International Journal of Advanced Research in Engineering and Technology (IJARET)
Volume 11, Issue 12, December 2020, pp. 2623-2633, Article ID: IJARET_11_12_247
Available online at http://iaeme.com/Home/issue/IJARET?Volume=11&Issue=12
ISSN Print: 0976-6480 and ISSN Online: 0976-6499
DOI: 10.34218/IJARET.11.12.2020.247

SOFTWARE DEFECT DETECTION BY IWD GENETIC FILTER AND NEURAL NETWORK MODEL

Bhagyashri Deshpande
Department of Computer Science, Savitribai Phule Pune University,
Pune, Maharashtra, India

Dr. Binod Kumar
Professor at JSPM'S Rajarshi Shahu College of Engineering,
Pune, Maharashtra, India

Dr. Ajay Kumar
Professor at JSPM'S Rajarshi Shahu College of Engineering,
Pune, Maharashtra, India

ABSTRACT

Human life dependency on digital world increases day by day. Software have trust, easy to understand for implementation and reduces labor cost. But defects in software leads to reduce the trust and may be harmful for the individual, community, organization, etc. Hence software defect detection came in to existence. This paper has resolved this issue of software defect detection by proposed hybrid model of Intelligent water drop genetic algorithm and neural network. Whole work is divided into two module first is feature selection from the dataset by Intelligent water drop algorithm and second is learning of feature set sets for desired output by neural network. Genetic algorithm-based feature selection increases the learning capability of neural network. Experimental work is done on real dataset. Result shows that Software Defect Detection Intelligent Water Drop Neural Network (SDDIWDNN) has improved the software defect detection accuracy as compared to other existing algorithm.

Keywords: Automatic Test Case Generation, Software Testing. Software Defect Detection, Genetic Algorithm, Neural Network

Cite this Article: Bhagyashri Deshpande, Dr. Binod Kumar and Dr. Ajay Kumar, Software Defect Detection by IWD Genetic Filter and Neural Network Model. *International Journal of Advanced Research in Engineering and Technology,* 11(12), 2020, pp. 2623-2633.
http://iaeme.com/Home/issue/IJARET?Volume=11&Issue=12

1. INTRODUCTION

Software programs has become a necessary part of the present computerized world. Software are utilized practically in each field of day-to-day life. Along these lines, a major task in giving exceptionally productive programming is highly important. The nature of any project is incredibly influenced by the presence of defects (Bugs, error) in the modules of whole software. Bugs lessen the quality, trust and utilization of the product. The bug makes the product module work undesirable in a favorable environment [1].

Automatic project imperfection detection is a way, which analyzes whole software modules for identifying the bugs present in the code. These models forecast some of software shortcomings by utilizing AI procedures that gathered a ton of significance features in the previous few years from the testing behavior. They distinguish the high hazard and afterward it will assist with planning a proficient testing plan utilizing least exertion, time, and cost by organizing the more difficult analyze modules [2]. Finding productive strategies for investigating and foreseeing bugs in programming modules is a prime field of examination in the computer algorithm who perform this work worldwide.

Machine learning neural network model are utilized to mine information from alternate points of view and empower engineers to recover valuable data. The AI procedures that can be utilized to distinguish bugs in programming datasets can be bunched. Grouping is an information mining and AI approach, helpful in programming bug forecast. It includes order of programming modules into defect or normal class that is indicated by a bunch of programming unpredictability measurements by using a grouping model that is gotten from before advancement projects information [3]. The measurements for programming multifaceted nature may comprise of code size [4], McCabe's cyclomatic intricacy [5] and Halstead's Complexity [6]. Bunching is a sort of non-hierarchal technique that moves information focuses among a bunch of groups until comparative thing bunches are shaped or an ideal set is obtained. Bunching techniques make suppositions about the informational collection. In the event that that supposition holds, at that point it results into a decent bunch. Be that as it may, it is an inconsequential undertaking to fulfill all suppositions. The combination of various grouping strategies and by shifting info boundaries might be advantageous. Affiliation rule miningis utilized for finding regular examples of various credits in a dataset [7, 8]. The associative classification most of the times provides a higher classification as compared to other classification methods.

