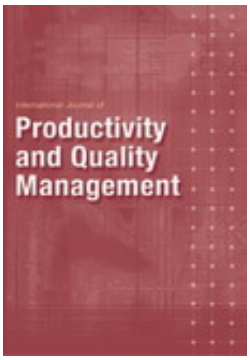
INDERSCIENCE PUBLISHERS
Linking academia, business and industry through research



International Journal of Productivity and Quality Management

ISSN online: 1746-6482 - ISSN print: 1746-6474
https://www.inderscience.com/ijpqm

Software effort estimation using cascade neural network optimised based on modified particle swarm optimisation (MPSO-CNN)

Mohammed Abdulmajeed Moharram, Saurabh Bilgaiyan, Santwana Sagnika

DOI: 10.1504/IJPQM.2023.10055552

Article History:

Received:	02 December 2022
Last revised:	10 March 2023
Accepted:	11 March 2023
Published online:	16 January 2025

Software effort estimation using cascade neural network optimised based on modified particle swarm optimisation (MPSO-CNN)

Mohammed Abdulmajeed Moharram*,
Saurabh Bilgaiyan and Santwana Sagnika

School of Computer Science and Engineering,
KIIT Deemed to be University,
Bhubaneswar, India
Email: MohammedAbdulmajeed1994@gmail.com
Email: saurabhbilgaiyan01@gmail.com
Email: santwana.sagnika@gmail.com
*Corresponding author

Abstract: Software effort estimation has a significant role in software development engineering. The inaccurate estimation will increase the failure possibilities of the project. On the contrary, accurate estimation enables the project developers to finalise the projects within the required time and budget. Furthermore, it is considered a big challenge to obtain the satisfactory accuracy of project development at the beginning. To tackle this problem, soft computing techniques such as artificial neural network (ANN) has already demonstrated a remarkable performance in software effort estimation. However, the optimal weights for the neural network are still considered a big dilemma. In this paper, a cascade neural network (CNN) is optimised based on modified particle swarm optimisation (PSO). The modified PSO can overcome the premature convergence of PSO as well as avoid falling into local optima effectively. The experimental results have shown the superiority of the proposed work compared with the standard PSO significantly.

Keywords: particle swarm optimisation; PSO; cascade neural network; CNN; Pearson correlation; standard deviation; effort estimation.

Reference to this paper should be made as follows: Moharram, M.A., Bilgaiyan, S. and Sagnika, S. (2025) 'Software effort estimation using cascade neural network optimised based on modified particle swarm optimisation (MPSO-CNN)', *Int. J. Productivity and Quality Management*, Vol. 44, No. 1, pp.40–70.

Biographical notes: Mohammed Abdulmajeed Moharram received his Bachelor's degree in Computer Science and Engineering from Aleppo University, Aleppo, Syria, in 2017, and MTech degree in Computer Science and Engineering from Kalinga Institute of Industrial Technology (KIIT), Orissa, India, in 2021. He is currently pursuing his PhD in the School of Computer Science and Engineering (SCOPE) at VIT-AP University, Amaravati, India. His reputed publications include research articles in peer-reviewed journals, namely *Environmental Science and Pollution Research*, and *Journal of Applied Remote Sensing*. His research interests include software effort estimation, swarm intelligence, neural networks, machine learning, deep learning, image processing, hyperspectral imaging, and remote sensing.

Saurabh Bilgaiyan is currently working as an Assistant Professor in KIIT Deemed to be University, Bhubaneswar, India since 2016. He has completed his Master's and PhD in Computer Science Engineering from KIIT Deemed to be University, Bhubaneswar, India in 2014 and 2018 respectively. He earned his Bachelor's degree of BE in Information Technology from B.I.R.T., Bhopal, India in 2012. He has published more than 50 research papers in various reputed international journals, conferences, and edited books. His area of interest includes soft computing, software engineering, cloud computing, image processing, and machine learning. He has reviewed various manuscripts in more than 25 international conferences and journals including *Soft Computing Springer*, *IEEE Access*, *IETE Journal of Research*, Taylor and Francis, *Future Generation Computer System*, Elsevier, *Concurrency* and *Computation: Practice and Experience*, Wiley, etc.

Santwana Sagnika received her PhD degree in Computer Science and Engineering from KIIT Deemed to be University in 2022. She is currently working as an Assistant Professor in School of Computer Engineering, KIIT Deemed to be University. Her research interests include machine learning, natural language processing, and soft computing.

1 Introduction

Software development is considered an essential approach for many enterprises due to the ever-rising significance of software project management effectively (Foy et al., 2019). The estimation factors mainly comprise cost, time, and effort required of the project (Thota et al., 2020). However, the delay of project delivery is considered a big dilemma due to the inaccurate estimation at the first phases of the project (Bhavsar et al., 2020). Many enterprises have been failed due to there being unsatisfactory results for software effort estimation (Saffar and Obeidat, 2020). Accurate estimates of resources have a critical role at the beginning of the project (Goswami et al., 2021). There are a lot of complicated procedures in software applications such as methodologies, methods, and software development tools. Therefore, the development and improvement process of software projects at the beginning is considered a major challenge. For many companies, the main aim represents by completing the software project within a short time, reasonable cost, and high-quality (Minku, 2019). The effort estimation that has been performed at the early phase is not eligible to be adequate till the final phase. Therefore, project estimates should be updated during all phases of software development to avoid overestimation or underestimation problems (Azzeh et al., 2010). Sometimes there are some aspects that may lead to inaccurate estimates such as the little information about influencing factors and hazards that may arise, the stress from the customers or the management side, and the estimation methods that depend on expert judgment. Consequently, all the previous issues will influence delivering a project within a particular time, budget (Agrawal and Chari, 2007), and good quality (Suri and Ranjan, 2012). Many approaches have been conducted using various effort estimation methods like expert judgment, algorithmic method, and analogy based method, and constructive cost model (COCOMO) is the most popular among them (Andrade et al., 2022). However, many problems have been raised according to these methods in the scope of software effort estimation (Dejaeger et al., 2011). Therefore, many researchers have

investigated various techniques to improve the estimation accuracy as well as ensure the project budget (Alsaadi and Saeedi, 2022). There are two major models for effort estimation, algorithmic and non-algorithmic models. Non-algorithmic models are characterised by high performance and good accuracy compared with algorithmic models due to depending on machine learning algorithms (Mahmood et al., 2022). Non-algorithmic models are based on the historical data and build systems based on learning from data (Attarzadeh and Ow, 2011). However, multiple machine learning approaches have been demonstrated significant improvement for the effort prediction (Huang et al., 2006; Finnie et al., 1997). Where the effort estimation has been accomplished by using a neural network (NN) and the results have shown good accuracy (Azath et al., 2018). In the past few years, NN is incorporated with many techniques to improve the overall performance such as genetic algorithm (GA), fuzzy system, and evolutionary models (Boehm et al., 2000). Several approaches could be employed to train the NNs and obtain the optimal weights which in turn enhance the performance for effort estimation such as particle swarm optimisation (PSO) (Zhang et al., 2000), GA (Angeline et al., 1994). In most cases, the performance of the hybrid methods is better than the performance of the non-hybrid methods (Okumus and Dinler, 2016). NN has been optimised by PSO and GA effectively, where the performance of PSO-NN is mostly better than GA-NN and artificial neural network (ANN), with a minor error. Besides that, the parameter of PSO-NN can be specified in a short time compared with the parameter of GA which is more complex. The major challenge for enhancing PSO is how to improve the convergence speed toward the optimal solution, increase the population diversity to speed up reaching the global solution effectively, and avoid being stuck into local optima (Nagra et al., 2019). Nowadays, many studies have been performed to enhance the accuracy of effort estimation but still many techniques have not been applied yet. Therefore, the main contributions of this study can be summarised as follows:

1 Overcoming premature convergence which is considered a big dilemma for most optimisation algorithms using modified PSO.

2 The proposed method comprises the information according to the worst and best particles which in turn improves the exploitation and exploration process within the search space.

3 The effort estimation process has been performed using a cascade neural network (CNN) which is optimised by modified PSO.

4 In this study, eight public datasets have been performed for software effort estimation, and the results have demonstrated a significant improvement in the performance convergence of PSO.

The manuscript has been arranged in the following sections: Section 2 has demonstrated related work, and then CNN has been presented in Section 3. Section 4 was a brief overview of PSO followed by the methodology in Section 5. Also, several evaluation criteria have been described in Section 6. Finally, experimental results have been shown in Section 7 with the conclusion and future works in Section 8 and Section 9, respectively.

Table 1 Table of abbreviations

Abbreviations	*Definitions*
PSO	Particle swarm optimisation
MPSO	Modified particle swarm optimisation
CNN	Cascade neural network
COCOMO	Constructive cost model
ANN	Artificial neural network
GA	Genetic algorithm
	The square of the multiple correlation coefficients
RMSSE	Root mean square error
MAPE	Mean absolute percentage error
MLP	Multilayer perceptron
GRNN	General regression neural network
RBFNN	Radial basis function neural network
CCNN	Cascade correlation neural network
ANFIS	Adaptive neuro fuzzy inference system
SVR	Support vector regression
MBRE	Mean balanced relative error
MIBRE	Mean inverted balanced relative error
SA	Standardised accuracy
FLANN	Functional link artificial neural network
WOA	Whale optimisation algorithm
DEP	Dilation-erosion perceptron
MGA	Modified genetic algorithm
PCA	Principal component analysis
MRLHD	Morphological-rank-linear hybrid design
CMPSO	Chaotically modified particle swarm optimisation
WCO	Whale crow optimisation
SBO	Satin bower-bird optimisation
FA	Firefly algorithm
SS	Scatter search
IFCM	Intuitionistic fuzzy c-means
AIS	Artificial immune system

