Journal of Engg. Research Vol. 8 No. (4) December 2020 pp. 99-116 https://doi.org/10.36909/jer.v8i4.7881

Hybrid Optimization Driven RideNN for Software Reusability Estimation

Ramu Vankudoth* and Dr. P. Shireesha**

**Research Scholar (Ph.D.), Department of Computer Science, Kakatiya University, Vidyaranyapuri, Warangal, Telangana 506009*
***Assistant Professor, Department of Computer Science and Engineering, Kakatiya Institute of Technology and Science, Bheemaram, Hanamkonda, Telangana 506015*
**Corresponding Author: ramuvankudoth86@gmail.com*

Submitted: 21/01/2019
Revised: 25/09/2019
Accepted: 01/10/2019

ABSTRACT

Measuring the software reusability has become a prime concern in maintaining the quality of the software. Several techniques use software related metrics and measure the reusability factor of the software, but still face a lot of challenges. This work develops the software reusability estimation model for efficiently measuring the quality of the software components over time. Here, the Rider based Neural Network has been used along with the hybrid optimization algorithm for defining the reusability factor. Initially, nine software related metrics are extracted from the software. Then, a holoentropy based log function identifies the Measuring the software reusability has become a prime concern in maintaining the quality of the software. Several techniques use software related metrics and measure the reusability factor of the software, but still face a lot of challenges. This work develops the software reusability estimation model for efficiently measuring the quality of the software components over time. Here, the Rider based Neural Network has been used along with the hybrid optimization algorithm for defining the reusability factor. Initially, nine software related metrics are extracted from the software. Then, a holoentropy based log function identifies the normalized metric function and provides it to the proposed Cat Swarm Rider Optimization based Neural Network (C-RideNN) algorithm for the software reusability estimation. The proposed C-RideNN algorithm uses the existing Cat Swarm Optimization (CSO) along with the Rider Neural Network (RideNN) for the training purpose. Experimentation results of the proposed C-RideNN are evaluated based on metrics, such as Magnitude of Absolute Error (MAE), Mean Magnitude of the Relative Error (MMRE), and Standard Error of the Mean (SEM). The simulation results reveal that the proposed C-RideNN algorithm has improved performance with 0.0570 as MAE, 0.0145 as MMRE, and 0.6133 as SEM.

Keywords: Software reliability; Software reusability estimation; Log function; RideNN; Cat Swarm Optimization.

INTRODUCTION

Software reliability has considered as the prime concern in software development as most of the companies try to build high quality software to meet up with the customer demands. Software developed for the particular task needs to fulfil the demand of the user and should not fail in important areas. Building high reliable software for meeting the customer needs has risen as the prime demand in the recent years. Building reliable software faces a lot of difficulties as the software reliability cannot be measured in the initial stages. In the software development cycle, the reliability of the software can be estimated in the later stages, and hence,there is a lot of chance for the error detection. The software reliability depends on the defect density, and further, it is necessary to maintain the defect density as low as possible to improve the quality of the software. For reducing the defect density in the software, the residual effects in the software need to be controlled (Rai, *et al.,* 2017). In the recent years, the software industry has tried to build the reusable components for the software construction.

For estimating the software reusability, the software needs to be downloaded from the website, which can be time consuming (Jalender, *et al.,* 2010). For easier download, drag and drop features are available in the latest applications. Drag and drop feature allows to transfer the software from one website to another. During the file transfer from one website to another, the components of the software are dropped to the library (Jalender, *et al.,* 2014).The software attributes help in mapping the module to the numerical value and thus, helps in software measure (Fenton & Kaposi, 1987). The software measure quantifies the quality of the software by defining its characteristics and the nature of the complexity(Pizzi, 2006).

The software contains components, necessarily a program, and a section of program, classes, modules or tests cases. The software component signifies an independent substitutable segment in the software and helps in executing the function (Caldiera&Basili, 2011). The software components are generated independently as the software package and used for supporting the functional unit. Further, it serves as an interface between other components and constitutes a big software package. While building the software, the processes are placed as the distinct class for understanding the relation of the functions and the data. Software development cycle requires more computation time and human resources. For reducing the computation time for the software development, researchers opt for reusing the software components. This technique is referred to as software reusability. Software reusability allows the user to create a cost effective, reliable and error-free software design. Software reusability effectively utilizes the modules and the components(Sommerville, 2011). It improves the software feasibility and the expected use proneness for the components in the software. Few of the advantages of the software reusability are reduced cost for software development, low time consumption in design, high reliability,quality of service, market acquisition, and low maintenance (Padhy, *et al.,* 2017a).

Developing the software with the high reusability factor can be considered as one of the design constraints. The software with the high reusability has high quality, and thus, component reusability is considered as the quality attributes in the development cycle(Singh &Tomar, 2017). Optimization algorithms (Yadav, P. 2014, Polepally, V.k and Chatrapati, K.S. 2017, Thomas, R and Rangachar, M.J.S. 2018) have been utilized in software reusability estimation. The software reuse approach saves 20percentage of the development costs, and thus, saves the design time. For building the reusable components, most researchers have used the Component based software engineering. This technique identifies and develops the reusable components for the various systems. The reusability(Goel& Bhatia, 2013)estimation for the software depends on the reusability of the individual components (Singh, *et al.,* 2014). Component-Based Software Development (CBSD) is a mechanism for reusing the existing software components for speeding the software development. Fuzzy logic offers significant advantages over the other approaches due to its ability to logically represent qualitative aspects of inspection data and apply flexible inference rules. The rule-based fuzzy model with more optimised weight values and updated fuzzy rulesis capable of measuring the reusability of a software component and further helps in the measurement of quality of Computer Based Software (CBS)(Singh &Tomar, 2017).A metric-based reusability score(Diamantopoulos, *et al.,* 2016) calculates the degreeof reusability of software components based on the valuesof eight static analysis metrics. The assessment depends onwhether the values of the metrics exceed certain thresholds,as defined in current literature. When several thresholds areexceeded, the returned reusability score is lower (Papamichail, *et al.,* 2018). In (Papamichail et al., 2018), a reusability score has been formulated by employing information from GitHub stars and forks, which denote the extent to which software components are adopted by developers. Here, various machine learning algorithms have been evaluated to construct models for reusability estimation at both class and package levels. A reusability index (REI) (Ampatzoglou, *et al*., 2018) is developed as a synthesis of various software metrics, which cover a number of related reusability aspects. An REI presents the highest predictive and discriminative power, and it is the most consistent in ranking reusable assets, and the most strongly correlated to their levels of reuse.

The main aim of this paper is to develop a technique for software reusability prediction model in order to maintain the optimal reusability of the software components with no aging probability and fault proneness. The criteria, such as complexity, cohesion, and coupling are considered for reusability with a total of nine metrics in this paper.The metrics considered are, Weighted Methods per Class (WMC), Coupling between Object (CBO), Number of Children (NOC),Comments Lines of Code (CLOC) (Kaur, *et al.,* 2015), Message Passing Coupling (MPC),Number of Statements (STAT), Depth of Inheritance Tree (DIT),Response for a Class (RFC), and Lack of Cohesion in Methods (LCOM). The normalized metric values are obtained and then, the Log function is applied over the data as the pre-processing step. The features are selected with the use of the Holo-entropy and then, the features are used to estimate the software to find the reusability factor. The estimation is performed with the proposed Cat Swarm Rider Optimization based Neural Network (C-RideNN) algorithm, which is obtained with the integration of the Cat Swarm optimization (CSO) algorithm (Chu, *et al.,*2006) and the Rider Optimization algorithm (ROA) based Neural Network (Binu&Kariyappa, 2018).