Rest of paper is arranged into various sections where second section briefs about the various work done by researcher and third section explains the experimental work done by authors of the paper. Further, paper has shown experimental work in fourth section which shows evaluation parameter-based comparison of software defect detection models. Finally, paper is concluded with different finding of proposed SDDIWDNN model in section five.

2. RELATED WORK

H. Wei in [9] proposed a CDLH model which holds the synthetic and lexical data on the basis of functional characteristics of code in a supervised manner. This paper has converted the real values into binary hash codes. Hence learning of hash codes by neural network increase the efficiency of work as compared to other methods of defect detection. It is also found that proposed model was feasible to work on any length of hash codes as it works efficiently on code of length 8 to 48 digits.

M. White in [10] proposed a software defect detection technique which consider structure of code with an identifier. This paper works on token system which have frequency count of the token and uses the greedy technique to transform the multiple tree structure to recurrent neural network model. Model was able to learn different granule of code for identifying the defects in the projects.

N. Marastoni in [11] proposed a project similarity model that transforms the binary formatted data into two-dimension image structure. This image matrix was passed in the convolutional neural network model for learning and finding the pattern in the image. As code are passed in form of image so classification was done in parts which was a major restriction of the model.

Soumi Ghosha in [12] utilizes a nonlinear project defect detection technique. It was proposed to eliminate undesirable dataset features which are higher irrelevant as per requirement. This dimension reduction of the dataset increases the detection accuracy of the work for enhancing the software quality.

Fredrik Asplund et. al. in [13] improves the life cycle of software testing by study the behavior of the individual software tester. Researcher has found that change in steps of testing and modifying the structure of tester techniques improves the performance of project defect detection.

Tanujit et. al in [14] proposed a new hybrid model for identifying the defect in the software based on the feature set of classes, object, codes, etc. In this paper a Hellinger tree was developed which help in increasing the leaning capacity of feed forward neural network model. Hellinger net model has used a skew intense distance handling class problem. Overall use of tree structure for learning of neural network works well. But feature reduction in this model may further increases the working accuracy.

3. PROPOSED METHODOLOGY

Proposed paper work of software defect detection is done in this section of paper. Whole working of Software Defect Detection Intelligent Water Drop Neural Network (SDDIWDNN)is segment into two modules first is feature selection for defect detection and second is learning of feature set for defect prediction. SDDIWDNN working block diagram is shown in fig. 1. along with explanation in the section.

Feature Selection Module: In this module of proposed work input dataset is pre-process and few columns were select for the training of neural network by Intelligent water Drop Genetic Algorithm [16]. Output feature set act as late design phase parameter for the project. As defects were mainly identified by this phase of the paper.

Pre-processing: Raw dataset columns having text and numeric values, so paper has selected numeric field data for learning. Hence text values were removed from the dataset [17]. Further it was found that data has few cells which are either blank or have some noisy data. So, in this pre-processing step null or noisy values were replaced by numeric value zero. Most of dataset feature values were related to methods, inheritance, code lines, classes, object, etc. So, counting of any feature having null are replaced by Zero.