2 Related work

Many studies have been conducted to improve the performance of effort estimation at the beginning. Soft computing approaches have been utilised widely in the scope of effort estimation such as a NN, fuzzy logic, random forest, regression analysis, and Bayesian network. Here, some of the best-related work is given as follows: Hammad and Alqaddoumi (2018) have introduced four machine learning methods to predict the effort

at the beginning phases; where the authors have employed ANN, support vector machines s (SVM), Kstar, and linear regression (LR). Shukla and Kumar (2019) have applied Pearson correlation on the dataset to discover the best variables and then have applied machine learning algorithms: LR, SVM, K-nearest neighbour (knn), and multi-layer perceptron neural network (MLPNN). The results have exhibited that the performance of MLPNN is the best one compared with the others algorithm. Shukla et al. (2019) have implemented different NN techniques' where MLPNN, Ridge-MLPNN, Lasso-MLPNN, Bagging-MLPNN, and AdaBoost-MLPNN models have been implemented. Goyal and Bhatia (2019) have used a nonlinear technique to predicate the effort using MLPNN. The authors have compared the nonlinear technique with the LR technique. The results have shown that the LR technique is worse than the nonlinear technique. Tariq et al. (2020) have described many techniques related to data preprocessing that may improve effort estimation, where the features selection technique has been used via the ReliefF algorithm as well as regression and M5P model tree have been applied. Shah et al. (2020) have used the analogy-based estimation model with artificial bee colony (BABE) to improve effort prediction. The function of an artificial bee colony (ABC) is to build the different weights that will be employed in the training phase of analogy-based estimation (ABE). Fadhil et al. (2020) have used a hybrid dolphin and bat algorithm (DolBat) to effectively improve cost estimation. Kumar et al. (2008) have presented a wavelet neural network (WNN) for effort estimation. The authors have exploited the Morlet function and Gaussian function as activation functions of WNN. Functional link artificial neural network (FLANN) with three types (Chebyshev-FLANN, Legendre-FLANN, and power series-FLANN) has proposed by Rao et al. (2009). Satapathy et al. (2016) have utilised the use case point (UCP) using random forest (RF) to obtain the best accuracy. The outcomes have demonstrated that the proposed work has minor errors compared with other techniques. Nassif et al. (2019) have introduced three fuzzy logic approaches for effort estimation. The results have shown that Sugeno with the linear output has outperformed the other models. Yadav and Singh (2014) have predicted the effort estimation using ANN with one hidden layer that is optimised by a GA. Standish Group has accomplished a statistical analysis during the last few years about the proportion of success and failure of projects. In 2015, it published a new report related to the period 2011–2015 with a new group of characteristics to measure the project success depending on the data from more than 10.000 software projects. The results in Table 2 show that in 2015, 29% of software projects have completed on-time, on-budget, and on-scope, besides that 19% of software projects have been stopped before completion, and 52% of projects have taken more budget, scope, or time (Avlijaš, 2020). The latest report from Standish Group was CHAOS 2020, which describes three essential factors (good sponsor, good team, and good place) that will affect the success of software projects. These are the only factors we should improve, which in turn will improve the project performance. Modern measurement in 2020 was 31% of successful software projects, 19% of failed software projects, and 50% of challenging software projects.

Table 2 Standish Group report 2015

	2011	*2012*	*2013*	*2014*	*2015*
Successful	29%	27%	31%	28%	29%
Challenged	49%	56%	50%	55%	52%
Failed	22%	17%	19%	17%	19%

Table 3 Comparative analysis of effort estimation using ANN

Author	*Methodology*	*Evaluation criteria*
de Barcelos Tronto et al. (2008)	ANN	MMRE
Kumar et al. (2008)	Wavelet neural network	MMRE, PRED (25), and MdMRE
Kaur et al. (2010)	ANN	MMRE, and RMSSE
Nassif et al. (2012)	Cascade correlation neural network	MMER and PRED
Hota et al. (2015)	Tuned artificial neural network	MAE, MAPE, MSE and RMSE
Rijwani and Jain (2016)	Multi-layer feed forward neural network	MSE, and MMRE
Azzeh et al. (2018)	MLP, RBFNN, GRNN and CCNN, ANFIS, and SVR	SA, effect size, MAE, MBRE, MIBRE
Kaushik et al. (2022)	FLANN-WOA and RBFN-WOA	MMRE, MdMRE, and PRED (25)

3 Cascade neural network

ANN is inspired by the human brain, which contains many nodes connected with links that interact with each other. However, the structure of ANN comprises three main layers; the input layer, the hidden layer, and the output layer. Data is treated through the network from the input layer with few calculation operations till the output layer. The output of the neuron is called an activation value. The main kinds of NN design are cascade feed-forward and feed-forward. In a feed-forward network, the error passes from the output layer to the input layer, whereas the other operations are calculated from the input to output layer (Mellit and Pavan, 2010). Both the cascade forward network and the feed-forward network utilise the backpropagation algorithm, but the difference between them is the connection of neurons in every layer. All neurons of a layer are interconnected in the cascade forward network. Both cascade network and feed-forward backpropagation network perform the Backpropagation algorithm to calculate the new weights (Goyal and Goyal, 2011; Badde et al., 2013). Cascade forward network can be referred to as Figure 1. In the cascade network, there is a weight connection from the input layer into all layers and from every layer into the consecutive layers. The extra connections may improve the learning capability. The most common transfer function used is tan-sigmoid/log-sigmoid for the hidden layers, whereas a pure linear procedure is used for the output layer (Dhanaseely et al., 2012). Figure 2 represents the illustration of the feed-forward network, and Figure 3 represents the illustration of the cascade forward network.

Mathematical equations of the structure in Figure 2 can be described as follows in equations (1) and (2) (Warsito et al., 2018):

$$y_i = f^h\left(w_j^b + \sum_{i=1}^{n} w_{ji}^h * x_i\right) \tag{1}$$

where x_i represents the input of the NN, w_{ji}^{h} represents the weights between the input and the hidden layers, w_j^b represents the bias applied on the hidden layers, y_i represents the output of neuron i after applying the activation function.

$$\text{Output} = f^{\text{out}}\left(w^b + \sum_{j=1}^{k} w_j^o * y_j\right) \tag{2}$$

where w_j^o represents the weights between the hidden and the output layers, w^b represents the bias applied on the output layers, the output represents the output of the entire NN after applying the activation function. Also, the mathematical equation of the structure in Figure 3 can be described as follows in equation (3) (Warsito et al., 2018):

$$\text{Output} = \sum_{i=1}^{n} f^{in} * w_i^{in} * x_i + f^{\text{out}}\left(w^b + \sum_{j=1}^{k} w_j^o * y_j\right) \tag{3}$$

where f^{in} represents the activation function of the connection between the input and output layer, w_i^{in} represents the weights between the input and the output layer. The parameters of CNN can be described as in Table 4.

Figure 1 Cascade forward network (see online version for colours)

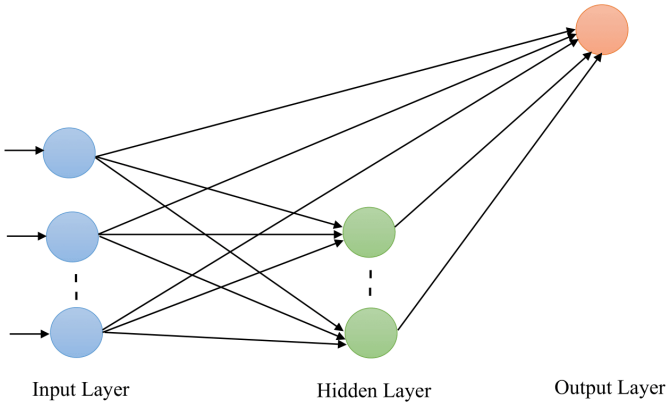


Figure 2 Architecture of feed forward network (see online version for colours)

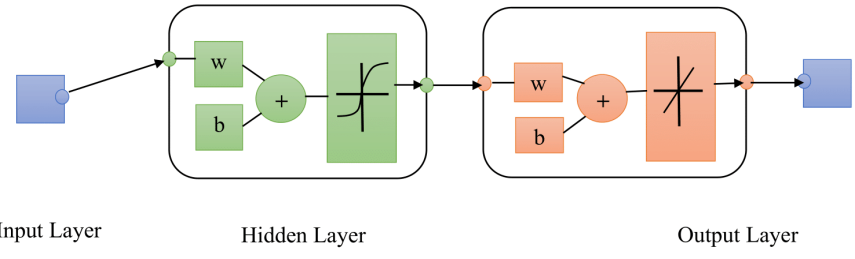


Figure 3 Architecture of cascade forward network (see online version for colours)

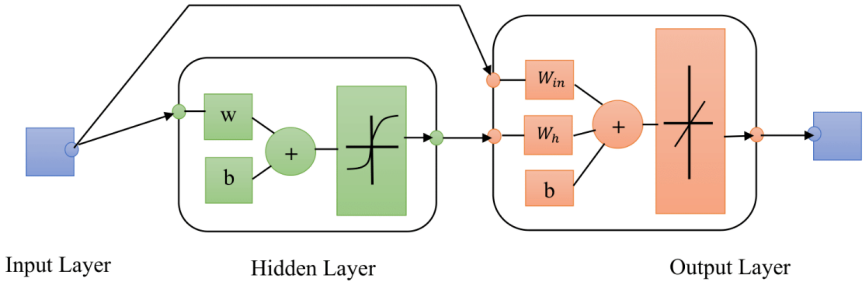


Table 4 Parameters of CNN

Layers	*Value*
Input layers	Features selected from each dataset separately
Hidden layers	Seven neurons in the hidden layer
Output layers	Effort predicted

The CNN is considered in this work because of its many advantages. It does not require a predefined structure and is self-organising, thereby growing from the training data dynamically. Further, since each neuron gets trained independently, it learns faster than most other NN architectures. It is also representative of the nonlinear relationship existing between the input and output data (Warsito et al., 2018).

4 Particle swarm optimisation

PSO was presented in 1995 by Dr James Kennedy and Dr Eberhart. PSO depends on swarm intelligence which can find the optimal solution to many optimisation problems. PSO imitate bird flocking and fish schooling behaviour. PSO considered as a local search and global search algorithm. Therefore, each particle will select its position based on its own experience; besides that, all particles experience (Yadav, 2019). PSO is considered an evolutionary stochastic optimisation technique that has been implemented successfully in many applications (Lodhi et al., 2018). Many studies have conducted using PSO to optimise the effort estimation process. Lin and Tzeng (2010), Sheta et al. (2010), and Hari and Reddy (2011) have used PSO to optimise the COCOMO model. PSO is a type of swarm intelligence that seeks to find the optimal solution through search space by sharing the best searching experiments between particles. Each particle in the swarm is considered a potential solution. In the beginning, the solutions (particles) generated randomly. The solutions have been updated according to the equation of velocity (4) and position (5) at each cycle (Qiu et al., 2018).

$$v_{new}(i,j) = w * v(i,j) + c1 * r1 * \left(P_{best(i,j)} - x(i,j)\right) + c2 * r2 * \left(G_{best(1,j)} - x(i,j)\right) \quad (4)$$

$$x_{new}(i,j) = x(i,j) + v_{new}(i,j) \quad (5)$$

where v represents the particle's velocity. x represents the particle's position. w represents inertial weight. c1, c2 represent the acceleration coefficients. r1, r2 represent a

random number between [0 1]. P_{best} represents the best individual experiment for each particle in the swarm. G_{best} represents the best experiment for the entire particles in the swarm.