The major contribution of this work is the development of the C-RideNN algorithm for computing the software reliability measure. The proposed C-RideNN algorithm is formulated by integrating the CSO with the ROA algorithm.

The paper organization is done as follows: Section 1 introduces the need for the software reusability and the ways of computing it. Section 2 reviews some literary works related to the software reusability estimation. Section 3 presents the proposed C-RideNN algorithm and the various software related metrics extracted from the software for the software reusability estimation. Section 4 depicts the simulation results of the same and the conclusion is given in section 5.

MOTIVATION

Literature Survey

Some works related to the software reusability estimation are discussed in detail below:

Neelamdhab Padhy *et al.* (2017a) presented the aging resilient software reusability prediction model. The model was used for the determining the earlier aging-resilient. Here, the reusability assessment is compactable for only two softwares and hence, cannot be generalized for major system. NeelamdhabPadhy*et al.* (2017b)presented the evolutionary based model and the machine learning scheme for the reusability estimation. The scheme was specifically designed for the regression tests for the software reusability estimation. Machine learning based scheme helps in software component reusability, reliability, survivability, aging prediction, and stability, and for software excellence assurance purposes. The model cannot be adopted for the cloud computing related software. Jens Otto *et al.* (2018) developed the parameter estimation approachfor optimizing the parameters related to the software reusability. The optimization scheme configures the Cyber-Physical Production Systems by selecting the optimal parametersand improves the discrete manufacturing. The combinatorial optimization does not help the black box parameter estimation. MichailPapamichail*et al.* (2018) proposed the software reusability estimation approach based on the rationale model. The scheme helped to find the software reusability even before integrating the component code to the source code of the software. The model was effective for defining the reusability of the components with the class and package level. In the repository level, the scheme was not adequate enough for the reusability estimation.

Aditya Pratap Singh and Pradeep Tomar(2017)presented the rule-based fuzzy inference system for defining the software reusability. The model used many metrics for defining the reusability and the metrics define the quality and reusability measure of the software components. The defined metrics have different impact on the components and thus, cannot be directly quantified as the crisp value. R. Selvarani, and P. Mangayarkarasi(2017) had developed thereusability estimation technique based on the optimization algorithm. Using the optimization algorithm for defining the software reusability has proven to have improved alignment among the predicted and expected outcome. The scheme has reduced results while the real-world constraints were introduced. Jatain Aman, and Sangeet Srivastava

(2017) developed the re-engineering and reverse engineering techniques for estimating the reusability of the software. The model had the three-stage framework along with the automated reverse engineering tool (ODRET) for the estimation. The model allowed estimating the reusability at component level, integration level and system level. The scheme lacked proper expertise, and adaptability.

Challenges

Some of the challenges faced during the software reusability estimation are presented as follows:

- Fuzzy Logic (FL) for the reusability estimation has been developed in the recent years as interpreting the fuzzy inference rules to the linguistic knowledge is an easy task. However, while using the complex multivariable systems,for the reusability estimation, adopting FL may fail in certain cases (Ahmed & Al-Jamimi, 2013).
- SVM classifier was used for modelling various Object Oriented (OO) related metrics, and thus, improved the planning and the testing phases. The SVM related testing helped in identifying the fault-prone parts in the software code. But, the SVM technique requires further investigation for the acceptability in real world scenario (Ahmed & Al-Jamimi, 2013).
- The efficiency of the machine learning models relay on the learning rate (probably NN), but less efforts have been made to use the evolutionary algorithm,like Genetic Algorithms (GA), for optimizing the learning rate. Also, the research was not focused towards the reusability estimation (Padhy, *et al.,* 2017a).
- While developing the software design with the object-oriented metrics, static coupling measures are required, but while explaining the change effort for the metrics, the coupling measures may be inadequate (Arisholm, *et al.,* 2004).
- Some works consider both Chidamber and Kemerer's Metrics Suite (CK) and Metrics for Object Oriented Design (MOOD) metrics for the reusability estimation. But both the metrics have the common disadvantage and they are less effective in large scale system (Chong & Lee, 2015).

PROPOSED C-RIDENN BASED SOFTWARE REUSABILITY ESTIMATION

This section presents the brief description of the proposed system with the C-RideNN algorithm for the software reusability factor estimation. Figure 1 presents the architecture of the proposed C-RideNNbased software reusability estimation. Software reusability estimation through the NN based optimization has proven to yield improved results, but some existing models have failed to consider various software related metrics for the estimation. Here, a total of nine software related metrics are extracted from the software, each representing cohesion, complexity and coupling related functions. After extracting the suitable features, the information is pre-processed by applying the log function. Most of the features extracted from the software may have the same information, and hence, contributes less to the classifier training. Hence, this work adopts the holoentropy based function for selecting the suitable features for the estimation. The software reusability estimation is carried out by the proposed C-RideNN algorithm. The proposed C-RideNN algorithm estimates the reusability factor for the software.

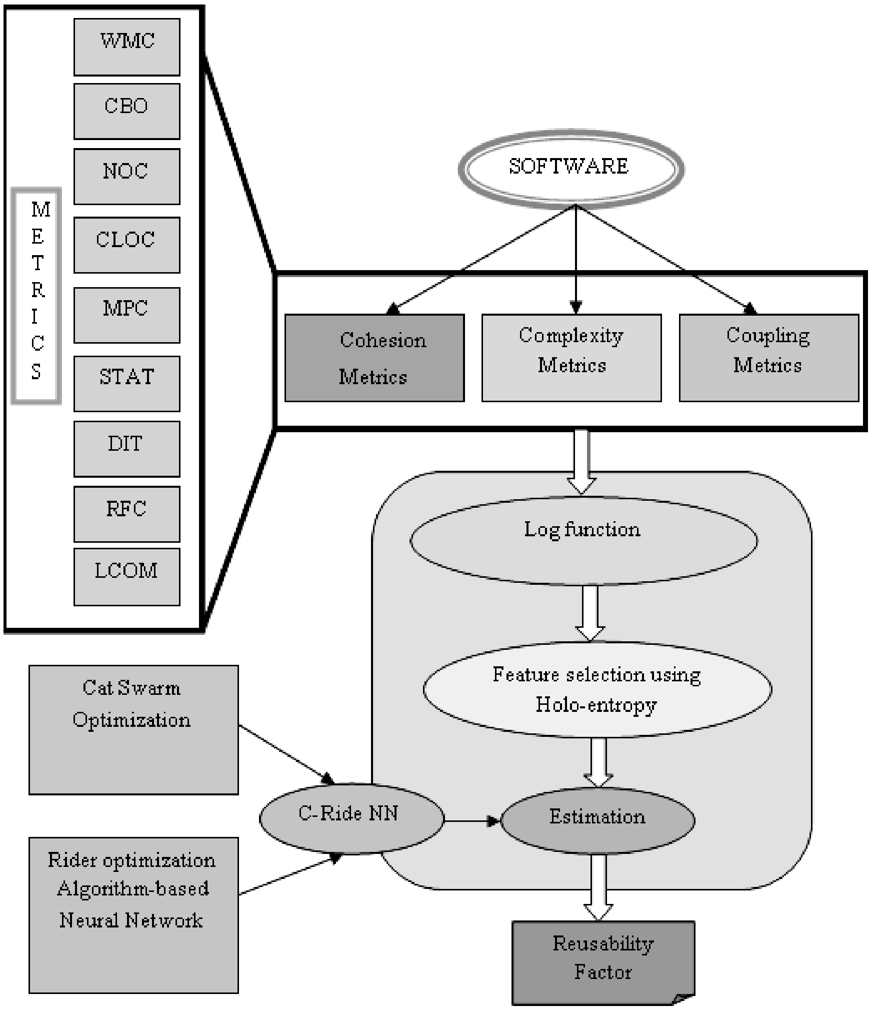


Figure 1. Bock diagram of the reusability factor estimation system with the proposed C-RideNN algorithm