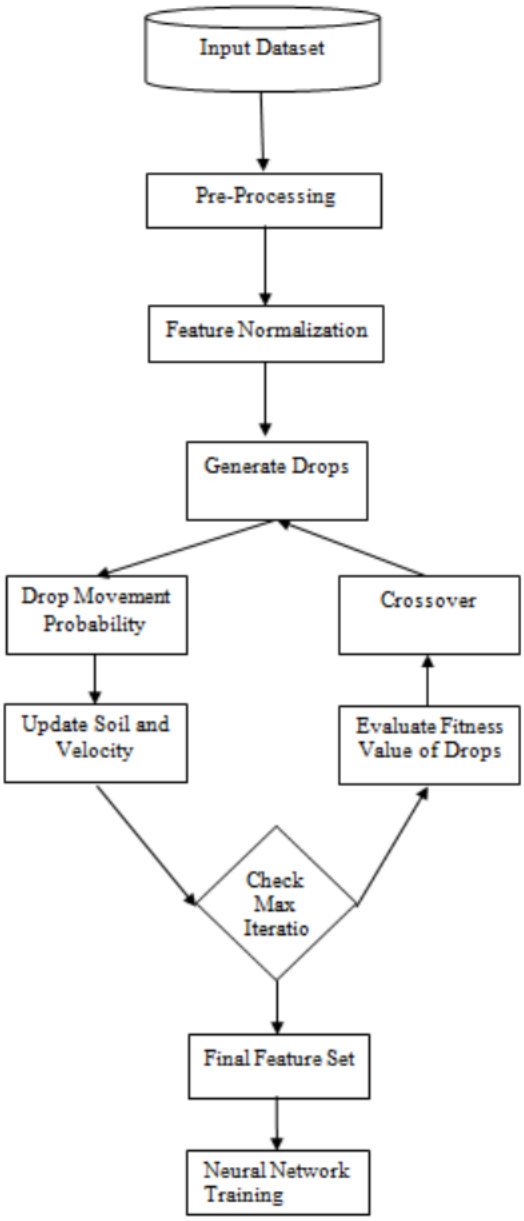

Figure 1 Block diagram of SDDIWDNN.

Feature Normalization: Normalization operation is also performed by the paper by dividing the maximum value count of a feature to each other value of the column. This help to balance the feature values as some of values are in range of 0 to 1 while some are in range of 1 to 100 or 1000. So, balancing between the feature values help to identify feature equally for detecting the defect class of software.

$$ND_f = \left(\frac{PD_d}{Max(PD_f)}\right)_1^d \quad (1)$$

In above eq. 1 ND is normalized dataset having f features and d is size of processed dataset.

IWD (Intelligent Water Drop)

Different combination of feature set lead to high confusion in the system, hence genetic algorithm finds the feasible solution in a smaller number of iterations. Paper has utilized intelligent water drop genetic algorithm for feature selection.

Generate Drops

Data has different features values as per project version, class. Random feature sets were prepared by Gaussian distribution function for finding the feasible solution [18]. Collection of feature sets is termed as population. Each dataset featurein the system isdrop. Drop has a binary vector having two values 1 or 0, where 1 represents a presence of feature and zero represent an absence of feature. So drops D were collection of binary vectors shown in Eq. 2, having f number of feature and n number of drops in population.

$$D \leftarrow \text{Generate\_Drops}(D, f, n) \quad (2)$$

Drop Soil

Distance between the features set were term as soil in the algorithm. Average value of training input feature values was taken as soil feature values. So absolute distance between the feature value set soil is stored in a matrix. As per distance drop movement velocity get affected. This can be understood that if drop choose path of next drop as per lower soil value. Means lesser the resistance more will be the velocity of soil and high chance of drop selection in the final feature set.

Drop Movement Probability

Drop move towards another drop for increasing its strength and soil value affect this selection. Therefore, a term selection probability is shown in Eq. 3 [16]. This selection probability is chance of n^{th} node movement towards N-1 node movement.

$$SP(i,j) = \frac{FS(i,j)}{\sum_{k=1}^{N} FS(i,k)} \quad (3)$$

$$FS(i,j) = \frac{1}{\delta + GS(i,k)}$$

$$GS(i,j) = \begin{cases} soil(i,j) & \text{if } \min(soil(i, \text{all element}) > 0 \\ soil(i,j) - \min(soil(i, \text{all element}) & \text{otherwise} \end{cases}$$

Where i, j are position of feature set in the soil matrix. In Eq. 4 FS is feature selection as per soil and, δ is random number range between 0 to 1.

