo81o	0.0069	0.0082	0.0212	0.0141	0.025	0.0115	0.55128
Cocomo81s	0.0059	0.0045	0.0178	0.0014	0.0127	0.0085	0.35287
Desharnais	6.21E-05	1.05E-04	1.95E-04	1.47E-04	1.11E-04	6.26E-05	0.60105
DesharnaisL1	3.79E-05	3.97E-05	1.27E-04	2.87E-05	9.43E-05	6.41E-05	0.66105
DesharnaisL2	5.26E-05	5.88E-05	1.53E-04	3.48E-05	8.87E-05	9.17E-05	0.46105
DesharnaisL3	9.61E-05	1.28E-04	4.20E-04	6.21E-05	4.08E-04	5.05E-04	0.77105
Maxwell	5.82E-05	8.02E-05	1.69E-04	1.06E-04	8.16E-05	6.69E-05	0.23105
Nasa93	0.0012	0.0022	0.0063	0.006	0.0022	0.0013	0.56105
Nasa93_center_1	0.003	0.0076	0.0086	1.73E-04	0.0052	0.0077	0.66105
Nasa93_center_2	0.0039	0.0088	0.0115	0.003	0.0037	0.009	0.36105
Nasa93_center_5	3.11E-04	5.89E-04	0.0014	8.88E-04	7.96E-04	4.45E-04	0.26105

TABLE 12
Pred Values

	PSO-FLANN	ACTIVE-FLANN-PSO	FLANN-FUZZY-PSO	SFNE	BP-ANN	BP- FLANN	RBF
Albrecht	0.83333	0.41667	0.20833	0.25	0.79167	0.83333	0.041667
China	1	1	1	1	1	1	0.96794
Cocomo81	1	1	1	1	1	1	0.61905
Cocomo81e	0.64286	0.5	0.42857	0.75	0.5	0.60714	0.035714
Cocomo81o	0.66667	0.54167	0.16667	0.6768	0.16667	0.54167	0.041667
Cocomo81s	0.54545	0.54545	0.36364	0.45455	0.90909	0.54545	0.090909
Desharnais	0.91358	0.82716	0.58025	0.74074	0.82716	0.91358	0.24
DesharnaisL1	1	1	0.95652	1	0.95652	1	0.2798
DesharnaisL2	1	0.91358	1	1	0.90909	1	0.96745
DesharnaisL3	0.5	0.6	0.3	0.6	0.5	0.5	0.19
Maxwell	0.8871	0.83871	0.66129	0.8342	0.82258	0.87097	0.2798
Nasa93	0.53763	0.4086	0.15054	0.17204	0.4086	0.51613	0.010753
Nasa93_center_1	0.2973	0.91892	0.081081	0.081081	0.18919	0.081081	0.00012
Nasa93_center_2	0.40541	0.51351	0.24324	0.2973	0.37838	0.35135	0.054054
Nasa93_center_5	0.51282	0.28205	0.10256	0.20513	0.23077	0.38462	0

TABLE 13
MDMRE Values

	PSO-FLANN	ACTIVE-FLANN-PSO	FLANN-FUZZY-PSO	SFNE	BP-ANN	BP-FLANN	RBF
Albrecht	0.001285	0.0031383	0.0080512	0.0012413	0.001591	0.0056909	0.030064
China	2.8814e-08	4.9809e-08	3.5046e-07	3.357e-07	2.9087e-08	5.6315e-08	0.00026997
Cocomo81	2.3469e-05	3.4694e-05	2.9592e-05	2.2449e-05	5.102e-05	2.3469e-05	0.0015763
Cocomo81e	4.8444e-06	3.7045e-06	1.5958e-06	1.1398e-06	9.9737e-06	5.1293e-06	0.00067966
Cocomo81o	0.00015007	0.00017835	0.00046109	0.00030667	0.00054374	0.00025012	0.01199
Cocomo81s	3.7821e-05	2.8846e-05	0.0001141	8.9744e-06	8.141e-05	5.4487e-05	0.002262
Desharnais	1.7028e-08	2.8791e-08	5.3469e-08	4.0307e-08	3.0436e-08	1.7165e-08	0.00016481
DesharnaisL1	9.3922e-09	9.8383e-09	3.1473e-08	7.1123e-09	2.3369e-08	1.5885e-08	0.00016382
DesharnaisL2	1.0916e-08	1.1434e-08	3.6578e-08	8.2661e-09	2.716e-08	1.8462e-08	0.00019039
DesharnaisL3	4.9836e-08	5.5711e-08	1.4496e-07	3.2972e-08	8.404e-08	8.6882e-08	0.00043683
Maxwell	1.1218e-08	1.5459e-08	2.2575e-08	1.0432e-08	1.5729e-08	1.2895e-08	4.4536e-05
Nasa93	4.7619e-06	8.7302e-06	2.5e-06	2.381e-06	8.7302e-06	5.1587e-06	0.0022264
Nasa93_center_1	4.5833e-05	2.6431e-06	0.00013139	0.00011611	7.9444e-05	0.00011764	0.010099
Nasa93_center_2	4.7561e-05	3.6585e-05	4.585e-05	3.2585e-05	4.5122e-05	0.00010976	0.004403
Nasa93_center_5	5.4428e-07	1.0308e-06	2.4501e-07	1.5541e-07	1.3931e-06	7.7879e-07	0.00045686