Feature extraction from the software

The initial step in the reusability estimation process is to extract the suitable features from the software, and in this work, a total of 9 features are extracted. The metrics considered are, WMC, CBO, NOC, CLOC (Kaur, *et al.,* 2015), MPC, STAT, DIT, RFC, and LCOM.

The metrics extracted from the software are defined as follows:

*WMC:*The WMC metric refers to various cumulative addition processes carried out in the software project, and further extracts the complexities of all the class functions.

*DIT:*The software has many root functions and the maximum length of the root node to access the tree is signified by the DIT metric.

NOC: The adjacent subclasses in the class function are adjacent to the class, and also, these class functions contribute to the class hierarchy. The NOC metric takes note of the adjacent subclasses in the software.

*CBO:*The classes are connected to the RFC and the quality of the connection is specified by the CBO metric. The connection established among the class and the RFC signify the message response provided by the objects in the class.

*LCOM:*The methods/functions in the class may have divergence information, and the dissimilarity among the methods occurred due to the instanced variables in the class is defined through LCOM.

CLOC: The CLOC metric defines the total number of line codes in each class, and further, signifies the presence of comments in the line.

*MPC:*It defines the total number of functions in the class.

STAT: As the name signifies, the STAT metric reports the total number of statements in the class. But, the STAT metric does not counts the statements in the inner classes and interfaces.

RFC: The RFC metric counts the total number of methods to be executed while the message is passed from the object.

Construction of the feature database

As a total of nine software related features are extracted from each software, the features are collected in together to form the feature database D. Let the number of software considered for the reusability estimation be S. Hence, the feature database has the size of $S \times 9$.

Pre-Processing

After estimating the metrics from the software, the feature database is subjected for the pre-processing. The pre-processing is done for the data normalization and it is carried out by applying the log function over the feature database D. Applying the log function over the data files removes the redundant feature information and simplifies the feature selection process. Most of the features extracted from the software may not contribute to the software reusability factor estimation and hence refined by applying the log function. The pre-processed database is expressed as,

$$P = \log(D) \tag{1}$$

where, D represents the feature database with the software metrics of size $S \times 9$. Applying the log function to the software metrics help in easy data transfer.

Feature selection based on Holoentropy

The feature database has lot of features and training each feature with the network requires more time. Also, each and every metric does not contribute in the estimation of the software reusability factor. For avoiding this issue, only the significant features are selected for the training based on the holoentropy factor (Mane & Jadhav, 2017). The holoentropy feature includes the additional weight parameters along with the entropy value so as to refine the selection process. Here, the holoentropy value for the software feature is calculated based on the following formula,

$$H(d_j) = Z.E(d_j) \tag{2}$$

where Z refers to the holoentropy weight parameter and $E(d_j)$ indicates the entropy value of the j^th^ feature in the database. The expression for the weight and the entropy is expressed by the following equations,

$$Z = 2\left(1 - \frac{1}{1 + \exp\left[E(d_j)\right]}\right) \tag{3}$$

$$E(d_j) = -\sum_{j=1}^{a(d_j)} p_j \log p_j \tag{4}$$

where p_j refers the probability of the occurrence of the jth feature in the database and $a(d_j)$ indicates the number of unique features in the database. The feature selection is done by the following selection criteria, while the features having minimal holoentropy value are selected and the other features are neglected. The selection criteria for the feature selection arebased on the condition $d_j < H$, i.e., the features having the holoentropy value less than H are selected as the suitable features for the classification and the other features are neglected. The finally selected features are stored in the feature database D_s. Also, D_s refer to the selected metrics for the software reusability estimation. The size of the database D_s is always less than D. The selected feature database has in total of K features and hence, the size is expressed as $S \times K$.

Reusability estimation using the Proposed C-RideNN

After selecting the suitable K features, the features are given to the proposed C-RideNN classifier for the reusability estimation. Here, the C-RideNN algorithm is developed for estimating the reusability of the software. The proposed C-RideNN algorithm is developed using the CSO (Chu, *et al.,* 2006) and the existing ROA.

Architecture of the proposed C-RideNN algorithm

The architecture of the proposed C-RideNN algorithm is presented here, along with the expressions for the various layers used, for the software reusability computation. The diagrammatic representation of the proposed C-RideNN classifier is presented in figure 2.

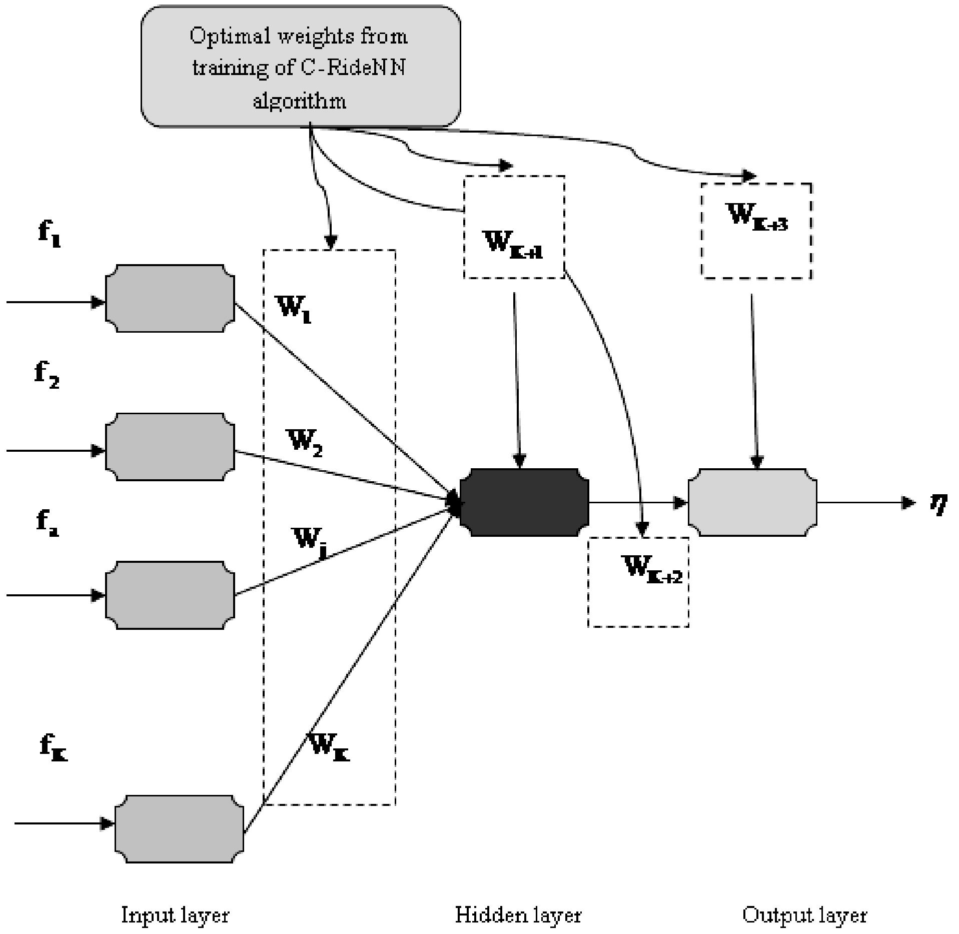


Figure 2. Architecture of the proposed C-RideNN algorithm.