Steps of PSO-CNN

Firstly, the weights and biases will be initialised. Then, PSO will be applied to get the optimal weights and biases for the CNN. Steps 1 into 4) will repeat until the termination criteria are satisfied. Two metrics for termination criteria have been used: when the maximum iteration is met or mean squared error (MSE < 0.0001).

Step 1 Initialisation process of the PSO parameters (x, v, P_{best}, and G_{best}).

Step 2 Calculate the particle's position and particle's velocity as in equations (4), and (5), respectively.

Step 3 Calculate the fitness function (MSE) for each particle through a CNN.

Step 4 Update P_{best} according to the own particle experience and G_{best} according to the experiences of all particles.

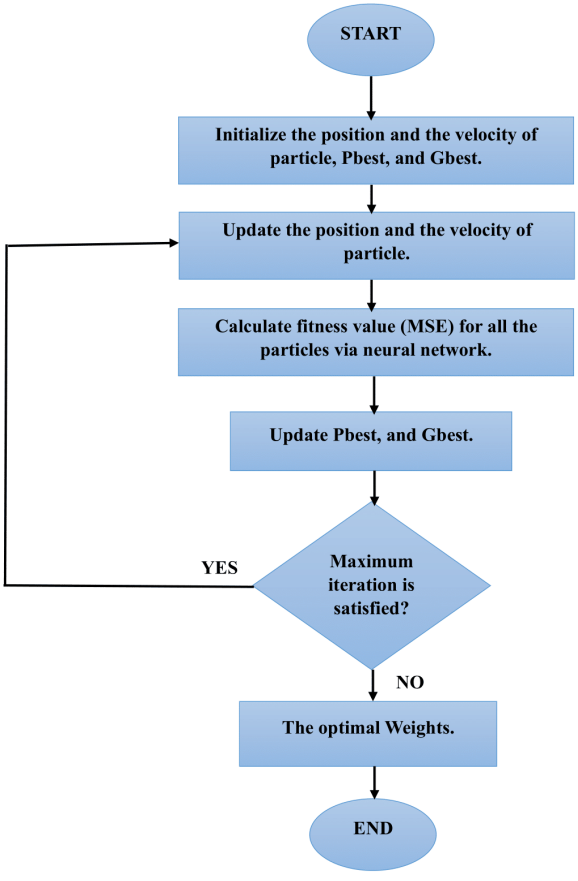
Step 5 Use G_{best} (optimal weights and biases) for effort estimation using a CNN.

The flowchart of PSO-CNN can be described as in Figure 4.

Algorithm 1: Pseudo code for PSO-CNN

```
FOR each particle
    Initialize x, v, Pbest
END FOR
FOR all swarm
    Initialize Gbest
END FOR
Initialize w, c1, c2
WHILE Max-Iteration OR MSE < 0.0001
    FOR each particle
        Find v according to equation /4/
        Find x according to equation /5/
    END FOR
    FOR each particle
        Find the fitness function for all the particles
        IF the current fitness value is better than previous one
            Set the current value as the new Pbest
        END IF
    END FOR
    Find Gbest
END WHILE
```

Figure 4 Flowchart of PSO-CNN (see online version for colours)



5 Methodology

Firstly, normalising the dataset has been done that will help to improve the learning process, then features selection for the best subset out of all the features. Moreover, the proposed work has been implemented for eight public datasets.

5.1 Normalisation

The normalisation is applied on the raw data to rearrange the values in a particular range that will help to accelerate the learning process. The data will be scaled between two values; upper and lower values. As the following equation (6):

$$x_{norm} = (b-a) * \left(\frac{x_i - \min}{\max - \min} \right) + a \tag{6}$$

where x_i represents the non-normalised data value. min, max describe the minimum and the maximum value of each feature, respectively. a, b represents the lower and upper bound of scaled data, respectively (Singh and Singh, 2020).

5.2 Feature selection

It is implemented based on selecting the most essential sub-features out of all features. Pearson correlation has been used for selecting the most relevant features. The range of Pearson correlation between [–1 1], where –1, 1 represent the strong negative correlation and the strong positive correlation, respectively, and 0 means no linear correlation. The used correlation coefficient of the two values x and y can be considered as follows in equation (7) (Rangkuti et al., 2018):

$$r_{x,y} = \frac{n\left(\sum xy\right) - \left(\sum x\right)\left(\sum y\right)}{\left[n\left(\sum x\right)^2 - \left(\sum x\right)^2\right]\left[n\left(\sum y\right)^2 - \left(\sum y\right)^2\right]} \tag{7}$$

5.3 Modified PSO (MPSO)

Many complicated engineering problems have been implemented by standard PSO. However, there is a shortcoming in the performance of PSO such as premature convergence, poor accuracy, falling into local optima (Wu and Wang, 2022). Also, many studies have been performed to enhance the performance of PSO by adding extra parameters of the velocity equation (Lazzus et al., 2020; Hacibeyoglu and Ibrahim, 2018). Lazzus et al. (2020) have added a new behaviour into the swarm by adding the fourth parameter represents the difference between the current particle and the randomly selected particle in the swarm that will improve the search capability and avoids the falling into local optima. (Hacibeyoglu and Ibrahim, 2018), have presented modified PSO by adding multi-swarm optimisation (MSO), especially for nonlinear continuous optimisation problems, to overcome the problems with high dimensionality in a short time. They have presented a new behaviour for particles in the swarm by adding the mean of the best individual experience of all particles of that swarm and the best solution of all swarms. That will improve the local search and add new information to each particle. In the proposed work, the extra parameter is added into the velocity equation, which is based on the local and global search strategies due to utilising the worst and best experiences of particles in the swarm. That will lead to overcoming premature convergence and reaching the global optimal solution within search space effectively compared with the standard PSO. The modified PSO also avoids falling into local optima. Hence, this work uses the modified PSO for a better and more accurate performance instead of other variants. The modified equation of velocity can be described as follows (8):

$$\begin{aligned} v_{new}(i,j) = {} & w * v(i,j) + c1 * r1 * \left(P_{best(i,j)} - x(i,j)\right) \\ & + c2 * r2 * \left(G_{best}(1,j) - x(i,j)\right) + c3 \\ & * \frac{\left(\text{mean}\left(G_{best(1,j)}\right) - x(i,j)\right)}{S\left(P_{worst(i,j)}\right) - S\left(P_{best(i,j)}\right)} \end{aligned} \tag{8}$$

where c3 represent the acceleration coefficients. $\frac{(\text{mean}(G_{\text{best}(1,j)}) - x(i,j)}{S(P_{\text{worst}(i,j)}) - S(P_{\text{best}(i,j)})}$ the numerator represents the difference between the mean of values of the global best particle (optimal weights and biases) in the swarm and particle position; and the denominator represents the difference between the standard deviation of the worst experiment and the standard deviation of the best experiment of every particle. The parameters of modified PSO can be described as in Table 5.

Table 5 Parameters of MPSO

Parameter	*Value*
Position bounds	[–1.5, 1.5]
Velocity bounds	[–1.5, 1.5]
Population	40
Iteration	800
C1	0.35
C2	0.75
C3	0.005
W	0.5

Algorithm 2: Pseudo Code for MPSO-CNN

```
FOR each particle do
    Initialize x, v, Pbest, Pworst
END FOR
FOR all swarm
    Initialize Gbest
END FOR
Initialize w, c1, c2, c3
WHILE Max-Iteration OR MSE < 0.0001
    FOR each particle
        Find v according to equation /8/
        Find x according to equation /5/
    END FOR
    FOR each particle
        Find the fitness function for all the particles
        IF the current fitness value is better than previous one
            Set the current value as the new Pbest
        END IF
        IF the current fitness value is worse than previous one
            Set the current value as the new Pworst
        END IF
    END FOR
```

Find Gbest
END WHILE

6 Evaluation criteria

Several assessment criteria were used to assess the performance of the proposed method for eight public datasets. Moreover, based on rigorous literature analysis, mean magnitude of relative error (MMRE) and PRED (25) evaluation criteria were selected for the comparative analysis among existing works (IR et al., 2022). These are the most commonly used measures as found in the literature. Other performance evaluation criteria such as MSE is not selected because of the inability for an end measure for comparison among different prediction models. Although, MSE can be used during the training process for driving the prediction model (Rhmann, 2021).

6.1 The magnitude of relative error

The magnitude of relative error (MRE) represents the difference between the actual effort and the predicted effort. MRE can be described in equation (9) as follows:

$$\mathrm{MRE_i} = \frac{\text{Actual Effort}_i - \text{Predicted Effort}_i}{\text{Actual Effort}_i} \tag{9}$$

6.2 Mean magnitude of relative error

MMRE represents the average of MRE. MMRE can be described in equation (10) as follows:

$$\mathrm{MMRE} = \frac{1}{n} * \sum_{i=1}^{n} \mathrm{MRE_i} \tag{10}$$

6.3 PRED

PRED represents the average proportion of estimates that is within *W* % of the actual values. PRED can be described in equation (11) as follows:

$$\mathrm{PRED(W)} = \frac{100}{n} * \sum_{n=1}^{n} \begin{cases} 1 & \Rightarrow \quad \text{if } \mathrm{MRE_i} = \dfrac{W}{100} \\ 0 & \Rightarrow \quad \text{Otherwise} \end{cases} \tag{11}$$

6.4 Median of magnitude of relative error (MdMRE)

MdMRE represents the median of Magnitude of Relative Error values. MdMRE can be described in equation (12) as follows:

$$\mathrm{MdMRE} = \text{median}\,(\mathrm{MRE_1}, \mathrm{MRE_2}, ..., \mathrm{MRE_n}) \tag{12}$$

The following evaluation criteria will use to evaluate the performance of MPSO-CNN. Five statistical measures were used for the performance evaluation (Al Asheeri and Hammad, 2019).