As movement towards a node may increase or decrease the velocity of drop, so Eq. 5 show this Update velocity formula. Similarly, soil value also get change when drops movement occur by Eq. 6.

$$V(t+1) = V(t) + \frac{V_1}{V_2 + V_3 * Soil(i,j)^2} \quad (5)$$

$$\Delta S(i,j) = \frac{S_1}{S_2 + S_3 * T(t+1)^2} \quad (6)$$

$$T(t+1) = \frac{HD}{V(t+1)} \quad (7)$$

HD is heuristic durability a constant value range in 0-1.

$$Soil(i,j) = (1 - \beta_L) * Soil(i,j) - \beta_L * \Delta S(i,j) \quad (8)$$

S_1, S_2 ,S_3 ,V_1, V_2, 1 constant values having range zero to one.

Fitness Function

Solution feature set feasibility is evaluated by this step of the algorithm. In this step, as per feature set availability training of neural network is done by passing actual values from the dataset. After training set, value set is used for testing and accuracy of trained model stored as fitness value.

Crossover

As per fitness value of the chromosome features were shuffled in other chromosomes for gaining new set of chromosomes. This new chromosome fitness value may be good as other fitness value or poor then parent one. Selection of feature for replacement of the chromosome is done by random function. SDDIWDNN uses Gaussian distribution function for random integer number generation.

Population Updation

As crossover changes the chromosomes of the population so retention of this chromosome depends on fitness value. This can be understood if child chromosome has good fitness value as compared to parent fitness value. Then new child is included in the population, otherwise parent chromosome will continue in population. Hence in all situation population size will never change from P number.

Final Feature Set

After sufficient number of iterations population updates get stopped and fittest chromosome in the population is drag out. As per presence of feature by 1 all corresponding feature vector is used for training, while other feature set which was 0 in the feature set were replace by 0 value. Hence for neural network training input training vector was identified by IWD algorithm and desired outcome is obtained from training dataset having two class 1 for software defect and other is 0.

Training of Neural Network: After sufficient number of training vectors three-layer neuron is prepared for learning the image segmentation by sigmoidal activation function shown in Eq. 9 [19, 20]. Input vector x and output vector o are used for adjusting the weights of layers as per desired output segment class.

$$y = \frac{1}{e^{-xw}} \tag{9}$$

After sufficient number of iterations trained neural network is obtained. For testing neural network accept same set of feature set and predict project Defect/Normal class.

Proposed SDDIWDNN Algorithm

Input RD: //Raw Dataset

Output: TNN// TNN: Trained Neural Network

- PD←Pre-Processing(RD)
- ND←Normalization(PD)
- D←Generate_Drops(D, f, n)
- S←Soil(f, ND)
- Loop 1:itr // itr: Iterations
- [S V]←Drop_Movement_Probablity(S, V)

- F←Fitness_Value(D, S) //F: Fitness Value
- B←Best(F)// B: Best Fitness Chromosome
- CD←Crossover(B, D)//CrossOver Drops
- D←Update_Drops(CD, D)
- EndLoop
- F←Fitness_Value(D, S) //F: Fitness Value
- B←Best(F)// B: Best Fitness Chromosome
- TD←Final_Feature(PD,B) // TD: Training Dataset
- TNN←Neural_Network(TD)

Above SDDIWDNN algorithm takes raw dataset as input and gives trained neural network for software defect detection.

4. EXPERIMENT AND RESULTS

Experimental work is done on MATLAB 2016 software. Implementation of SDDIWDNN of software detection is done on machine having i3 processor of 6th generation with 4 GB RAM. Comparing model Hellinger Netis taken from [14]. This model is also implemented on MATLAB. Results were compared on the basis of dataset having six different project set. Detail description of the dataset is shown in table 1, IC-DePress (Defect Prediction in Software Systems) [15].

Table 1 Detail Description of Software Defect Detection Model.