As given in the above figure, the proposed C-RideNN algorithm has three layers, i.e. input, hidden and output layers, with each layer have weights and bias for output computation. The neurons in each layer has the weights and bias. This work selects the optimal values for the C-RideNN algorithm through the optimization procedure. The selected K features in the previous section are fed as the input to the input layer, and it is expressed as,

$$f = \{f_1, f_2, \ldots, f_a \ldots f_K\} \tag{5}$$

where f_a refers to the ath feature fed to the input layer of the C-RideNN classifier. As there are K number of selected features, the total neurons considered for the input layer is K. After providing the inputs to the input layer, it is necessary to define the weights and biases for the C-RideNN. The input and the hidden layers of the C-RideNN classifier are interconnected through the weights and are given as,

$$W = \{W_1, W_2, \ldots, W_a \ldots W_K\} \tag{6}$$

Where W_a refers to the ath weight. Now, based on the weights and the biases in the input, and hidden layer, the input is processed and the final output of the proposed C-RideNN classifier is represented as follows,

$$\eta = W_{K+2} * \left[\log\mathrm{sig}\left(\sum_{a=1}^{K} f_a * W_a + W_{K+1} \right) \right] + W_{K+3} \tag{7}$$

where W_{K+1}, and W_{K+3} refer to the biases in the hidden and the output layer, and W_{K+2} refers to the weight amidst the hidden and output neurons. For the output computation, the weights $\{W_1, W_2, \ldots, W_a \ldots W_K\}$, W_{K+2} and the bias W_{K+1}, and W_{K+3} need to be optimally selected and it is done in the training phase.

A sigmoid function is a mathematical function having a characteristic «S»-shaped curve or sigmoid curve. It refers to the special case of the logistic function and defined by the formula

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \tag{8}$$

Training of proposed C-RideNN algorithm

Here, the training procedure of the proposed C-RideNN algorithm is briefly explained. Selecting the optimal values for the weights and biases improves the accuracy of the software reusability estimation. The C-RideNN algorithm integrates ROA and CSO, for finding the position update. The steps involved in the proposed C-RideNN algorithm are explained in brief as follows:

Initialization: The position of the rider is randomly fixed and the solution space has K + 3 elements. Consider there are N number of riders and thus, it is necessary to find $N * P$ elements as the optimal values, which equals K + 3 positions.

Fitness: Here, the success rate is referred as the fitness for the C-RideNN algorithm. For the each rider in the solution space, error value is computed and the solution having the minimal error is considered as the fittest solution, and the expression for the error based fitness function is expressed as follows:

$$\mathrm{Fitness} = \frac{1}{\mathrm{MSE}} \tag{9}$$

where MSE refers to the mean square error and it is expressed as the following equation,

$$\mathrm{MSE} = |\eta - \eta_{\mathrm{approx}} \tag{10}$$

where η and η_{approx} refer to the reusability value, and its approximate value, respectively

*Position update:*The proposed C-RideNN algorithm has four components, such as bypassrider, follower, overtaker, and attacker. For finding the actual positions of the rider, the positions of the bypassrider, follower, overtaker and attacker are updated. In this work, the bypass rider update of ROA is modified using the CSO for improving the selection of the weights more optimally. Here, the update process is modified using the CSO such that the weights are most suitable for the output estimation. The CSO algorithm depends on the behavior of the cats, and most probably two general characteristics, such as seeking and tracking are considered for the update. For updating the bypass rider of RideNN, the CSO update for the tracing mode is considered. The general expression of the RideNN bypass rider (Binu & Kariyappa, 2018) is expressed as,

$$G^{B}_{t+1}(p,q)=\alpha[G_t(\chi,q)*\omega(q)+G_t(\delta,q)*(1-\omega(q))] \quad (11)$$

where α specifies the random number in range of [0,1]. The above expression signifies the relation of the solution update provided by the bypass rider of the RideNN algorithm. Here, in equation (9),we consider the functions $\chi,\delta=p$, and thus, the above equation gets modified as,

$$G_{t+1}(p,q)=\alpha[G_t(p,q)*\omega(q)+G_t(p,q)*(1-\omega(q))] \quad (12)$$

Rearranging the above equation,

$$G_{t+1}(p,q)=\alpha G_t(p,q)[\omega(q)+(1-\omega(q))] \quad (13)$$

For modifying the solution update of the bypass rider of RideNN, this work considers the solution update of the CSO in the tracing mode. In the tracing mode of the CSO, the solution gets altered based on the velocity and it is given as,

$$G_{t+1}(p,q)=G_t(p,q)+V_{t+1}(p,q) \quad (14)$$

where $V_{tl+1}(p,q)$ refers to the velocity during the tracing mode of the CSO (Chu, *et al.,* 2006). The velocity update of the CSO depends on various factors, and thus, the equation alters as,

$$G_{t+1}(p,q)=G_t(p,q)+V_t(p,q)+u_1\times v_1\times[G_{best,q}-G_t(p,q)] \quad (15)$$

Now, modifying the above equation,

$$G_{tl+1}(p,q)=G_{tl}(p,q)[1-u_1v_1]+V_{tl}(p,q)+u_1\times v_1\times G_{best,q} \quad (16)$$

$$G_t(p,q)=\frac{1}{[1-u_1v_1]}[G_{t+1}(p,q)-V_t(p,q)-u_1\times v_1\times G_{best,q}] \quad (17)$$

Now, substitute the above equation to the solution update provided by the RideNN i.e. substitute equation (17) in equation (13), and the modified equation is expressed as,

$$G_{t+1}(p,q)=\alpha\frac{1}{[1-u_1v_1]}[G_{t+1}(p,q)-V_t(p,q)-u_1\times v_1\times G_{best,q}][\omega(q)+(1-\omega(q))] \quad (18)$$

Rearranging the above equation,

$$G_{t+1}(p,q)=\frac{[1-u_1v_1]}{[1-u_1v_1-\alpha[\omega(q)+(1-\omega(q))]]}\left\{\frac{\alpha}{[1-u_1v_1]}[V_t(p,q)+u_1\times v_1\times G_{best,q}][\omega(q)+(1-\omega(q))]\right\} \quad (19)$$

where u_l and v_l refer to the constant for the optimization. The follower helps in reaching the target position quickly. Next, the position of the follower is updated as follows,

$$G^{F}_{t+1}(p,r) = G_t(m,r) + \left[\cos\left(A^{t}_{p,r}\right) * G_t(m,r) * h^{t}_{p}\right] \tag{20}$$

where h^{t}_{p} refers to the distance travelled by the pth rider, and $A^{t}_{p,r}$ refers to the steering angle. The overtaker position is updated based on the following equation,

$$G^{O}_{t+1}(p,r) = G_t(p,v)r + \left[G_t(m,r) * Z^{t}_{p}\right] \tag{21}$$

where Z^{t}_{p} indicates the direction indicator for the pth rider. Finally, the position of the attacker is updated as,

$$G^{A}_{t+1}(p,q) = G_t(m,q) + \left[\cos\left(A^{t}_{p,q}\right) * G_t(m,q)\right] + h^{t}_{p} \tag{22}$$

Finding the rider with the maximum success rate:

After the position update, the rider having the best success rate is calculated and the winner is declared as the best solution.