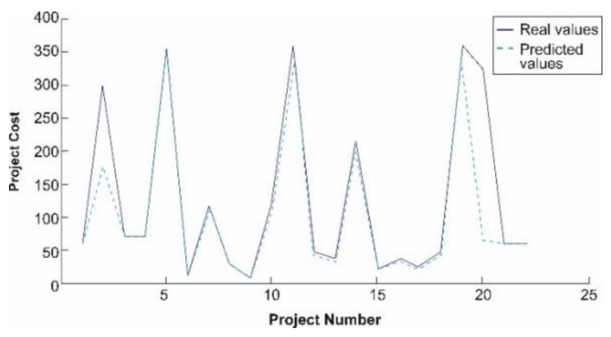


FIGURE 6. Prediction results for COCOMO'81 dataset (test set) using SFNE algorithm; actual values are denoted as solid lines, and predicted values are denoted as dashed lines

V. DISCUSSION

The proposed Swarm Intelligence-Based Functional Link Fuzzy Neural Estimator (SFNE) was rigorously evaluated against six benchmark models: BP-FLANN [22], FLANN [23], PSO-FLANN [24], ACTIVE-FLANN-PSO [25], RBF [30], and BP-ANN [31]. Utilizing six datasets from the PROMISE repository, the comparative analysis presented in Tables 9 through 13 underscores the superior performance of SFNE across multiple evaluation metrics.

A. ACCURACY COMPARISON

SFNE consistently outperformed all benchmark models across the evaluated datasets. For example, on the DesharnaisL3 dataset, SFNE achieved a standardized accuracy (SA) of 99.983%, which is marginally higher than PSO-FLANN [24] (99.973%) and BP-FLANN [22] (99.964%). Notably, the RBF model [30] lagged significantly with an SA of 53.74% (Table 9). Similarly, on the COCOMO81 dataset, SFNE attained an SA of 99.917%, demonstrating its robustness and adaptability across diverse datasets.

B. ERROR METRICS ANALYSIS

SFNE demonstrated substantial improvements in error reduction compared to conventional approaches. Specifically, on the DesharnaisL1 dataset, SFNE achieved a mean magnitude of relative error (MMRE) of 2.87E-05, indicating a significant reduction relative to the RBF model's MMRE of 0.66105 (Table 11). Additionally, the median magnitude of relative error (MdMRE) for SFNE on the same dataset was 7.1123E-09, markedly lower than the RBF model's MdMRE of 0.00016382(Table 13). These findings underscore SFNE's enhanced capability to minimize prediction errors compared to traditional models.

C. EFFECT SIZE AND PREDICTION ACCURACY

The Cohen's d effect size (Delta) further emphasizes SFNE's efficacy. On the COCOMO81 dataset, SFNE achieved a Delta of 2.8498, indicating a significant improvement over competing models (Table 10). Moreover, SFNE attained a PRED(0.25) of 100% on the COCOMO81 dataset, far exceeding the best value of 66% reported by authors in [23] for their FLANN models (Table 12). This further underscores the SFNE model's robustness in delivering accurate predictions within an acceptable error range, making it highly effective in practical applications of SDEE.

D. LIMITATIONS OF THE PROPOSED MODEL

While the Swarm Intelligence-Based Functional Link Fuzzy Neural Estimator (SFNE) demonstrates superior performance in Software Development Effort Estimation (SDEE), certain limitations must be acknowledged to guide future research and practical adoption.

1. Dataset Dependency: SFNE's performance heavily depends on the quality and diversity of the datasets used for training and validation. Although six datasets from the PROMISE repository were utilized, these datasets represent only a subset of real-world scenarios. This limitation may impact the model's generalizability across domains or less-represented software project characteristics.
2. Computational Complexity: Integrating advanced methodologies, such as Interval Type-2 Fuzzy Logic Systems (IT2FLS) and Particle Swarm Optimization (PSO), increases the computational overhead. This complexity may pose challenges in scenarios with limited computational resources or time-critical applications.
3. Sensitivity to Parameter Tuning: The SFNE model's accuracy is influenced by the choice of parameters, such as the membership functions in IT2FLS and the weights optimized by PSO. Improper parameter tuning could result in suboptimal performance, necessitating automated or adaptive approaches to parameter optimization.
4. Handling of Dynamic Data: The current model assumes static datasets and does not account for dynamic or evolving project environments. Real-time adaptation to changing project parameters and requirements remains an open area for enhancement.
5. Scalability for Large-Scale Projects: While SFNE has shown promising results for moderate-sized datasets, its scalability for large-scale, complex software projects with extensive attributes has not been extensively validated.

Addressing these limitations through further research, such as incorporating more diverse datasets, optimizing computational efficiency, and exploring adaptive techniques, will enhance the robustness and applicability of SFNE in broader software engineering contexts.