6.5 Mean absolute error

Mean absolute error (MAE) represents the average of the absolute value of the difference between the actual effort and the predicted effort. MAE can be described in equation (13) as follows:

$$\text{MAE} = \frac{1}{\text{n}} * \sum_{\text{i=1}}^{\text{n}} (\text{Predicted Effort}_{\text{i}} - \text{Actual Effort}_{\text{i}}) \tag{13}$$

6.6 Root mean squared error

Root mean squared error (RMSE) represents the square root of the average of the difference square between the actual effort and the predicted effort. RMSE can be described in equation (14) as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{\text{n=1}}^{\text{n}} \left(\text{Predicted Effort}_{\text{i}} - \text{Actual Effort}_{\text{i}}\right)^2}{\text{n}}} \tag{14}$$

6.7 Relative absolute error

Relative absolute error (RAE) represents the sum of the absolute value of the difference between the actual effort and the predicted effort divided into the sum of the absolute value of the difference between the actual effort and the mean of the actual effort values. RAE can be described in equation (15) as follows:

$$\text{RAE} = \frac{\sum_{\text{i=1}}^{\text{n}} \left(\text{Predicted Effort}_{\text{i}} - \text{Actual Effort}_{\text{i}}\right)}{\sum_{\text{i=1}}^{\text{n}} \text{Actual Effort}_{\text{i}} - \text{mean}(\text{Actual Effort}_{\text{i}})} \tag{15}$$

6.8 Root relative squared error (RRSE)

Root relative squared error (RRSE) represents the square root of the sum of the difference square between the actual effort and the predicted effort divided into the sum of the difference square between the actual effort and the mean of the actual effort values. RRSE can be described in equation (16) as follows:

$$\text{RRSE} = \frac{\sum_{\text{1}}^{\text{n}} (\text{Predicted Effort}_{\text{i}} - \text{Actual Effort}_{\text{i}})^2}{\sum_{\text{i}}^{\text{n}} \left(\text{Actual Effort}_{\text{i}} - \text{mean}(\text{Actual Effort}_{\text{i}})^2\right)} \tag{16}$$

6.9 Mean squared error

MSE represents the average of the difference square between the actual effort and the predicted effort. MSE can be described in equation (17) as follows:

$$\text{MSE} = \sum_{i=1}^{n} \frac{(\text{Predicted Effort}_i - \text{Actual Effort}_i)^2}{n} \tag{17}$$

7 Experimental results

Evaluation results for eight datasets will be shown in this section, several measures of the performance evaluation have been used: MMRE, MdMRE, MAE, RMSE, RAE, RRSE, MSE, PRED (25), and PRED (100). The results have been done using CNN optimised based on modified particle swarm optimisation (MPSO-CNN). The values have been averaged over ten iterations, to account for deviations. The comparison has been implemented according to different studies for each dataset separately, as obtained from existing literature.

7.1 Desharnais dataset

Desharnais dataset contains 81 projects described by 11 attributes, nine independent and two dependents (Araújo et al., 2012a). The evaluation results for the Desharnais dataset as follows in Table 6. Table 7 represents the comparison with other models for the Desharnais dataset, where MMRE only has been considered for WCO + linear regression and WCO + Kerenel Regression as MMRE is the most suitable metric for these methods. The results have shown that proposed MPSO-CNN has outperformed the other models.

Table 6 Evaluation results of the Desharnais dataset using MPSO-CNN

Iteration	*MMRE*	*MdMRE*	*MAE*	*RMSE*	*RAE*	*RRSE*	*MSE*	*PRED (25)*
200	0.0554	0.0422	0.0703	0.0976	0.5199	0.5201	0.0095	98.7654
400	0.0521	0.0410	0.0655	0.0897	0.4844	0.4776	0.0080	100
800	0.0519	0.0415	0.0646	0.0876	0.4780	0.4669	0.0077	100

Table 7 Comparison with other models for the Desharnais dataset

Model	*MMRE*	*PRED (25)*
Proposed MPSO-CNN	0.0519	100
DEP(MGA) (Araújo et al., 2012a)	0.0829	90.00
ANN-PSO-PCA-GA (Bisi and Goyal, 2016)	0.2589	88.89
MRLHD (Araújo et al., 2012b)	0.0981	90.00
DEP-CMPSO (Bilgaiyan et al., 2018)	0.0811	90.27
WCO + linear regression (Rhmann, 2021)	0.0733	-
WCO + Kernel Regression (Rhmann, 2021)	0.1468	-

7.2 Albrecht dataset

Albrecht dataset contains 24 projects classified into three groups, where eighteen projects are written in COBOL language, four are written in PL1 language, and two are written in DMS language. Besides, it includes five independent attributes: Inpcount, Outcount, Quecount, Filcount, and SLOC (Oliveira et al., 2010). The evaluation results for the Albrecht dataset as follows in Table 8. Table 9 represents the comparison with other models for the Albrecht dataset. The results using an MPSO-CNN have been outperformed the other models.

Table 8 Evaluation results of the Albrecht dataset using MPSO-CNN

Iteration	*MMRE*	*MdMRE*	*MAE*	*RMSE*	*RAE*	*RRSE*	*MSE*	*PRED (25)*
200	0.0279	0.0213	0.0320	0.0420	0.1777	0.1581	0.0018	100
400	0.0274	0.0202	0.0311	0.0381	0.1725	0.1435	0.0015	100
800	0.0267	0.0262	0.0304	0.0380	0.1687	0.1429	0.0014	100

Table 9 Comparison with other models for the Albrecht dataset

Model	*MMRE*	*PRED (25)*
Proposed MPSO-CNN	0.0267	100
ANN-PSO-PCA-GA (Bisi and Goyal, 2016)	0.3061	79.17
ANFIS-SBO (Moosavi and Bardsiri, 2017)	0.235	62.5
MRLHD (Araújo et al., 2012b)	0.3810	75.00
DEP(MGA) (Araújo et al., 2012a)	0.3512	75.00
DEP-CMPSO (Bilgaiyan et al., 2018)	0.3211	78.00

7.3 Nasa93 dataset

Nasa93 dataset was gathered by NASA from five development positions followed to it. The whole dataset contains 93 projects for the period between 1971 and 1987 described with 24 features. The evaluation results for Nasa93 dataset as follows in Table 10. Table 11 represents the comparison with other models for the Nasa93 dataset. The results using an MPSO-CNN have been outperformed the other models.

Table 10 Evaluation results of the Nasa93 dataset using MPSO-CNN

Iteration	*MMRE*	*MdMRE*	*MAE*	*RMSE*	*RAE*	*RRSE*	*MSE*	*PRED (25)*
200	0.0453	0.0355	0.0515	0.0817	0.6604	0.5930	0.0067	98.9247
400	0.0404	0.0274	0.0464	0.0778	0.5949	0.5647	0.0060	98.9247
800	0.0398	0.0250	0.0455	0.0744	0.5838	0.5402	0.0055	100

7.4 Nasa60 dataset

Nasa60 dataset contains the information for 60 projects described with 17 features. The evaluation results for Nasa60 dataset as follows in Table 12. Table 13 represents the comparison with other models for the Nasa60 dataset, where the value of PRED (25) has

been assigned approximately for hybrid model (AIS-GA). The results using an MPSO-CNN have been outperformed the other models.

Table 11 Comparison with other models for the Nasa93 dataset

Model	*MMRE*	*PRED (25)*
Proposed MPSO-CNN	0.0398	100
Dolphin Bat algorithm (Fadhil et al., 2020)	0.5027	25.806
Optimal FA (Resmi et al., 2019)	0.0562	88.25
Hybrid Model (GA -FA) (Maleki et al., 2014b)	0.2253	88.17
Hybrid Model (SS-GA) (Maleki et al., 2014a)	0.2385	87.09
FA-FLANN-IFCM (Kaushik et al., 2016b)	0.17	95
FA-RBFN-IFCM (Kaushik et al., 2016b)	0.14	90

Table 12 Evaluation results of the Nasa60 dataset using MPSO-CNN

Iteration	*MMRE*	*MdMRE*	*MAE*	*RMSE*	*RAE*	*RRSE*	*MSE*	*PRED (25)*
200	0.0344	0.0215	0.0414	0.0663	0.3182	0.3288	0.0044	100
400	0.0329	0.0183	0.0399	0.0629	0.3065	0.3122	0.0040	100
800	0.0248	0.0141	0.0300	0.0455	0.2302	0.2258	0.0021	100

Table 13 Comparison with other models for the Nasa60 dataset

Model	*MMRE*	*PRED (25)*
Proposed MPSO-CNN	0.0248	100
Dolphin bat algorithm (Fadhil et al., 2020)	0.1457	66.66
Optimal FA (Resmi et al., 2019)	0.0781	85.5
Hybrid model (AIS-GA) (Gharehchopogh et al., 2014)	0.1204	95
Hybrid model (SS-GA) (Maleki et al., 2014a)	0.0756	91.66

7.5 Kemerer dataset

Kemerer dataset contains 15 projects and seven features, where six features are independent: Language, Hardware, Duration, KSLOC, AdjFP, RAWFP, and one dependent feature that will be predicted called Effort. The evaluation results for the Kemerer dataset as follows in Table 14. Table 15 represents the comparison with other models for the Kemerer dataset. The results using an MPSO-CNN have been outperformed the other models.

Table 14 Evaluation results of the Kemerer dataset using MPSO-CNN

Iteration	*MMRE*	*MdMRE*	*MAE*	*RMSE*	*RAE*	*RRSE*	*MSE*	*PRED (25)*
200	0.0287	0.0230	0.0318	0.0425	0.2247	0.1815	0.0018	100
400	0.0231	0.0141	0.0264	0.0387	0.1861	0.1652	0.0015	100
800	0.0212	0.0103	0.0241	0.0375	0.1702	0.1598	0.0014	100

Table 15 Comparison with other models for the Kemerer dataset

Model	*MMRE*	*PRED (25)*
Proposed MPSO-CNN	0.0212	100
ANFIS-SBO (Moosavi and Bardsiri, 2017)	0.268	60
MRLHD (Araújo et al., 2012b)	0.2779	73.33
DEP(MGA) (Araújo et al., 2012a)	0.2578	73.33
DEP-CMPSO (Bilgaiyan et al., 2018)	0.2389	75.00

7.6 Cocomo81 dataset

Cocomo81 dataset includes 81 projects that have 17 features and 63 instances. The evaluation results for Cocomo81 dataset as follows in Table 16. Table 17 represents the comparison with other models for the Cocomo81 dataset, where MMRE only has been mentioned in the last two model. The results using an MPSO-CNN have been outperformed the other models.