Parameters	Values
Projects	6
Training Percentage	50%
Testing Percentage	50%
Total Sessions	13533
Feature Set	25
Training Feature	20

4.1. Results

Table 2 Accuracy Based Software Defect Detection Comparison

Projects	SDDIWDNN	Hellinger Net
Ant	0.879	0.8782
Camel	0.8776	0.8563
IVY	0.8568	0.8312
JEdit	0.8826	0.8691
Licene	0.7748	0.6921
POI	0.7722	0.6868

Table 2 and fig. 2 shows that proposed software defect detection model has increased the accuracy of defect detection as compared to other existing algorithm [14]. This improvement in defect detection is achieved by Intelligent water drop genetic algorithm as this has select feature values as per good combination of neural network leaning model. Hence feature reduction by IWD genetic algorithm has increased the detection accuracy of work. It is also shown that SDDIWDNN has average increased the accuracy value defect detection by 4.54% as compared to Hellinger Net [14].

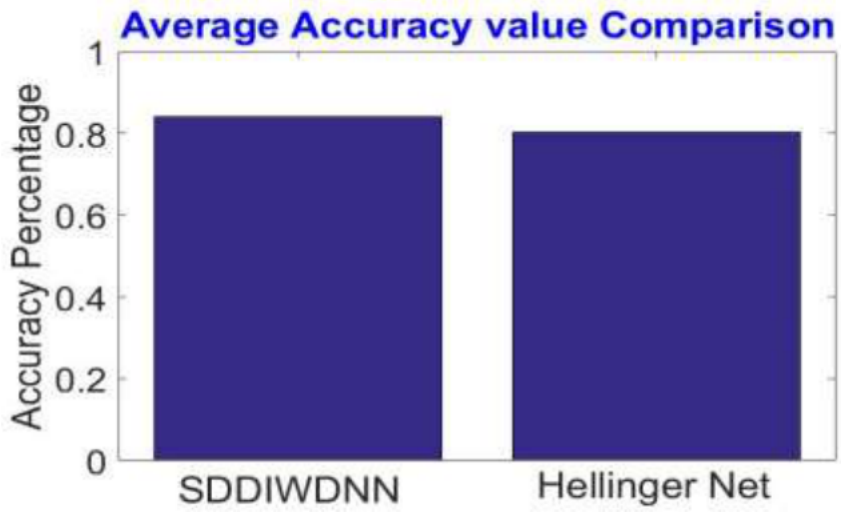

Figure 2 Average Accuracy-Based Comparison of Software Defect Detection Techniques.

Table 3 Precision Based Software Defect Detection Comparison

Projects	SDDIWDNN	Hellinger Net
Ant	0.9925	0.9803
Camel	0.9347	0.9744
IVY	0.9856	0.9849
JEdit	0.98	0.993
Licene	0.8524	0.9898
POI	0.833	0.9948

Table 3 precision values shows that Hellinger Net and SDDIWDNN both have higher values in different project sessions. This high values in Hellinger Net are achieved by specifying most of the project session in one class so overall value of precision in some projects were high.

Table 4 Recall Based Software Defect Detection Comparison.

Projects	SDDIWDNN	Hellinger Net
Ant	0.8833	0.8911
Camel	0.9193	0.8686
IVY	0.8629	0.8344
JEdit	0.8943	0.8729
Licene	0.8111	0.6813
POI	0.8359	0.6889

Table 4 shows that proposed software defect detection model has increased the recall of defect detection as compared to another existing algorithm [14]. This improvement in defect detection is achieved by Intelligent water drop genetic algorithm as this has selected feature values as per good combination of neural network leaning model. Hence feature reduction by IWD genetic algorithm has increased the detection accuracy of work. It is also shown that SDDIWDNN has average increased the recall value defect detection by 7.09% as compared to Hellinger Net [14].

Table 5 F-Measure Based Software Defect Detection Comparison.