E. COMPARISON WITH PRIOR STUDIES

SFNE's advancements are particularly noteworthy when juxtaposed with prior studies. For instance, FLANN [23] and PSO-FLANN [24] have been pivotal in functional link neural network research. However, SFNE integrates interval type-2 fuzzy logic systems and particle swarm optimization, which allows it to handle better complex relationships and uncertainties inherent in software effort estimation. ACTIVE-FLANN-PSO [25] introduced adaptive mechanisms within the FLANN framework, yet SFNE's comprehensive approach results in superior performance metrics. Similarly, compared to BP-ANN [31], which relies on traditional backpropagation, SFNE's swarm intelligence-based optimization leads to faster convergence and higher accuracy. These comparisons affirm that SFNE builds upon existing methodologies and sets a new benchmark in the field.

F. SIGNIFICANCE AND PRACTICAL IMPLICATIONS

The empirical results validate SFNE as a robust and reliable software development effort estimation model. Its superior accuracy and error minimization across various datasets make it a valuable tool for practitioners seeking precise effort predictions. Furthermore, integrating swarm intelligence and fuzzy neural networks in SFNE offers a scalable and adaptable framework, paving the way for future research and development in effort estimation models.

In conclusion, SFNE outperforms existing benchmark models and advances state-of-the-art software effort estimation by leveraging advanced swarm intelligence and fuzzy logic techniques. These findings establish SFNE

VI. THREATS TO VALIDITY

Several threats to the validity of our study results exist. We discuss these threats following the guidelines provided by Runeson and Höst (2009) [32].

A. INTERNAL VALIDITY

One potential threat to this study's internal validity is the selection bias in the datasets used for training and testing the SFNE model. The six datasets from the PROMISE repository may have inherent characteristics influencing the model's performance. To mitigate this, a rigorous cross-validation technique, specifically leave-one-out cross-validation (LOOCV), was employed to ensure that the results are not overly optimistic due to specific data characteristics.

B. CONSTRUCT VALIDITY

Construct validity could be compromised if the performance metrics used do not adequately capture the effectiveness of the SFNE model. The study uses recognized metrics such as MMRE, MdMRE, PRED(0.25), SA, and Delta to evaluate the model. These metrics are standard in the field of software effort estimation, which strengthens the construct validity. However, relying solely on these metrics might not fully encapsulate all dimensions of model performance, and incorporating additional metrics or qualitative assessments could provide a more comprehensive evaluation.

C. EXTERNAL VALIDITY

The nature of the datasets limits the external validity of the findings. The six datasets from the PROMISE repository are diverse but may not cover the full spectrum of real-world software projects. Consequently, the generalizability of the results to other datasets or domains not represented in the study may be limited. Future work should include a wider variety of datasets from different domains to enhance the generalizability of the results.

D. CONCLUSION VALIDITY

Conclusion validity concerns whether the statistical analysis is accurate and whether the conclusions drawn from the data are valid. The study employs robust statistical techniques to analyze the performance of the SFNE model. However, the findings might be affected by the potential overfitting of the model to the specific datasets used. Regularization techniques and additional independent validation sets could help confirm that the model's performance is not overstated.

By acknowledging these threats and implementing strategies to mitigate them, the study strives to present a reliable and valid evaluation of the SFNE model. Future research should address these limitations by incorporating a broader range of datasets and performance metrics and validating the findings across different contexts and domains.

The structured approach ensures that the findings are critically evaluated and the conclusions are well-supported, enhancing the study's contribution to estimating software development effort.

VII. CONCLUSIONS AND FUTURE WORK

This study presents the Swarm Intelligence-Based Functional Link Fuzzy Neural Estimator (SFNE) for Software Development Effort Estimation (SDEE). The SFNE model integrates interval type-2 fuzzy logic systems (IT2FLS), active learning, and particle swarm optimization (PSO) to improve the accuracy and reliability of effort predictions. Empirical evaluations conducted on six real-world datasets from the PROMISE repository demonstrate that SFNE consistently outperforms traditional models, including PSO-FLANN, Active-FLANN-PSO, FLANN with BP learning, ANN with BP learning, and radial basis functions (RBFs). Key performance indicators, such as Mean Magnitude of Relative Error (MMRE), Median Magnitude of Relative Error (MdMRE), and Standardized Accuracy (SA), highlight the superiority of SFNE in delivering more accurate and consistent estimates. For example, the SFNE model achieved an MMRE of 2.87E-05 and a MdMRE of 7.1123E-09 on the DesharnaisL1 dataset, significantly outperforming the RBF model, which recorded an MMRE of 0.34391. Furthermore, SFNE's SA score reached 99.983% on the DesharnaisL3 dataset, while the RBF model only achieved 41.734% on the Albrecht dataset. These findings underscore SFNE's robustness and adaptability across diverse datasets, making it a valuable tool for accurate effort estimation in software development projects.

Future research will focus on refining the SFNE model further. One direction involves comparing the performance of different membership functions, such as Gaussian and triangular, to assess their impact on the accuracy of fuzzy logic systems. Additionally, optimizing the parameters α and β, which influence the generation of interval type-2 fuzzy sets, will be explored to enhance model performance further. Contemporary computational intelligence techniques, including deep learning and reinforcement learning, will be investigated for higher estimation accuracy and model robustness. These approaches could potentially improve the adaptive learning capabilities of SFNE. Furthermore, efforts will be directed toward developing automated tools and software based on the SFNE model, facilitating its practical adoption in the software industry for real-time effort estimation and decision-making.