Table 16 Evaluation results of the Cocomo81 dataset using MPSO-CNN

Iteration	*MMRE*	*MdMRE*	*MAE*	*RMSE*	*RAE*	*RRSE*	*MSE*	*PRED (25)*
200	0.0429	0.0340	0.0490	0.0791	0.6259	0.4989	0.0063	100
400	0.0371	0.0304	0.0429	0.0730	0.5481	0.4601	0.0053	100
800	0.0353	0.0287	0.0407	0.0656	0.5195	0.4139	0.0043	100

Table 17 Comparison with other models for the Cocomo81 dataset

Model	*MMRE*	*PRED (25)*
Proposed MPSO-CNN	0.0353	100
Optimal FA (Resmi et al., 2019)	0.156	81
FA-FLANN-IFCM (Kaushik et al., 2016b)	0.17	90
FA-RBFN-IFCM (Kaushik et al., 2016b)	0.22	87
WCO + linear regression (Rhmann, 2021)	0.2416	-
WCO + Kernel Regression (Rhmann, 2021)	0.2435	-

7.7 Maxwell dataset

Maxwell dataset contains 62 instances and 27 features. The evaluation results for the Maxwell dataset as follows in Table 18. Table 19 represents the comparison with other models for the Maxwell dataset. The results using an MPSO-CNN have been outperformed the other models.

Table 18 Evaluation results of the Maxwell dataset using MPSO-CNN

Iteration	*MMRE*	*MdMRE*	*MAE*	*RMSE*	*RAE*	*RRSE*	*MSE*	*PRED (25)*
200	0.0406	0.0322	0.0475	0.0691	0.4827	0.4188	0.0048	100
400	0.0405	0.0299	0.0473	0.0685	0.4800	0.4150	0.0047	100
800	0.0410	0.0331	0.0477	0.0655	0.4847	0.3970	0.0043	100

Table 19 Comparison with other models for the Maxwell dataset

Model	*MMRE*	*PRED (25)*
Proposed MPSO-CNN	0.0410	100
FA-FLANN-IFCM (Kaushik et al., 2016b)	0.16	89
FA-RBFN-IFCM (Kaushik et al., 2016b)	0.14	91
IFCM-FLANN (with Chebyshev polynomials) (Kaushik et al., 2016a)	0.28	69
IFCM-FLANN (with Legendre polynomials) (Kaushik et al., 2016a)	0.33	82
IFCM-FLANN (with Power Series polynomials) (Kaushik et al., 2016a)	0.36	68
IFCM-FLANN (with Boubaker polynomials) (Kaushik et al., 2016a)	0.31	80
IFCM-FLANN (with Fibnocci polynomials) (Kaushik et al., 2016a)	0.33	83

7.8 China dataset

China dataset contains 499 instances and 19 features. The evaluation results for the China dataset as follows in Table 20. Table 21 represents the comparison with other models for the China dataset. The results using an MPSO-CNN have been outperformed the other models.

Table 20 Evaluation results of the China dataset using MPSO-CNN

Iteration	*MMRE*	*MdMRE*	*MAE*	*RMSE*	*RAE*	*RRSE*	*MSE*	*PRED (25)*
200	0.0061	0.0026	0.0073	0.0182	0.1084	0.1533	0.0003	100
400	0.0054	0.0024	0.0066	0.0176	0.0968	0.1483	0.0003	100
800	0.0054	0.0023	0.0065	0.0175	0.0956	0.1473	0.0003	100

Table 21 Comparison with other models for the China dataset

Model	*MMRE*	*PRED (25)*
Proposed MPSO-CNN	0.0054	100
FA-FLANN-IFCM (Kaushik et al., 2016b)	0.22	91
FA-RBFN-IFCM (Kaushik et al., 2016b)	0.15	90
IFCM-FLANN (with Chebyshev polynomials) (Kaushik et al., 2016a)	0.32	74
IFCM-FLANN (with Legendre polynomials) (Kaushik et al., 2016a)	0.34	75
IFCM-FLANN (with Power Series polynomials) (Kaushik et al., 2016a)	0.32	72
IFCM-FLANN (with Boubaker polynomials) (Kaushik et al., 2016a)	0.27	72
IFCM-FLANN (with Fibnocci polynomials) (Kaushik et al., 2016a)	0.30	72

The following part represents the comparison between the predicted effort and the actual effort for eight datasets: Figure 5 represents the performance of the Desharnais dataset using PSO-CNN, and Figure 6 represent the performance of the Desharnais dataset using MPSO-CNN. Figure 7 represents the performance of the Albrecht dataset using PSO-CNN, and Figure 8 represents the performance of the Albrecht dataset using MPSO-CNN. Figure 9 represents the performance of the Nasa93 dataset using PSO-CNN and Figure 10 represent the performance of the Nasa93 dataset using MPSO-CNN.

Figure 5 Desharnais dataset using PSO-CNN (see online version for colours)

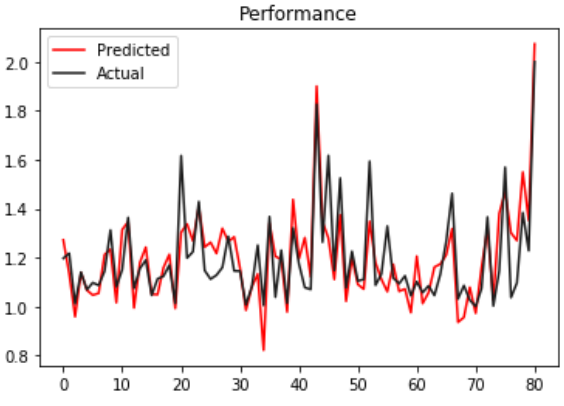


Figure 6 Desharnais dataset using MPSO-CNN (see online version for colours)

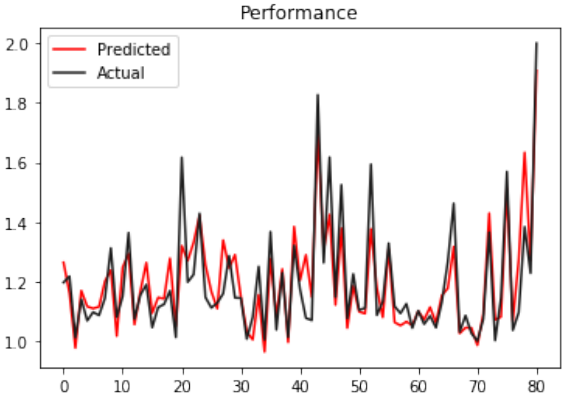


Figure 7 Albrecht dataset using PSO-CNN (see online version for colours)

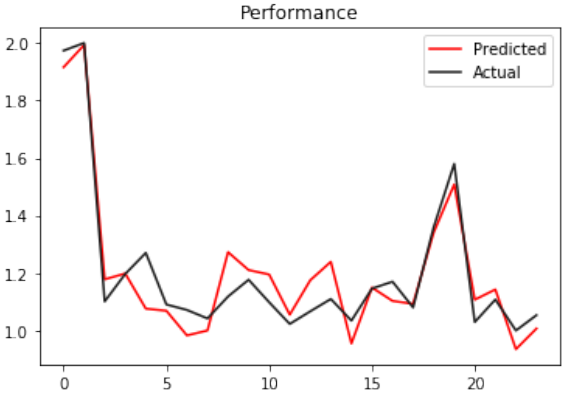


Figure 8 Albrecht dataset using MPSO-CNN (see online version for colours)

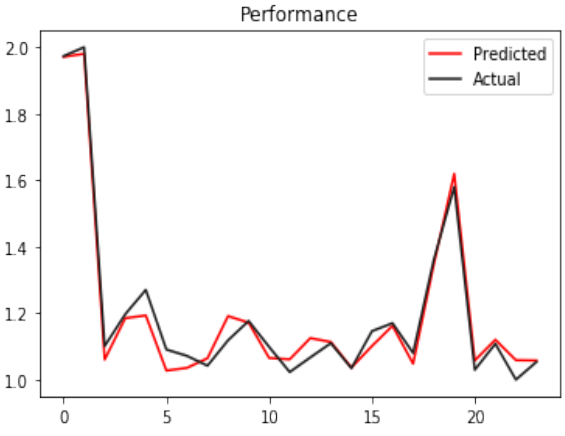


Figure 9 Nasa93 dataset using PSO-CNN (see online version for colours)

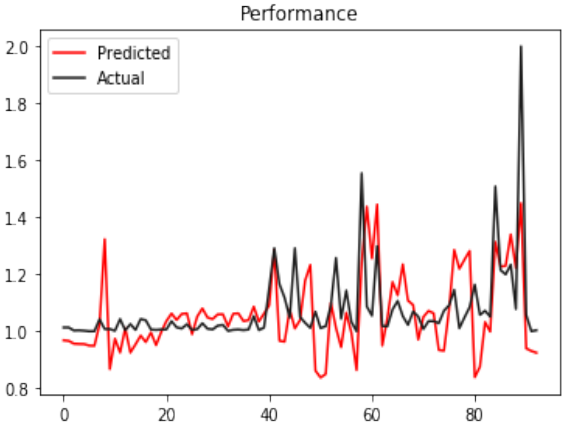


Figure 10 Nasa93 dataset using MPSO-CNN (see online version for colours)

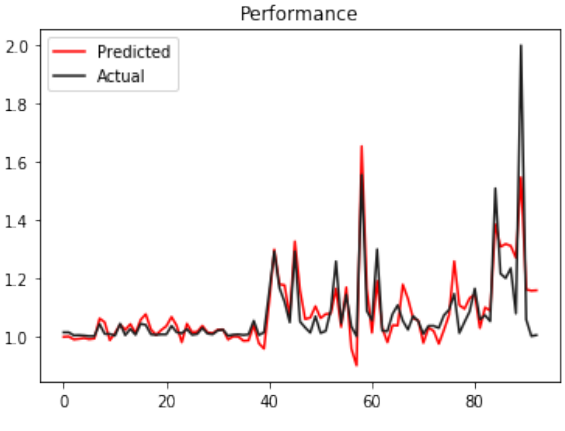


Figure 11 Nasa60 dataset using PSO-CNN (see online version for colours)