Termination: The iteration continues until the maximum iteration, and the final position of the rider refers to the K+3 optimal weights for the classifier.

Testing of proposed C-RideNN algorithm

After finding the optimal weights from the training phase, the output of the C-RideNN classifier is determined. For any test input, the proposed C-RideNN algorithm identifies the software reusability factor η through the optimal weights.

RESULTS AND DISCUSSION

The simulation results achieved by the proposed C-RideNN algorithm for the software reusability estimation are presented in this section. The experimentation of the proposed scheme is done by considering the web related software from the standard database.

Experimental Setup

The proposed software reusability estimation factor with the C-RideNN algorithm is implemented in the JAVA tool. The PC used for the experimentation has the following specific configurations: Windows 10 OS, 4 GB RAM, and Intel I3 processor.

Database description

The software required for the reusability score estimation is considered from the Github database (Github database, https://github.com/ accessed on August 2018). The Github contains large range of open source softwares andit is considered as the open platform for downloading the applications and the software related services. It has in total of more than millions of software related services, and for this work a total of 100 services are considered.

Performance metrics

The evaluation of the reusability factor identified by the proposed C-RideNN algorithm is evaluated by considering three metrics, such as Magnitude of Absolute Error (MAE), Mean Magnitude of the Relative Error (MMRE), and

Standard Error of the Mean (SEM). The evaluation metrics used in this work are described in detail as follows,

*MAE:*MAE can be calculated as the magnitude of difference among the absolute and the approximate value.

*MMRE:*MMRE projects the ratio of the difference in ground value and the actual value to the score value estimated by the proposed scheme.

*SEM:*SEM signifies the ratio of the standard deviation of the population to the total number of observations.

Comparative techniques

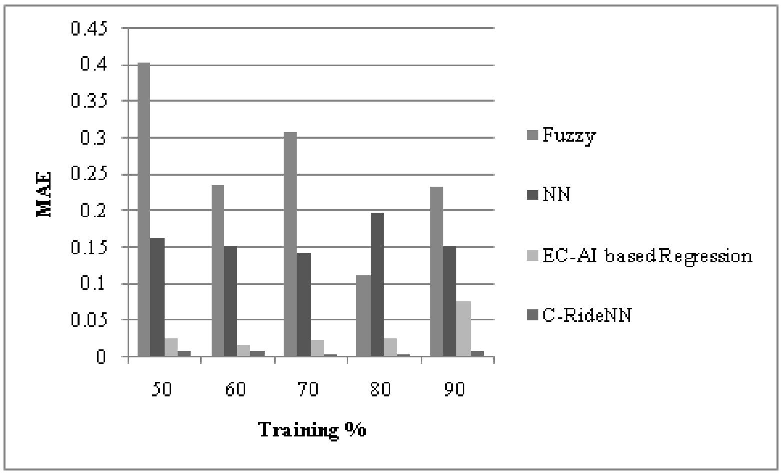
The simulation results of the proposed software reusability scheme is compared with the several works, such as NN (Padhy, *et al.,* 2017a) ,Evolutionary computing enriched artificial intelligence (EC-AI) based regression (Padhy, *et al.,* 2018), Fuzzy(Singh &Tomar, 2017), and User-Perceived Reusability Estimation (Papamichail *et al*., 2018).

Comparative analysis

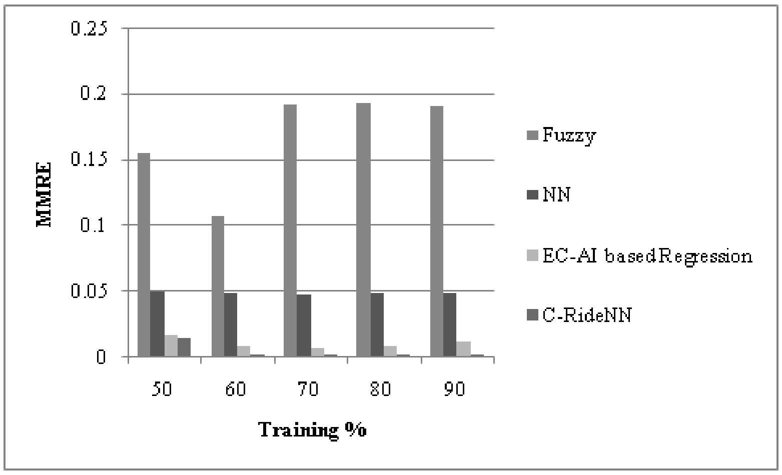
Here, the performance of the proposed C-RideNN algorithm is compared with the three other existing works by varying the value of the holoentropy. In the proposed work, the holoentropy factor is the key element in determining the suitable features for the training, and hence the analysis is done how the holoentropy value affects the performance of each model. The holoentropy value ranges between 4, 5, 6, and 7.

Analysis based on the Holoentropy = 4

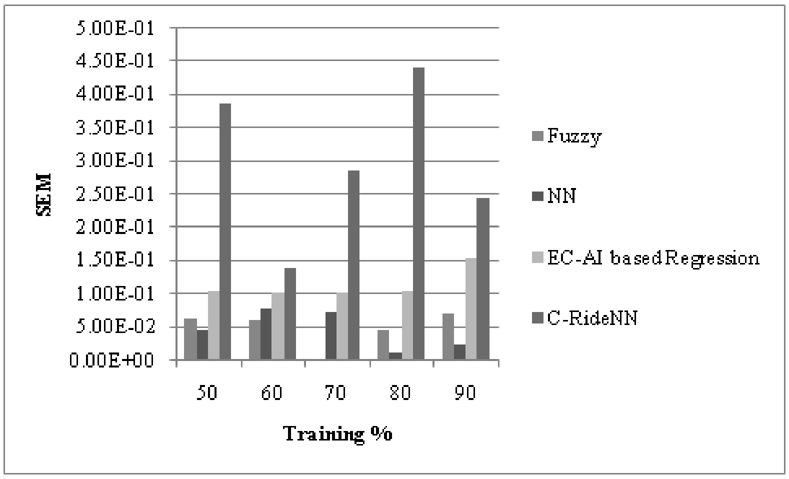
The analysis done here varies the training percentage of the data from the database, and chooses the holoentropy value as 4, and the same is depicted in figure 3. Figure 3.a states the performance of the comparative techniques with the holoentropy value = 4 based on MAE metric. While the training percentage is 50, the existing works, fuzzy, NN, and EC-AI based regression have MAE as 0.4046, 0.1629, and 0.0258. For the same training percentage, the proposed C-RideNN has low MAE value as 0.0078. For the high training percentage of 90, the comparative techniques,fuzzy, NN, and EC-AI based regression achieved the MAE values of 0.2345, 0.1520, and 0.0781, and the proposed C-RideNN has the MAE as 0.0084. Increase in the training percentage has lead to improved performance and also, the error performance is very low. Figure 3.b presents the analysis based on the MMRE metric for the holoentropy value of 4. For the training percentage = 50, fuzzy, NN, and EC-AI based regression have MMRE as 0.1556, 0.0497, and 0.0168. For the same training percentage, the proposed C-RideNN has MMRE value as 0.0146. For the high training percentage of 90, the comparative techniques fuzzy, NN, and EC-AI based regression achieved the MMRE values of 0.1909, 0.0484, and 0.0127, and the proposed C-RideNN has the MMRE as 0.0023. Figure 3.c presents the comparative analysis based on SEM metric for the holoentropy value of 4. For training percentage as 50, fuzzy, NN, and EC-AI based regression have SEM as 0.06145, 0.0476, and 0.1045. For the same training percentage, the proposed C-RideNN has high SEM value as 0.38605. For the high training percentage of 90, the comparative techniques fuzzy, NN, and EC-AI based regression achieved the SEM values of 0.0688, 0.0255, and 0.1533, and the proposed C-RideNN has the SEM as 0.2461.