REFERENCES

[1] F. P. Brooks Jr., "Three great challenges for half-century-old computer science," *J. ACM*, vol. 50, no. 1, pp. 25–26, Jan. 2003.
[2] S. Hastie and S. Wojewoda, "Standish Group 2015 Chaos Report-Q&A with Jennifer Lynch," Standish Group, 2015. [Online]. Available: www.standishgroup.com. [Accessed: 2016].
[3] B. W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1981.
[4] L. H. Putnam, "A general empirical solution to the macro software sizing and estimating problem," *IEEE Transactions on Software Engineering*, vol. 4, no. 4, pp. 345–361, 1978.
[5] T. R. Benala, S. Dehuri, and R. Mall, "Computational intelligence in software cost estimation: An emerging paradigm," *ACM SIGSOFT Software Engineering Notes*, vol. 37, no. 3, pp. 1–7, 2012.
[6] G. M. Méndez, O. Castillo, R. Colás, and H. Moreno, "Finishing mill strip gage setup and control by interval type-1 non-singleton type-2 fuzzy logic systems," *Applied Soft Computing*, vol. 24, pp. 900–911, 2014.
[7] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.
[8] J. M. Mendel, R. I. John, and F. Liu, "Interval type-2 fuzzy logic systems made simple," *IEEE Transactions on Fuzzy Systems*, vol. 14, no. 6, pp. 808–821, 2006.
[9] J. M. Mendel and X. Liu, "Simplified interval type-2 fuzzy logic systems," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 6, pp. 1056–1069, 2013.
[10] T. Nguyen, A. Khosravi, D. Creighton, and S. Nahavandi, "EEG signal classification for BCI applications by wavelets and interval type-2 fuzzy logic systems," *Expert Systems with Applications*, vol. 42, no. 9, pp. 4370–4380, June 2015.
[11] O. Castillo, P. Melin, J. Kacprzyk, and W. Pedrycz, "Type-2 fuzzy logic: theory and applications," in *IEEE International Conference on Granular Computing (GRC 2007)*, pp. 145–145, 2007.
[12] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Reading, MA, USA: Addison-Wesley, 1989.
[13] S. Dasgupta, "Two faces of active learning," *Theoretical Computer Science*, vol. 412, no. 19, pp. 1767–1781, 2011.
[14] E. Kocaguneli, T. Menzies, J. Keung, D. Cok, and R. Madachy, "Active learning and effort estimation: Finding the essential content of software effort estimation data," *IEEE Transactions on Software Engineering*, vol. 39, no. 8, pp. 1040–1053, Aug. 2012.
[15] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Nov. 1995.
[16] Z. Muzaffar and M. A. Ahmed, "Software development effort prediction: A study on the factors impacting the accuracy of fuzzy logic systems," *Information and Software Technology*, vol. 52, no. 1, pp. 92–109, 2010.
[17] M. A. Ahmed and Z. Muzaffar, "Handling imprecision and uncertainty in software development effort prediction: A type-2 fuzzy logic based framework," *Information and Software Technology*, vol. 51, no. 3, pp. 640–654, 2009.
[18] Z. Xu and T. M. Khoshgoftaar, "Identification of fuzzy models of software cost estimation," *Fuzzy Sets and Systems*, vol. 145, no. 1, pp. 141–163, 2004.
[19] M. Azzeh, D. Neagu, and P. Cowling, "Software effort estimation based on weighted fuzzy grey relational analysis," in *Proceedings of*

the 5th International Conference on Predictor Models in Software Engineering (PROMISE '09), Vancouver, BC, Canada, May 2009, Article no. 8, 10 pages.

[20] A. Sheta, "Software effort estimation and stock market prediction using Takagi-Sugeno fuzzy models," in *Proceedings of the IEEE International Conference on Fuzzy Systems* (FUZZ-IEEE 2006), Vancouver, BC, Canada, July 2006, pp. 171–178.

[21] J. Lee, W.-T. Lee, and J.-Y. Kuo, "Fuzzy logic as a basis for use case point estimation," in *Proceedings of the IEEE International Conference on Fuzzy Systems* (FUZZ-IEEE 2011), Taipei, Taiwan, June 2011, pp. 2702–2707.

[22] B. T. Rao, B. Sameet, G. K. Swathi, K. V. Gupta, C. Ravi Teja, and S. Sumana, "A novel neural network approach for software cost estimation using functional link artificial neural network (FLANN)," *International Journal of Computer Science and Network Security*, vol. 9, no. 6, pp. 126–131, 2009.

[23] B. T. Rao, S. Dehuri, and R. Mall, "Functional link artificial neural networks for software cost estimation," *International Journal of Applied Evolutionary Computation*, vol. 3, no. 2, pp. 62–82, 2012.