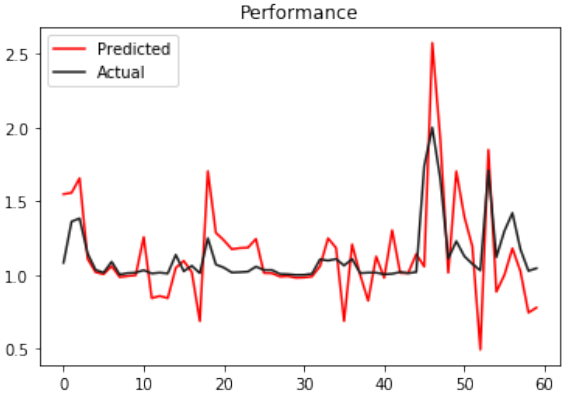


Figure 12 Nasa60 dataset using MPSO-CNN (see online version for colours)

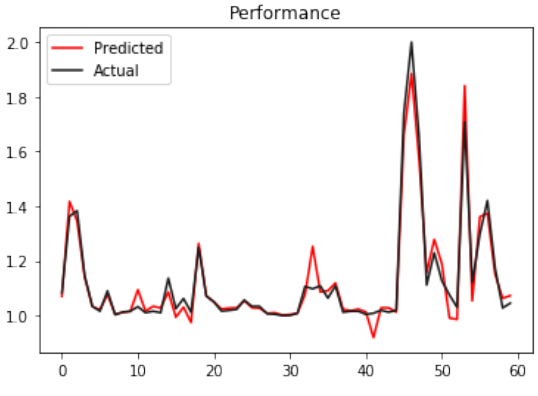


Figure 13 Kemerer dataset using PSO-CNN (see online version for colours)

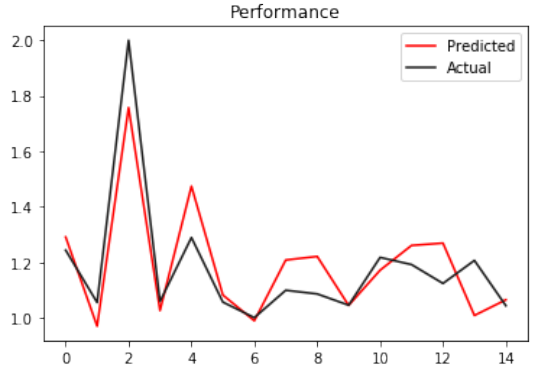


Figure 11 represents the performance of the Nasa60 dataset using PSO-CNN, and Figure 12 represents the performance of the Nasa60 dataset using MPSO-CNN. Figure 13 represents the performance of the Kemerer dataset using PSO-CNN, and Figure 14

represents the performance of the Kemerer dataset using MPSO-CNN. Figure 15 represents the performance of the Cocomo81 dataset using PSO-CNN, and Figure 16 represents the performance of the Cocomo81 dataset using MPSO-CNN.

Figure 14 Kemerer dataset using MPSO-CNN (see online version for colours)

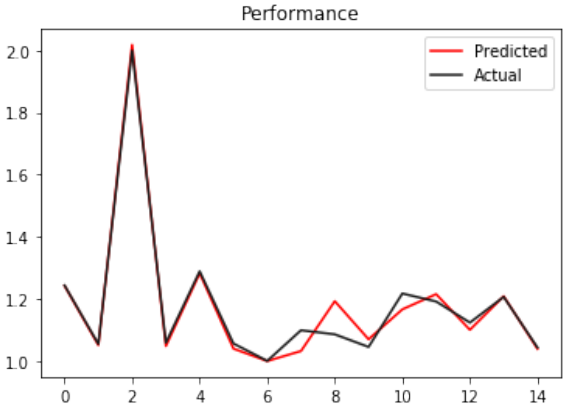


Figure 15 Cocomo81 dataset using PSO-CNN (see online version for colours)

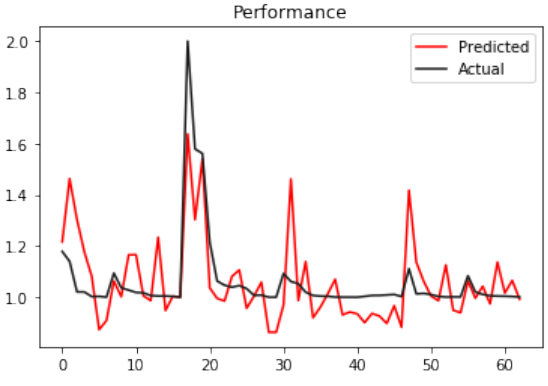


Figure 16 Cocomo81 dataset using MPSO-CNN (see online version for colours)

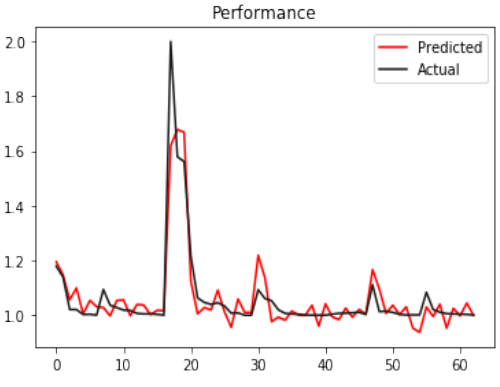


Figure 17 Maxwell dataset using PSO-CNN (see online version for colours)

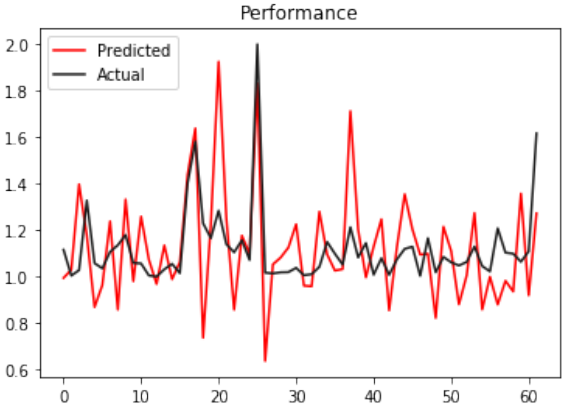


Figure 18 Maxwell dataset using MPSO-CNN (see online version for colours)

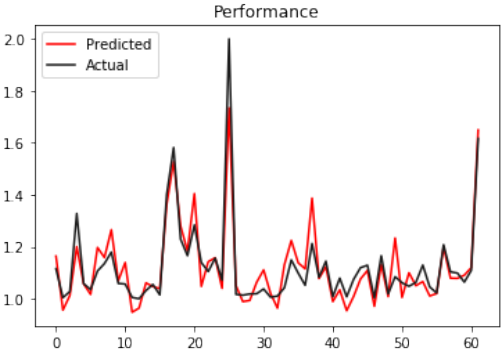


Figure 19 China dataset using PSO-CNN (see online version for colours)

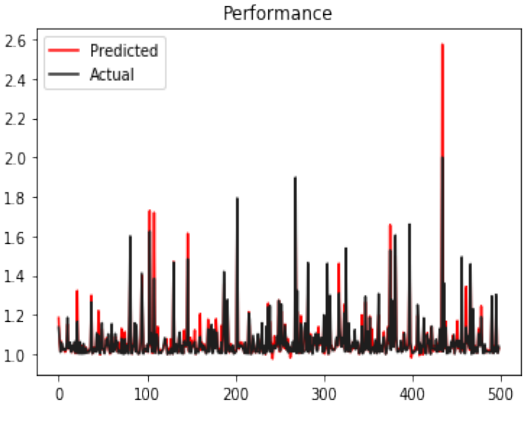


Figure 17 represents the Maxwell dataset's performance using PSO-CNN, and Figure 18 represents the performance of the Maxwell dataset using MPSO-CNN. Figure 19

represents the performance of the China dataset using PSO-CNN, and Figure 20 represents the performance of the China dataset using MPSO-CNN.

Figure 20 China dataset using MPSO-CNN (see online version for colours)

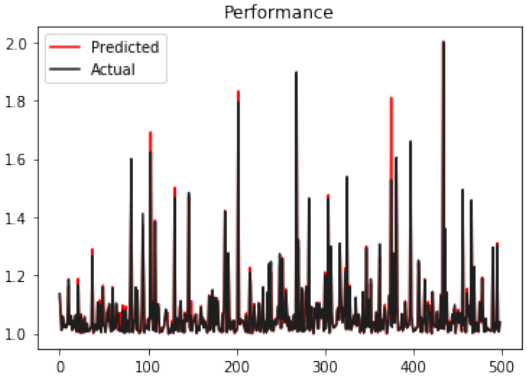


Figure 21 represents the scattered values of MMRE regarding (200, 400, 800) iterations obtained from eight datasets. Figure 22 shows the comparison of MRE values according to eight datasets using the box plots. The China dataset has the lowest median compared with other datasets. Figure 23 shows the convergence process for eight datasets using modified PSO. Three datasets: China, Kemerer, and Albrecht, are converged in a minor number of iterations compared with other datasets.