(a)



(b)

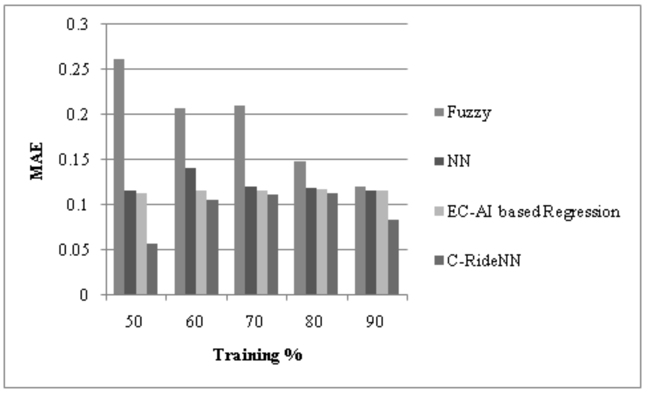


(c)

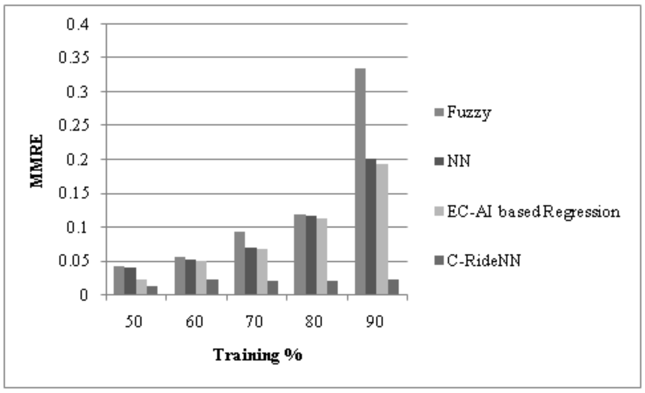
Figure 3. Comparative analysis for the Holoentropy = 4 based on (a) MAE, (b) MMRE, and (c) SEM.

Analysis based on the Holoentropy = 5

Figure 4 depicts the analysis of the comparative methods by varying the training percentage for the holoentropy value of 5. The analysis based on MAE metric is shown in Figure 4.a. Initially, for 50% of the training data, the comparative methods, such as fuzzy, NN, EC-AI based regression, and the proposed C-RideNN have MAE of 0.26109, 0.1168, 0.11353, and 0.0570. For 90% of the training data, the comparative techniques, such as fuzzy, NN, EC-AI based regression, and the proposed C-RideNN achieved the MAE values of 0.1207, 0.116, 0.1165, and 0.0840, respectively. The analysis based on the MMRE metric is shown in Figure 4.b. For 90% of the training data, the existing techniques, such as fuzzy, NN, and EC-AI based regression has the MMRE values of 0.3361, 0.2026, and 0.1944, respectively. On the other hand, the proposed C-RideNN has the MMRE as 0.0232. Figure 4.c shows the comparative analysis based on SEM metric. While seeing Figure 4.c, it can be shown that the best performance is achieved by the proposed method and the worst performance is achieved by the existing fuzzy based technique. For 90% of the training data, the proposed C-RideNN has the SEM of 0.6118, while the fuzzy has the SEM of 0.0840. From Figure 4, it can be seen that the proposed method has the minimum MAE and MMRE, maximum SEM than the existing methods, which means that the proposed method offers better performance than the existing methods.



(a)

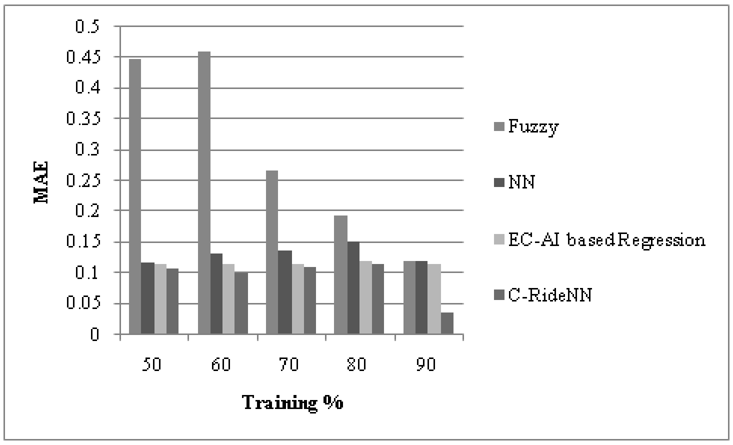


(b)

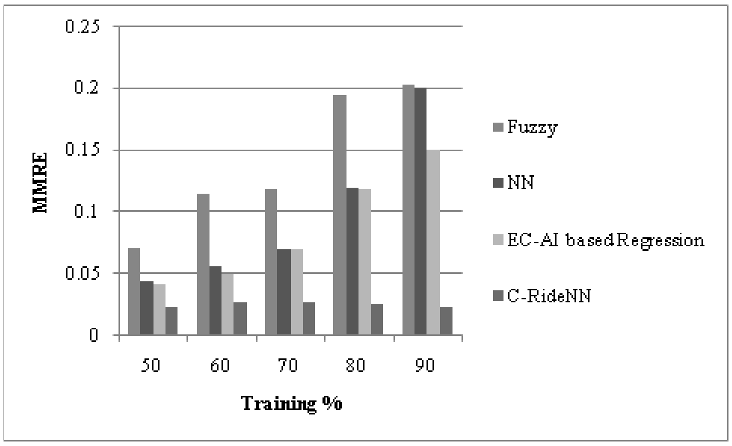
Figure 4. Comparative analysis for the Holoentropy = 5 based on (a) MAE, (b) MMRE, and (c) SEM.