[24] T. R. Benala, K. Chinnababu, R. Mall, and S. Dehuri, "A particle swarm optimized functional link artificial neural network (PSO-FLANN) in software cost estimation," in *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)*, S. Satapathy, S. Udgata, and B. Biswal, Eds., Advances in Intelligent Systems and Computing, vol. 199. Berlin, Heidelberg: Springer, 2013, pp. 59–66. doi: 10.1007/978-3-642-35314-7_8.

[25] T. R. Benala, R. Mall, S. Dehuri, and P. Swetha, "Software effort estimation using functional link neural networks tuned with active learning and optimized with particle swarm optimization," in *Swarm, Evolutionary, and Memetic Computing: SEMCCO 2014*, B. Panigrahi, P. Suganthan, and S. Das, Eds., Lecture Notes in Computer Science, vol. 8947. Cham, Switzerland: Springer, 2015, pp. 223–238. doi: 10.1007/978-3-319-20294-5_20.

[26] P. Manchala and M. Bisi, "TSoptEE: Two-stage optimization technique for software development effort estimation," *Cluster Computing*, vol. 27, pp. 8889–8908, 2024. doi: 10.1007/s10586-024-04418-2.

[27] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414-1427, Nov./Dec. 1992.

[28] N. N. Karnik and J. M. Mendel, "Operations on type-2 fuzzy sets," *Fuzzy Sets and Systems*, vol. 122, no. 2, pp. 327-348, 2001.

[29] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens, "Data mining techniques for software effort estimation: A comparative study," *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 375–397, 2011.

[30] M. Shin and A. L. Goel, "Empirical data modelling in software engineering using radial basis functions," *IEEE Transactions on Software Engineering*, vol. 26, no. 6, pp. 567-576, 2000.

[31] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural Network Design*. Boston, MA, USA: PWS Publishing Co., 1997.

[32] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, pp. 131-164, 2009.

[33] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," Inf. Softw. Technol., vol. 54, no. 8, pp. 820–827, 2012.

[34] T. R. Benala and R. Mall, "DABE: Differential evolution in analogy-based software development effort estimation," Swarm and Evolutionary Computation, vol. 38, pp. 158–172, 2018.

AUTHOR'S BIOGRAPHY

Tirimula Rao Benala is an Assistant Professor in the Department of Information Technology at JNTU-GV College of Engineering, Vizianagaram, Andhra Pradesh, India. With a Ph.D. in Computer Science and Engineering from JNTU Kakinada, his research focuses on computational intelligence in software development effort estimation. He has published over 30 papers in reputable international journals and conferences, contributing significantly to machine learning, evolutionary computation, and software engineering. His professional memberships include Senior Member IEEE, Professional Member ACM, and Senior Life Member Computer Society of India. Dr. Benala has received several awards, including the Young Engineering of the Year Award and Best Teacher Award. He has been a visiting fellow at the Centre for Theoretical Studie, IIT Kharagpur, and has participated in the US-India 21st Century Knowledge Initiative grant at Chicago State University. His extensive teaching portfolio encompasses artificial intelligence, algorithms, cryptography, and software testing.

Dr. Anupama Kaushik is working as an associate professor at Maharaja Surajmal Institute of Technology, affiliated to Guru Gobind Singh Indraprastha University, New Delhi. She has 20+ years of teaching experience. She is an active member of IEEE. She has done PhD computer science from Indira Gandhi Delhi Technical University for Women and Sharda University. Her research areas are software engineering, neural networks, optimizations techniques and other soft computing techniques. She has over 30 research papers in international and national journals. She is the reviewer of many well reputed journals like IEEE Transactions on Software Engineering, Applied Soft Computing, Cluster Computing etc. She is very dedicated and committed towards her work.

Satchidananda Dehuri (SMIEEE) is working as a Professor in the Department of Computer Science (Erstwhile Department of Information and Communication Technology), Fakir Mohan University, Balasore, Odisha, India since 2013. Prior to this appointment, for a short stint (i.e., from Oct. 2012 to May 2014) he was an Associate Professor in the Department of Systems Engineering, Ajou University, South Korea. He received his M.Tech. and Ph.D. degrees in Computer Science from Utkal University, Vani Vihar, Odisha in 2001 and 2006, respectively. He visited as a BOYSCAST Fellow to the Soft Computing Laboratory, Yonsei University, Seoul, South Korea under the BOYSCAST Fellowship Program of DST, Govt. of India in 2008. In 2010 he received the Young Scientist Award in Engineering and Technology for the year 2008 from Odisha Vigyan Academy, Department of Science and Technology, Govt. of Odisha. In 2021 he received Teachers Associateship and Research Excellence (TARE) Fellowship from SERB, DST, Govt. of India for three years to carry out intensive research on Higher Order Neural Networks for Big Data Analysis at host Institute, ISI Kolkata and Parent Institute, Fakir Mohan University, Balasore. His research interests include Multi-objective Optimization, Machine Learning, and Data Science. He has already published 275 research papers in reputed journals and conference proceedings. Under his direct supervision, 20 PhD. Scholars have been successfully awarded in Computer Science. He has completed three different research projects obtained from DST, UGC, and DRDO. His h-index as per Google Scholar is more than 31. As a part of Academic Collaboration, he has visited Ireland, New Zealand, Hong Kong, France, South Korea, and Nepal.