Figure 21 Scattered values of MMRE obtained from eight datasets (see online version for colours)

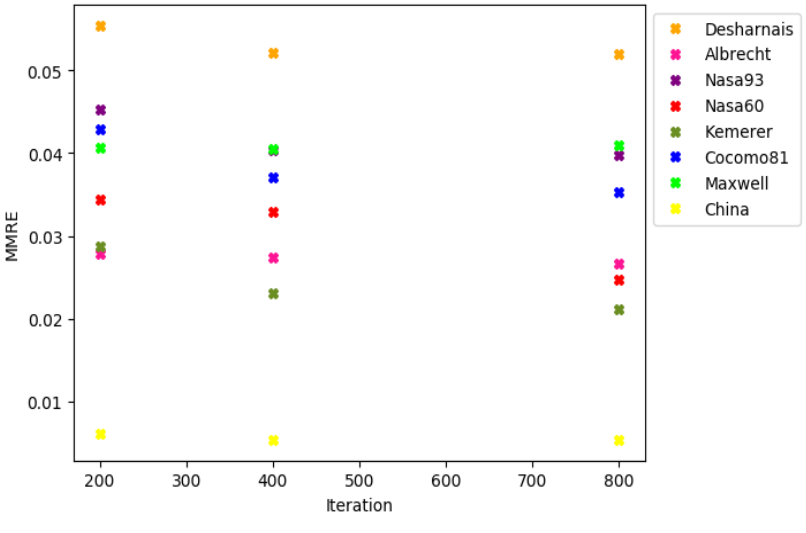


Figure 22 Box plot on MRE results obtained from eight datasets (see online version for colours)

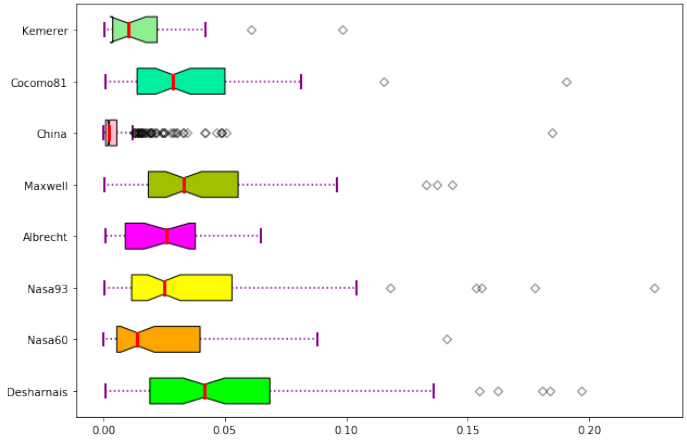
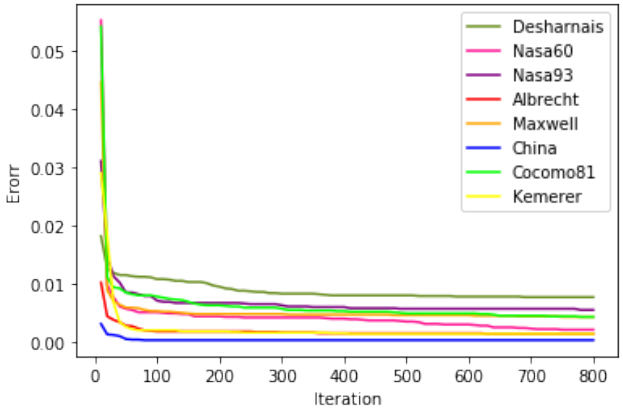


Figure 23 Fitness function (MSE) using MPSO-CNN (see online version for colours)



8 Conclusions

Effort estimation has an important function for project development at early stages. Accurate effort estimation is considered an essential factor for successful companies. Therefore, many types of research have been conducted to improve the effort estimation process, although many techniques have not been used yet. So, in this paper, a cascade forward network has been trained using a modified PSO, which will improve the convergence rate of PSO that is considered the biggest weakness of PSO. Many techniques have been presented to get better convergence of PSO, so here we added a new parameter into the velocity equation of particle that will improve the convergence rate of PSO. We have implemented this methodology on eight datasets, where the normalisation process of the dataset has been done. Also, feature selection has been used to select the best subset of features using Pearson correlation. Several evaluation criteria have been used to evaluate the performance. The convergence rate for three datasets:

China, Kemerer, and Albrecht are converged in a small number of iterations compared with other datasets. In Figure 6, Figure 8, Figure 10, Figure 12, Figure 14, Figure 16, Figure 18, and Figure 20 show a comparison between the predicted effort and the actual effort for eight datasets using modified PSO which has outperformed the standard PSO. The results for all datasets used have outperformed on the previous studies, where the China dataset has the least error of (MMRE =0.0054) compared with other datasets.

9 Future works

We can use other datasets to predict the effort with the data pre-processing required to make the best accuracy as much as possible. Also, we can apply other algorithms such as krill herd (KH) (Moharram and Sundaram, 2022), arithmetic optimisation algorithm (AOA) (Mahajan et al., 2022), and prairie dog optimisation (PDO) (Ezugwu et al., 2022) which will improve the accuracy of effort estimation. Moreover, we can merge more than one optimisation algorithm to get more accurate results.

References

Agrawal, M. and Chari, K. (2007) 'Software effort, quality, and cycle time: a study of CMM level 5 projects', *IEEE Transactions on Software Engineering*, Vol. 33, No. 3, pp.145–156.

Al Asheeri, M.M. and Hammad, M. (2019) 'Machine learning models for software cost estimation', *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, IEEE, September, pp.1–6.

Alsaadi, B. and Saeedi, K. (2022) 'Data-driven effort estimation techniques of agile user stories: a systematic literature review', *Artificial Intelligence Review*, Vol. 55, No. 7, pp.5485–5516.

Andrade, G., Griebler, D., Santos, R., Kessler, C., Ernstsson, A. and Fernandes, L. G. (2022) 'Analyzing programming effort model accuracy of high-level parallel programs for stream processing', *48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, August, pp.229–232.

Angeline, P.J., Saunders, G.M. and Pollack, J.B. (1994) 'An evolutionary algorithm that constructs recurrent neural networks', *IEEE Transactions on Neural Networks*, Vol. 5, No. 1, pp.54–65.

Araújo, R.D.A., Oliveira, A.L., Soares, S. and Meira, S. (2012a) 'An evolutionary morphological approach for software development cost estimation', *Neural Networks*, Vol. 32, pp.285–291.

Araújo, R.D.A., Soares, S. and Oliveira, A.L. (2012b) 'Hybrid morphological methodology for software development cost estimation', *Expert Systems with Applications*, Vol. 39, No. 6, pp.6129–6139.

Attarzadeh, I. and Ow, S.H. (2011) 'Improving estimation accuracy of the COCOMO II using an adaptive fuzzy logic model', *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011*, IEEE, June, pp.2458–2464.

Avlijaš, G. (2020) 'Modern approach to project management, usage, and success rate of agile methodologies: an evidence from Serbia', *Sinteza 2020-International Scientific Conference on Information Technology and Data Related Research*, pp.154–159, Singidunum University.

Azath, H., Mohanapriya, M. and Rajalakshmi, S. (2018) 'Software effort estimation using modified fuzzy C means clustering and hybrid ABC-MCS optimization in neural network', *Journal of Intelligent Systems*, Vol. 29, No. 1, pp.251–263.

Azzeh, M., Cowling, P.I. and Neagu, D. (2010) 'Software stage-effort estimation based on association rule mining and fuzzy set theory', *10th IEEE International Conference on Computer and Information Technology*, IEEE, June, pp.249–256.

Azzeh, M., Nassif, A.B. and Banitaan, S. (2018) 'Comparative analysis of soft computing techniques for predicting software effort based use case points', *IET Software*, Vol. 12, No. 1, pp.19–29.

Badde, D.S., Gupta, A.K. and Patki, V.K. (2013) 'Cascade and feed forward back propagation artificial neural network models for prediction of compressive strength of ready mix concrete', *IOSR Journal of Mechanical and Civil Engineering*, Vol. 3, No. 1, pp.1–6.

Bhavsar, K., Shah, V. and Gopalan, S. (2020) 'Scrumban: an agile integration of scrum and kanban in software engineering', *International Journal of Innovative Technology and Exploring Engineering*, Vol. 9, No. 4, pp.1626–1634.

Bilgaiyan, S., Aditya, K., Mishra, S. and Das, M. (2018) 'A swarm intelligence based chaotic morphological approach for software development cost estimation', *International Journal of Intelligent Systems and Applications*, Vol. 11, No. 9, p.13.

Bisi, M. and Goyal, N.K. (2016) 'Software development efforts prediction using artificial neural network', *IET Software*, Vol. 10, No. 3, pp.63–71.

Boehm, B., Abts, C. and Chulani, S. (2000) 'Software development cost estimation approaches – a survey', *Annals of Software Engineering*, Vol. 10, Nos. 1–4, pp.177–205.

de Barcelos Tronto, I.F., da Silva, J.D.S. and Sant'Anna, N. (2008) 'An investigation of artificial neural networks based prediction systems in software project management', *Journal of Systems and Software*, Vol. 81, No. 3, pp.356–367.

Dejaeger, K., Verbeke, W., Martens, D. and Baesens, B. (2011) 'Data mining techniques for software effort estimation: a comparative study', *IEEE Transactions on Software Engineering*, Vol. 38, No. 2, pp.375–397.

Dhanaseely, A.J., Himavathi, S. and Srinivasan, E. (2012) 'Principal component analysis based cascade neural network for face recognition', *2012 International Conference on Emerging Trends in Science, Engineering and Technology (INCOSET)*, IEEE, December, pp.255–259.

Ezugwu, A.E., Agushaka, J.O., Abualigah, L., Mirjalili, S. and Gandomi, A.H. (2022) 'Prairie dog optimization algorithm', *Neural Computing and Applications*, Vol. 34, No. 22, pp.20017–20065.

Fadhil, A.A., Alsarraj, R.G. and Altaie, A.M. (2020) 'Software cost estimation based on dolphin algorithm', *IEEE Access*, Vol. 8, pp.75279–75287.

Finnie, G.R., Wittig, G.E. and Desharnais, J.M. (1997) 'A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models', *Journal of Systems and Software*, Vol. 39, No. 3, pp.281–289.

Foy, T., Dwyer, R.J., Nafarrete, R., Hammoud, M.S.S. and Rockett, P. (2019) 'Managing job performance, social support and work-life conflict to reduce workplace stress', *International Journal of Productivity and Performance Management*, Vol. 68, No. 6, pp.1018–1041.

Gharehchopogh, F.S., Maleki, I., Ghoyunchizad, N. and Mostafaee, E. (2014) 'A novel hybrid artificial immune system with genetic algorithm for software cost estimation', *Magnt Research Report*, Vol. 2, No. 6, pp.506–517.

Goswami, D.K., Bilgaiyan, S. and Mishra, S. (2021) 'Effort estimation of web-based projects: a systematic review', *International Journal of Productivity and Quality Management*, Vol. 32, No. 2, pp.179–202.

Goyal, S. and Bhatia, P.K. (2019) 'A non-linear technique for effective software effort estimation using multi-layer perceptrons', *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, IEEE, February, pp.1–4.

Goyal, S. and Goyal, G.K. (2011) 'Cascade and feedforward backpropagation artificial neural networks models for prediction of sensory quality of instant coffee flavoured sterilized drink', *Canadian Journal on Artificial Intelligence, Machine Learning and Pattern Recognition*, Vol. 2, No. 6, pp.78–82.

Hacibeyoglu, M. and Ibrahim, M.H. (2018) 'A novel multimean particle swarm optimization algorithm for nonlinear continuous optimization: application to feed-forward neural network training', *Scientific Programming*, Vol. 2018.