Analysis based on the Holoentropy = 6

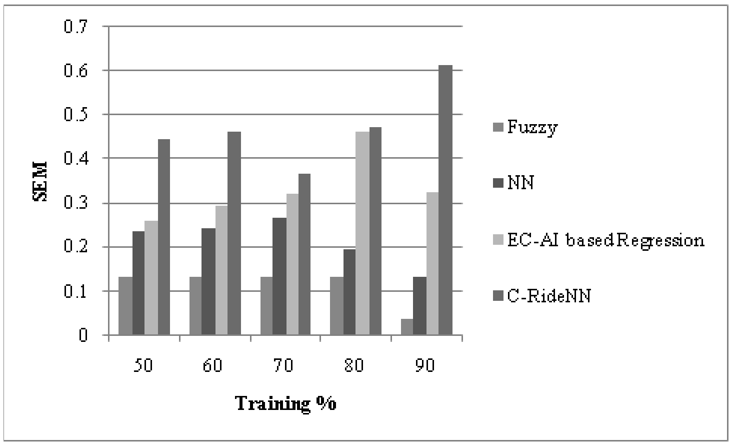
The comparative analysis based on the holoentropy value of 6 is depicted in figure 5. The analysis based on MAE, MMRE, and SEM metrics has been depicted in Figure 5.a, Figure 5.b, and Figure 5.c, respectively. For the training percentage of 90, the MAE and MMRE of the proposed C-RideNN are 0.0376 and 0.02283, which are minimum than the MAE and MMRE of the existing techniques, such as fuzzy, NN, and EC-AI based regression. Similarly, the proposed method has the SEM of 0.6127 for 90% of the training data, which is maximum than the SEM of the other comparative methods.



(a)



(b)

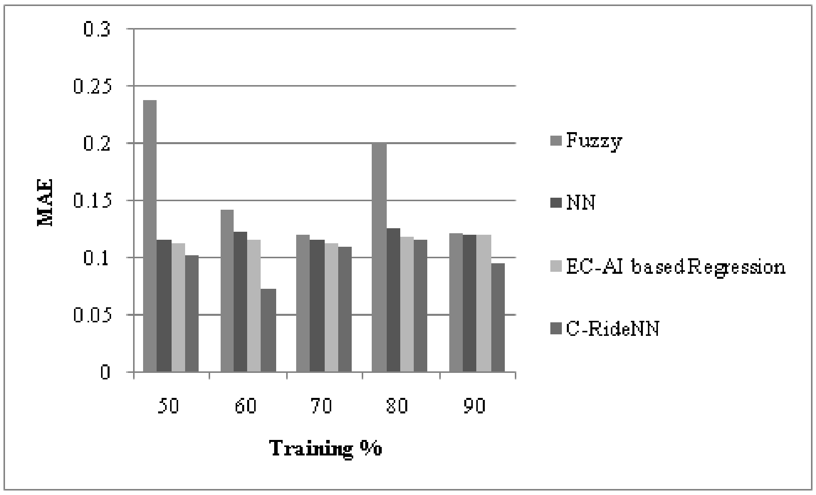


(c)

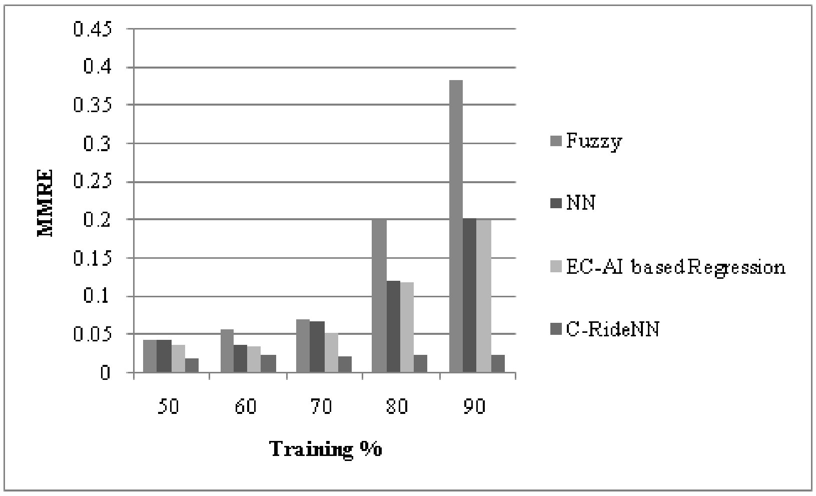
Figure 5. Comparative analysis for the Holoentropy = 6 based on (a) MAE, (b) MMRE, and (c) SEM.

Analysis based on the Holoentropy = 7

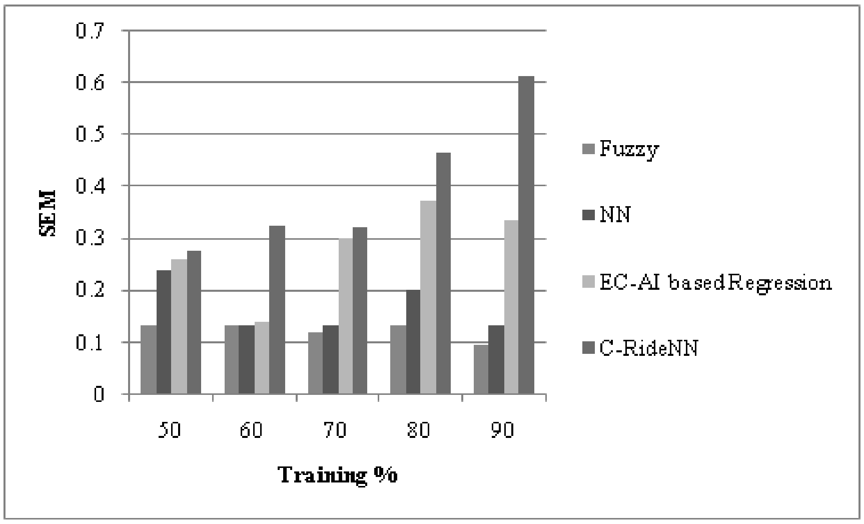
The comparative analysis based on the holoentropy value 7 is depicted in figure 6. Figure 6.a shows the comparative analysis based on MAE metric. For 90% of the training data, the proposed method has the MAE of 0.0957, the fuzzy has the MAE of 0.1216, NN has the MAE of 0.1208, and EC-AI based regression has the MAE of 0.1207. Similarly, the comparative analysis based on the MMRE metric is shown in Figure 6.b. Here also, the proposed C-RideNN has a minimum MMRE than the existing methods. For 50% of the training data, the existing methods, such as fuzzy, NN, and EC-AI based regression have MMRE as 0.044, 0.0436, and 0.0381. Meanwhile, the proposed method has the MMRE of 0.0205248. Figure 6.c shows the comparative analysis based on SEM metric. For training percentage of 50, the proposed method has the SEM of 0.2776, which is 51.22%, 14.05%, and 5.55% maximum than the SEM of the existing methods, such as fuzzy, NN, and EC-AI based regression, respectively.



(a)



(b)



(c)

Figure 6. Comparative analysis for the Holoentropy = 7 based on (a) MAE, (b) MMRE, and (c) SEM.

Comparative discussion

Here, the comparative discussion about the performance of the existing works and the proposed C-RideNN algorithm is presented. The best performance achieved by each technique is discussed in Table 1 based on the evaluation metrics. The metrics MAE and MMRE depicts the deviation from actual performance and hence need to be low for achieving the better performance, while the SEM should be as high as possible.

Table 1. Comparative discussion.