APPENDIX A: ALGORITHMS

Algorithm 1:

```
$[\underline{\mu}_{s_1}, \overline{\mu}_{s_1}, \underline{\mu}_{s_2}, \overline{\mu}_{s_2}]$ = $\boldsymbol{Fuzzy}\ (\boldsymbol{NTD}[\mathbf{1} \dots \boldsymbol{N}.\mathbf{1} \dots \boldsymbol{M}])$

          For each $i = 1 \dots N$
                    For each j=1…M-1
                        $Lower(S_1[i,j]) = e^{-\left((x(i,j)-LowerMean(i,j,s=1))^2 / 2\sigma s_1(i,j)^2\right)}$
                        $Upper(S_1[i,j]) = e^{-\left((x(i,j)-UpperMean(i,j,s=1))^2 / 2\sigma s_1(i,j)^2\right)}$
                        $Lower(S_2[i,j]) = e^{-\left((x(i,j)-LowerMean(i,j,s=2))^2 / 2\sigma s_2(i,j)^2\right)}$
                        $Upper(S_2[i,j]) = e^{-\left((x(i,j)-UpperMean(i,j,s=2))^2 / 2\sigma s_2(i,j)^2\right)}$
                    End
          End
Step 4.7: Perform T-norm that is logical AND to the membership values. It is simply the minimum membership value.
          For each row $i = 1 \dots N$
                    For each column j=1…M-1
                              $\underline{\mu}_{s_1}(i,1) = Minimum(Lower(S_1[i,j]))$
                              $\overline{\overline{\mu}}_{s_1}(i,1) = Minimum(Upper(S_1[i,j]))$
                              $\underline{\mu}_{s_2}(i,1) = Minimum(Lower(S_2[i,j]))$
                              $\overline{\overline{\mu}}_{s_2}(i,1) = Minimum(Upper(S_2[i,j]))$
                    End
          End

Thus, we obtained the Interval Type-2 fuzzy set for each attribute.
Step 1: Using Mamadhani Fuzzy Inference system tool create a FIS and membership function as Gaussian function.
Step 2: Initially calculate mean and standard deviation as $Mean\ (NTD\ (1 \dots N, 1 \dots M])\ and std\ (NTD\ [1 \dots N, 1 \dots M])$.
Step 3: Update the Gaussian parameter standard deviation with calculated standard deviation values of $NTD\ [1 \dots N, 1 \dots M]$. % Crisp to Type 1 fuzzy set %
     $Membesrshipvalues[1 \dots N, 1 \dots M, 1 \dots k] = gaussmf\,(NTD(1 \dots N, 1 \dots M), [\sigma, c])$
     Step 4: Convert the Type-1 to Interval Type 2 fuzzy set using mean and standard deviation of membership values of type -1
     Step 4.1: $Type2Mean(1 \dots N, 1 \dots M-1) = Mean(Membesrshipvalues[1 \dots N, 1 \dots M-1, 1 \dots k])$
               $Type2std(1 \dots N, 1 \dots M-1) = std(Membershipvalues[1 \dots N, 1 \dots M-1, 1 \dots k])$
     Step 4.2: Randomly assign values between range [0, 1] to $\alpha[1 \dots N, 1 \dots M-1]$ and
          $\beta[1 \dots N, 1 \dots M-1]$.
     Step 4.3: Calculate $R(1,i) = sum(Type2std[1 \dots N, 1 \dots M-1])/N$
     Step 4.4: Divide each value of $Type2Mean(1 \dots N, 1 \dots M-1)$ into two segments $s_1$ and $s_2$ and calculate midpoint of each segment $Ms_1, Ms_2$
     Step 4.5: Calculate lower and upper means and standard deviation values for both segments $s_1$ and $s_2$ over the midpoints $Ms_1, Ms_2$
          For each row $i = 1 \dots N$
                    For each column $j = 1 \dots M-1$
                              For each segment $s = 1 \dots 2$
                                        $LowerMean(i,j,s=1) = Ms_1 - (\alpha(i,j) * R(1,j))$
                                        $UpperMean(i,j,s=1) = Ms_1 + (\alpha(i,j) * R(1,j))$
                                        $LowerMean(i,j,s=2) = Ms_2 - (\alpha(i,j) * R(1,j))$
                                        $UpperMean(i,j,s=2) = Ms_2 \pm (\alpha(i,j) * R(1,j))$
                                        $\sigma s_1(i,j) = \beta(i,j) * R(1,j)$
                                        $\sigma s_2(i,j) = \beta(i,j) * R(1,j)$
                                        End
                              End
                    End
     Step 4.6: Calculate lower and upper membership values for both segments for each attribute of every project in a dataset using Gaussian membership function.
```

Algorithm 2:

Let us assume, D is the dataset, ND is the normalized Dataset. TrainDataset and TestDataset are the training and testing parts respectively, P is the data point of reduced size, L and H are lower and higher dimensions respectively, and TC is the termination criteria, O is the output layer, W is the weighted sum, E is the error, current best fitness value is CF. Fitness value is F, GB represents global best, PV is the particle velocity.

```
Step 1: For each Dataset $D[1 \dots N, 1 \dots M]$
        Step 1.1: $ND[1 \dots N, 1 \dots M] = Normalization(D[1 \dots N, 1 \dots M])$
Step 2: Divide ND 2/3rd parts into TrainDataset and 1/3rd part into TestDataset.
Step 3: $\left[\underline{\mu}_{s_1}[1 \dots N, 1..M], \overline{\mu}_{s_1}[1 \dots N, 1..M], \underline{\mu}_{s_2}[1 \dots N, 1..M], \overline{\mu}_{s_2}[1 \dots N, 1..M]\right] = FIT(ND[1 \dots N, 1 \dots M])$
        Returns upper and lower membership values of segment1 ($s_1$) and segnent2 ($s_2$)
Step 3: For each TrainDataset
Step 3.1: Apply $Quick\_TrainDataset[1 \dots n-1, 1 \dots m] = Quick(TrainDataset[1 \dots N-1, 1 \dots M])$
Step 4: For each P
Step 4.1: Map from L to H.
Step 5: For each particle initialize with small values from [-1, 1].
Step 6: While (!TC)
        {
        Apply
    $FLANN\_PSO(Y, Output, c_1, c_2, r_1 r_2, velocity, position, \omega, pbest, gbest, Quick\_TrainDataset[1 \dots n - 1, 1 \dots m])$
        {
                For each swarm
                        {
                        For each particle in the swarm
                                {
                                For each sample in the training sample
                                        {
                                        Calculate W, and send it as input to O.
                                        The tuned values of Interval Type 2 fuzzy
                                        $\underline{\mu}_{s_1}[1 \dots N, 1..M], \overline{\mu}_{s_1}[1 \dots N, 1..M], \underline{\mu}_{s_2}[1 \dots N, 1..M]$,
                                        $\overline{\mu}_{s_2}[1 \dots N, 1..M]$ are converted to type 1 fuzzy Using

                                        outputs ($Y_1$ & $Y_2$) in output layer
                                        $Y_l = (\underline{\mu}_{s_1} \times Y_1 + \underline{\mu}_{s_2} \times Y_2) / (\underline{\mu}_{s_1} + \underline{\mu}_{s_2})$
                                        $Y_r = (\overline{\mu}_{s_1} \times Y_1 + \overline{\mu}_{s_2} \times Y_2) / (\overline{\mu}_{s_1} + \overline{\mu}_{s_2})$
Then again defuzzyfy the type 1 values to crisp set using Output= $\lambda \times Y_l + (1 - \lambda) \times Y_r$
                                        Calculate$E = Y - Output.$
                                        }
                                Assign E to F.
                                If (F is better than CF)
                                        {
                                        Assign F to CF
                                        }
                                Assign CF to GB
                                }
                        For each particle
                                {
                                Call Reduced () and Find PV.
                                Update Particle Positions
                                }
                        }
                }
        }
```

APPENDIX B: BUILDING INTERVAL TYPE-2 FL

Step 1 The mean of membership values of all the attributes of the dataset is calculated as follows:

Mean (1, 1) $= (\mu_{verylow} + \mu_{low} + \mu_{moderate} + \mu_{high} + \mu_{very\ high})/5$
$= (0.7980 + 0.9860 + 0.6018 + 0.1815 + 0.0270)/5$
$= 0.51886$

Step 2 The system calculates type-2 standard deviation of every attribute. The results of type-2 standard deviation of the first row and first column of the hypothetical dataset are as follows:

$$\begin{aligned}\mathrm{Type2std}(1,1) &= ((0.7980 - 0.51886)^2 \\ &+ (0.9860 - 0.51886)^2 \\ &+ (0.6018 - 0.51886)^2 \\ &+ (0.1815 - 0.51886)^2 \\ &+ \quad (0.0270 - 0.51886)^2) \\ &= (0.27914)^2 + (0.46714)^2 \\ &+ (0.08294)^2 + (-0.33736)^2 \\ &+ (-0.49186)^2 \\ &= (0.07791914 + 0.21821978 \\ &+ 0.006879044 + 0.11381177 \\ &+ 0.24192626) \\ &= 0.658755994/ \\ &\quad 5(number\ of\ linguistic\ variables) \\ &= 0.131751199\end{aligned}$$

Step 3 α and β are assigned random values between [0,1].

Step 4 The system obtains the R matrix by the following procedure:
$\mathrm{R}(1,1 \ldots 4) = \mathrm{sum}(\mathrm{Type2std}[1 \ldots 4,1 \ldots 4])/$ (number of rows) i.e.
R(1,1)=(Type2std(1,1)=0.1318+Type2std(1,2)+ Type2std(1,3)+Type2std(1,4))

Step 5 The type-2 mean is divided into equal interval of t segments. Here, we consider t = 2.
Range of Segment1= from 0 to ((0.51886)/2=0.25943)

Range of segment2= from 0.25943 to 0.51886

Step 6 To calculate the interval type-2 membership values, the mean and standard deviation of each segment is deducted as follows:

Mean:

Take midpoint of segment 1 and segment 2, that is,
Ms_1= (0+0.25943)/2=0.129715
Ms_2= (0.25943+0.51886)/2=0.389145

$$\begin{aligned}LowerMean(1,1,segment = 1) &= Ms_1 - (\alpha(1,1) * R(1,1)) \\ &= 0.1297 - (0.4 * R(1,1)) \\ UpperMean(1,1,segment = 1) &= Ms_1 + (\alpha(1,1) * R(1,1)) \\ &= 0.1297 + (0.6 * R(1,1) \\ LowerMean(1,1,segment = 2) &= Ms_2 - (\alpha(1,1) * R(1,1)) \\ &= 0.3892 - (0.1 * R(1,1)) \\ UpperMean(1,1,segment = 2) &= Ms_2 \pm (\alpha(1,1) * R(1,1)) \\ &= 0.3892 + (0.3 * R(1,1))\end{aligned}$$

Standard deviation:

$$\sigma segment_1(1,1) = \beta(1,1) * R(1,1) = 0.6 * R(1,1)$$
$$\sigma segment_2(1,1) = \beta(1,1) * R(1,1) = 0.7 * R(1,1)$$

$Lower_{s_1}[1 \ldots 4,1..4], Upper_{s_1}[1 \ldots 4,1..4], Lower_{s_2}[1 \ldots 4,1..4], Uppe$

Step 7 T-norm is computed for $Lower_{s_1}[1 \ldots 4,1..4]$, $Lower_{s_2}[1 \ldots 4,1..4]$, $Upper_{s_1}[1 \ldots 4,1..4]$, and $Upper_{s_2}[1 \ldots 4,1..4]$ as follows:

For example,

$$Lower_{s_1}(1 \ldots 4,1 \ldots 4) = \begin{bmatrix}0.0023 & 0.0036 & 0.0012 & 0.0032 \\ 0.0253 & 0.0085 & 0.0124 & 0.3211 \\ 0.1248 & 0.3214 & 0.0258 & 0.1289 \\ 0.0001 & 0.0014 & 0.0005 & 0.2587\end{bmatrix}$$

$$Lower_{s_2}(1 \ldots 4,1 \ldots 4) = \begin{bmatrix}0.0123 & 0.0026 & 0.012 & 0.02 \\ 0.0153 & 0.0055 & 0.0134 & 0.411 \\ 0.138 & 0.3144 & 0.0148 & 0.1089 \\ 0.0034 & 0.0214 & 0.0045 & 0.2657\end{bmatrix}$$

Then, T-norm of each matrix, that is, minimum for each row, is obtained as follows:

$$\underline{\mu}_{s_1}(i,1) = Minimum(Lower(S_1[i,j]))$$
$$\overline{\mu}_{s_1}(i,1) = Minimum(Upper(S_1[i,j]))$$
$$\underline{\mu}_{s_2}(i,1) = Minimum(Lower(S_2[i,j]))$$
$$\overline{\mu}_{s_2}(i,1) = Minimum(Upper(S_2[i,j]))$$

Hence, $\underline{\mu}_{s_1}(1,1) = \min(0.0023,0.0036,0.0012,0.0032) = 0.0012$

$\underline{\mu}_{s_1}(2,1) = \min(0.0253,0.0085,0.0124,0.3211) = 0.0085$

Step 8 The local outputs of FLANN, Y_1 and Y_2, are used to convert the interval type-2 fuzzy set output to type-1 fuzzy set output. Let us assume $Y_1 = 0.125$ and $Y_2 = 0.325$ after one iteration and for one instance.

$$Y_l = (\underline{\mu}_{s_1} \times Y_1 + \underline{\mu}_{s_2} \times Y_2) \Big/ (\underline{\mu}_{s_1} + \underline{\mu}_{s_2})$$
$$Y_r = (\overline{\mu}_{s_1} \times Y_1 + \overline{\mu}_{s_2} \times Y_2) \Big/ (\overline{\mu}_{s_1} + \overline{\mu}_{s_2})$$

Let us obtain the value of Y_l; Y_r value is obtained in a similar manner. For one instance of the dataset,

$$Y_l = (\underline{\mu}_{s_1}(1,1) \times Y_1 + \underline{\mu}_{s_2}(1,1) \times Y_2) \Big/ (\underline{\mu}_{s_1} + \underline{\mu}_{s_2})$$
$$= ((0.0012 \times 0.125) + (0.0026 \times 0.325)) \Big/ (0.0012 + 0.0026)$$
$$Y_l = 0.0038$$

Thus, after getting Y_l & Y_r, i.e., type-1 fuzzy set outputs, we defuzzify them into crisp set outputs by using the following formula:

$$\lambda \times Y_l + (1 - \lambda) \times Y_r$$

Here, λ is the defuzzify parameter, which takes any random number from the range [0, 1]. Thus, we get the final calculated output as:

$$\hat{Y} = \lambda \times Y_l + (1 - \lambda) \times Y_r$$

Step 9 The system calculates the error values, and weights are updated using the adaptive PSO technique reported by Benala et al. (2015).

Step 10 The quality measures, namely mean magnitude of relative error (MMRE), MdMRE, prediction (PRED), standardized accuracy (SA), Delta, etc., are obtained.