Hammad, M. and Alqaddoumi, A. (2018) 'Features-level software effort estimation using machine learning algorithms', *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, IEEE, November, pp.1–3.

Hari, C.H. and Reddy, P.V.G.D. (2011) 'A fine parameter tuning for COCOMO 81 software effort estimation using particle swarm optimization', *Journal of Software Engineering*, Vol. 5, No. 1, pp.38–48.

Hota, H.S., Shukla, R. and Singhai, S. (2015) 'Predicting software development effort using tuned artificial neural network', *Computational Intelligence in Data Mining-Volume 3: Proceedings of the International Conference on CIDM*, 20–21 December 2014, Springer India, pp.195–203.

IR, P.J., Malathy, E.M., Aishwarya, S., Akila, R. and Akshaya, A. (2022) 'A hybrid PSO-ACO algorithm to facilitate software project scheduling', *International Journal of e-Collaboration (IJeC)*, Vol. 18, No. 2, pp.1–12.

Kaur, J., Singh, S., Kahlon, K.S. and Bassi, P. (2010) 'Neural network-a novel technique for software effort estimation', *International Journal of Computer Theory and Engineering*, Vol. 2, No. 1, p.17.

Kaushik, A., Soni, A. K. and Soni, R. (2016a) 'An improved functional link artificial neural networks with intuitionistic fuzzy clustering for software cost estimation', *International Journal of System Assurance Engineering and Management*, Vol. 7, Suppl. 1, pp.50–61.

Kaushik, A., Tayal, D.K. and Yadav, K. (2022) 'The role of neural networks and metaheuristics in agile software development effort estimation', *Research Anthology on Artificial Neural Network Applications*, IGI Global, pp.306–328.

Kaushik, A., Tayal, D.K., Yadav, K. and Kaur, A. (2016b) 'Integrating firefly algorithm in artificial neural network models for accurate software cost predictions', *Journal of Software: Evolution and Process*, Vol. 28, No. 8, pp.665–688.

Kumar, K.V., Ravi, V., Carr, M. and Kiran, N.R. (2008) 'Software development cost estimation using wavelet neural networks', *Journal of Systems and Software*, Vol. 81, No. 11, pp.1853–1867.

Lazzus, J.A., Vega-Jorquera, P., Lopez-Caraballo, C.H., Palma-Chilla, L. and Salfate, I. (2020) 'Parameter estimation of a generalized lotka-volterra system using a modified PSO algorithm', *Applied Soft Computing*, Vol. 96, p.106606.

Lin, J.C. and Tzeng, H.Y. (2010) 'Applying particle swarm optimization to estimate software effort by multiple factors software project clustering', *2010 International Computer Symposium (ICS2010)*, IEEE, December, pp.1039–1044.

Lodhi, V., Chakravarty, D. and Mitra, P. (2018) 'A study of PSO and its variants for fractional abundance estimation in hyperspectral data', *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, IEEE, November, pp.197–201.

Mahajan, S., Abualigah, L., Pandit, A.K. and Altalhi, M. (2022) 'Hybrid Aquila optimizer with arithmetic optimization algorithm for global optimization tasks', *Soft Computing*, Vol. 26, No. 10, pp.4863–4881.

Mahmood, Y., Kama, N., Azmi, A., Khan, A.S. and Ali, M. (2022) 'Software effort estimation accuracy prediction of machine learning techniques: a systematic performance evaluation', *Software: Practice and experience*, Vol. 52, No. 1, pp.39–65.

Maleki, I., Ebrahimi, L. and Gharehchopogh, F.S. (2014b) 'A hybrid approach of firefly and genetic algorithms in software cost estimation', *MAGNT Research Report*, Vol. 2, No. 6, pp.372–388.

Maleki, I.G., Ayat, F.S. and Ebrahimi, L. (2014a) 'A novel hybrid model of scatter search and genetic algorithms for software cost estimation', *MAGNT Research Report*, Vol. 2, No. 6, pp.359–371.

Mellit, A. and Pavan, A.M. (2010) 'A 24-h forecast of solar irradiance using artificial neural network: application for performance prediction of a grid-connected PV plant at Trieste, Italy', *Solar Energy*, Vol. 84, No. 5, pp.807–821.

Minku, L.L. (2019) 'A novel online supervised hyperparameter tuning procedure applied to cross-company software effort estimation', *Empirical Software Engineering*, Vol. 24, No. 5, pp.3153–3204.

Moharram, M.A. and Sundaram, D.M. (2022) 'Spatial-spectral hyperspectral images classification based on Krill Herd band selection and edge-preserving transform domain recursive filter', *Journal of Applied Remote Sensing*, Vol. 16, No. 4, p.044508.

Moosavi, S.H.S. and Bardsiri, V.K. (2017) 'Satin bowerbird optimizer: a new optimization algorithm to optimize ANFIS for software development effort estimation', *Engineering Applications of Artificial Intelligence*, Vol. 60, pp.1–15.

Nagra, A.A., Han, F. and Ling, Q.H. (2019) 'An improved hybrid self-inertia weight adaptive particle swarm optimization algorithm with local search', *Engineering Optimization*, Vol. 51, No. 7, pp.1115–1132.

Nassif, A.B., Azzeh, M., Idri, A. and Abran, A. (2019) 'Software development effort estimation using regression fuzzy models', *Computational Intelligence and Neuroscience*, Vol. 2019, Article ID 8367214, 17pp.

Nassif, A.B., Capretz, L.F. and Ho, D. (2012) 'Software effort estimation in the early stages of the software life cycle using a cascade correlation neural network model', *2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, IEEE, August, pp.589–594.

Okumus, I. and Dinler, A. (2016) 'Current status of wind energy forecasting and a hybrid method for hourly predictions', *Energy Conversion and Management*, Vol. 123, pp.362–371.

Oliveira, A.L., Braga, P.L., Lima, R.M. and Cornélio, M.L. (2010) 'GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation', *Information and Software Technology*, Vol. 52, No. 11, pp.1155–1166.

Qiu, G.Q., Zhao, W.M. and Xiong, G.Y. (2018) 'Estimation of power battery SOC based on PSO-Elman neural network', *2018 Chinese Automation Congress (CAC)*, IEEE, November, pp.91–96.

Rangkuti, F.R.S., Fauzi, M.A., Sari, Y.A. and Sari, E.D.L. (2018) 'Sentiment analysis on movie reviews using ensemble features and pearson correlation based feature selection', *2018 International Conference on Sustainable Information Engineering and Technology (SIET)*, IEEE, November, pp.88–91.

Rao, B.T., Sameet, B., Swathi, G.K., Gupta, K.V., RaviTeja, C. and Sumana, S. (2009) '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.

Resmi, V., Vijayalakshmi, S. and Chandrabose, R.S. (2019) 'An effective software project effort estimation system using optimal firefly algorithm', *Cluster Computing*, Vol. 22, Suppl. 5, pp.11329–11338.

Rhmann, W. (2021) 'An ensemble of hybrid search-based algorithms for software effort prediction', *International Journal of Software Science and Computational Intelligence (IJSSCI)*, Vol. 13, No. 3, pp.28–37.

Rijwani, P. and Jain, S. (2016) 'Enhanced software effort estimation using multi layered feed forward artificial neural network technique', *Procedia Computer Science*, Vol. 89, pp.307–312.

Saffar, N. and Obeidat, A. (2020) 'The effect of total quality management practices on employee performance: the moderating role of knowledge sharing', *Management Science Letters*, Vol. 10, No. 1, pp.77–90.

Satapathy, S.M., Acharya, B.P. and Rath, S.K. (2016) 'Early stage software effort estimation using random forest technique based on use case points', *IET Software*, Vol. 10, No. 1, pp.10–17.

Shah, M.A., Jawawi, D.N.A., Isa, M.A., Younas, M., Abdelmaboud, A. and Sholichin, F. (2020) 'Ensembling artificial bee colony with analogy-based estimation to improve software development effort prediction', *IEEE Access*, Vol. 8, pp.58402–58415.

Sheta, A.F., Ayesh, A. and Rine, D. (2010) 'Evaluating software cost estimation models using particle swarm optimisation and fuzzy logic for NASA projects: a comparative study', *International Journal of Bio-Inspired Computation*, Vol. 2, No. 6, pp.365–373.

Shukla, S. and Kumar, S. (2019) 'Applicability of neural network based models for software effort estimation', *2019 IEEE World Congress on Services (SERVICES)*, July, Vol. 2642, pp.339–342, IEEE.

Shukla, S., Kumar, S. and Bal, P.R. (2019) 'Analyzing effect of ensemble models on multi-layer perceptron network for software effort estimation', *2019 IEEE World Congress on Services (SERVICES)*, July, Vol. 2642, pp.386–387, IEEE.

Singh, D. and Singh, B. (2020) 'Investigating the impact of data normalization on classification performance', *Applied Soft Computing*, Vol. 97, Part B, p.105524.

Suri, P.K. and Ranjan, P. (2012) 'Comparative analysis of software effort estimation techniques', *International Journal of Computer Applications*, Vol. 48, No. 21, pp.12–19.

Tariq, S., Usman, M. and Fong, A.C. (2020) 'Selecting best predictors from large software repositories for highly accurate software effort estimation', *Journal of Software: Evolution and Process*, Vol. 32, No. 10, p.e2271.

Thota, M.K., Shajin, F.H. and Rajesh, P. (2020) 'Survey on software defect prediction techniques', *International Journal of Applied Science and Engineering*, Vol. 17, No. 4, pp.331–344.

Warsito, B., Santoso, R. and Yasin, H. (2018) 'Cascade forward neural network for time series prediction', *Journal of Physics: Conference Series*, May, Vol. 1025, No. 1, p.012097, IOP Publishing.

Wu, D. and Wang, G. G. (2022) 'Employing reinforcement learning to enhance particle swarm optimization methods', *Engineering Optimization*, Vol. 54, No. 2, pp.329–348.

Yadav, C.S. and Singh, R. (2014) 'Prediction model for object oriented software development effort estimation using one hidden layer feed forward neural network with genetic algorithm', *Advances in Software Engineering*, Vol. 2014, p.2.

Yadav, N.K. (2019) 'Rescheduling-based congestion management scheme using particle swarm optimization with distributed acceleration constants', *Soft Computing*, Vol. 23, No. 3, pp.847–857.