Comparative techniques	**Evaluation metrics**		
	MAE	**MMRE**	**SEM**
Fuzzy (Singh &Tomar, 2017)	0.2610	0.0440	0.095
NN (Padhy, *et al.,* 2017a)	0.1163	0.0417	0.1354
EC-AI based regression (Padhy, *et al.,* 2018)	0.1135	0.0232	0.3359
User-Perceived Reusability Estimation (Papamichail *et al.*, 2018)	0.1546	0.2657	0.3126
Proposed C-RideNN algorithm	0.0570	0.0145	0.6133

From the above discussion, it can be concluded that the performance of the proposed C-RideNN algorithm is comparatively better than the existing works. Using the fuzzy approach for the reusability estimation has yielded poor performance with the values of 0.2610 as MAE, 0.0440 as MMRE, and 0.095 as SEM. The EC-AI based regression based technique achieved comparatively better performance among the existing techniques with the 0.1135 as MAE, 0.0232 as MMRE, and 0.3359 as SEM. The overall improved performance is achieved by the proposed C-RideNN algorithm with the values of 0.0570 as MAE, 0.0145 as MMRE, and 0.6133 as SEM.

CONCLUSION

This work has developed the software reusability estimation model for predicting the optimal reusability of the software. Here, the software reusability is measured by developing the optimization driven RideNN algorithm along with the various software metrics. A total of nine metrics, such as WMC, CBO, NOC, CLOC, MPC, STAT, DIT, RFC, and LCOM are derived from the software, representing the complexity, cohesion, and coupling. After obtaining the necessary features, the log function is applied to the metrics, to get the required features for the reusability factor determination. Here, C-RideNNclassifier is developed for calculating the reusability factor based on the defined metrics. For the experimentation of the proposed C-RideNN algorithm, few softwares are considered from the GitHub database, and the performance is evaluated based on metrics, such as MAE, MMRE, and SEM. For the comparative analysis, existing works, like fuzzy, NN, and EC-AI based regression are considered. The overall improved performance is achieved by the proposed C-RideNN algorithm with the values of 0.0570 as MAE, 0.0145 as MMRE, and 0.6133 as SEM.

REFERENCES

Ahmed, M.A and Al-Jamimi, H.A. (2013). Machine learning approaches for predicting software maintainability: a fuzzy-based transparent model, *IET Software*, **7**(6): 317-326.

Aman, J and Srivastava, S. (2017). A Hybrid Software Evaluation Software Transition Model with reusability Evaluation, *The Northcap University*.

Ampatzoglou, A., Bibi, S., Chatzigeorgiou, A., Avgeriou, P and Stamelos, I. (2018). Reusability index: A measure for assessing software assets reusability,» In Proceedings of the International Conference on Software Reuse, 43-58.

Arisholm E., Briand L. C and Foyen, A. (2004). Dynamic Coupling Measurement for Object-Oriented Software,*IEEE Transactions on Software Engineering*, **30**(8): 491-506.

Binu, D and Kariyappa, B.S. (2018). RideNN: A New Rider Optimization Algorithm-Based Neural Network for Fault Diagnosis in Analog Circuits,*IEEE Transactions on Instrumentation and Measurement*, PP (99): 1-25.

Caldiera, G and Basili, V.R. (2011). Identifying and qualifying reusable software components,*IEEE Software*, **24**: 61-70,

Chong,C.Y and Lee, S.P. (2015). Analyzing maintainability and reliability of object-oriented software using weighted complex network, *Journal of Systems and Software*, **110**: 28-53.

Chu, S., Tsai, P and Pan, J. (2006). Cat Swarm Optimization, Pacific Rim International Conference on Artificial Intelligence, *Trends in Artificial Intelligence*, **4099**: 854-858.

Diamantopoulos, T., Thomopoulos, K and Symeonidis, A. (2016). QualBoa: reusability-aware recommendations of source code components, in IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR), *IEEE*, 488-491.

Fenton, N.E and Kaposi, A.A. (1987). Metrics and software structure, *Information and Software Technology*, **29**(6): 301-320.

Github database,https://github.com/" accessed on August 2018.

Goel, B.M and Bhatia, P.K. (2013). Analysis of reusability of object-oriented systems using object-orientedmetrics, *ACMSIGSOFTSoftw.Eng.* Notes, **38**:1-5.

J., Otto, Vogel-Heuser, Band Niggemann, O. (2018). Automatic Parameter Estimation for Reusable Software Components of Modular and Reconfigurable Cyber-Physical Production Systems in the Domain of Discrete Manufacturing, *IEEE Transactions on Industrial Informatics*, 14(1).

Jalender, B., Dr Govardhan, A and Dr Premchand, P. (2010). A Pragmatic Approach to Software Reuse,*Journal of Theoretical and Applied Information Technology (JATIT)*, **14**(2): 87-96.

Jalender, B., Govardhan, A and Premchand, P. (2014). A Novel approach for classifying software reusable components for upload and download, *International Conference on Contemporary Computing and Informatics (IC3I)*.

Kaur, A., Kaur, K and Pathak, K.(2015). A proposed new model for maintainability index of open source software, Proceedings of 3rd International Conference on Reliability, *Infocom Technologies and Optimization*.

Mane, V.M and Jadhav, D.V. (2017). Holoentropy enabled-decision tree for automatic classification of diabetic retinopathy using retinal fundus images,*Biomedical Engineering/Biomedizinische Technik*, **62**(3): 321-332.

Padhy, N., Singh, R.P and Satapathy, S.C. (2017a). Enhanced evolutionary computing based artificial intelligence model for web-solutions software reusability estimation, *Cluster Computing*, 1-18.

Padhy, N., Singh, R.P and Satapathy, S.C. (2017b). Software reusability metrics estimation: Algorithms, models and optimization techniques, *Computers & Electrical Engineering*.

Papamichail, M., Diamantopoulos, T., Chrysovergis, I., Samlidis, P and Symeonidis, A. (2018). User-perceived reusability estimation based on analysis of software repositories," In Proceedings of IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE), pp. 49-54.

Pizzi, N.J. (2006). A Computational Intelligence Strategy for Software Complexity Prediction, *The IEEE International Joint Conference on Neural Network Proceedings*.

Polepally, V.k and Chatrapati, K.S. (2017). Dragonfly optimization and constraint measure-based load balancing in cloud computing,» Cluster Computing, 1-13.

Rai, A., Choudhury, T., Sharma, S and Ting, K. (2017). An efficient method to predict software quality using soft computing techniques, *3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*.

Selvarani, R and Mangayarkarasi, P. (2017). Modeling of Reusability Estimation in Software Design with External Constraints, *Software Project Management for Distributed Computing*, 3-23.

Singh, A.P and Tomar, P. (2017). Rule-based fuzzy model for reusability measurement of a software component, *International Journal of Computer Aided Engineering and Technology*, **9**(4).

Singh, C., Pratap, A and Singha, A. (2014). An Estimation of Software Reusability using Fuzzy Logic Technique,*5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*.

Sommerville, l. (2011). Software Engineering, *9th edn. Addison-Wesley*, New York.

Thomas, R and Rangachar, M.J.S. (2018). Hybrid Optimization based DBN for Face Recognition using Low-Resolution Images,» Multimedia Research, **1**(1): 33-43.

Yadav, P. (2014). Dimensionality Reduction of Weighted Word Affinity Graph using Firefly Optimization,» International Journal of Engineering Research & Technology, **3**(10): 1324-1327.