Projects	SDDIWDNN	Hellinger Net
Ant	0.9347	0.9336
Camel	0.9269	0.9185
IVY	0.9198	0.9076
JEdit	0.9352	0.9292
Licene	0.8313	0.8071
POI	0.8345	0.8141

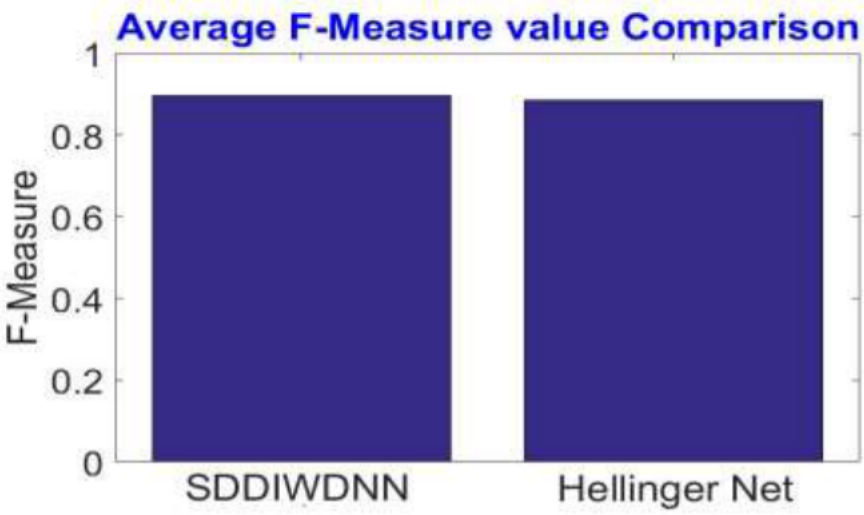

Figure 3 Average F-Measure Based Comparison of Software Defect Detection Techniques.

Table 5 and fig. 3 shows that proposed software defect detection model has increased the F-measure of defect detection as compared to other existing algorithm [14]. This improvement in defect detection is achieved by Intelligent water drop genetic algorithm as this has select feature values as per good combination of neural network leaning model. Hence feature reduction by IWD genetic algorithm has increased the detection accuracy of work. It is also shown that SDDIWDNN has average increased the F-measure value defect detection by 1.34% as compared to Hellinger Net [14].

5. CONCLUSIONS

Software defect detection in early stages of development reduces the testing time and increasing the accuracy of work before deployment. This paper has proposed a genetic algorithm and neural network hybrid model for defect prediction in a software. Use of intelligent water drop for feature reduction by selecting few drops (feature) for training of neural network model has increased the learning of the proposed model. Extracted feature set not only used for training of neural network but also consider as late design phase of the project. Experimental work is done on real dataset having project related feature values. Results were compared with existing methods. It is obtained that proposed SDDIWDNN has increases the accuracy of work by 4.54% and F-measure value is enhanced by 1.34% as compared to Hellinger Net [14]. In future researcher can improve the prediction accuracy by involving other genetic algorithm for reducing the feature set of datasets.

REFERENCES

[1] M. Gayathri and A. Sudha, "Software defect prediction system using multilayer perceptron neural network with data mining," International Journal of Recent Technology and Engineering (IJRTE), vol. 3, no. 2, pp. 54–59, 2014.

[2] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, "A general software defect-proneness prediction framework," IEEE Transactions on Software Engineering, vol. 37, no. 3, pp. 356 370, 2011.

[3] Shiwei, Deng (2009) "Defect prevention and detection of DSP-Software", World Academy of Science, Engineering and Technology, Vol. 3, Issue 10, pp. 406-409.

[4] Trivedi, Prakriti& Pachori, Som (2010) "Modelling and analyzing of software defect prevention using ODC", International Journal of Advanced Computer Science and Applications, Vol. 1, No. 3, pp. 75- 77.

[5] Nair, T.R. Gopalakrishan & Suma, V. (2010) "The pattern of software defects spanning across size complexity", International Journal of Software Engineering, Vol. 3, Issue 2, pp. 53- 70.

[6] Lessmann, Stephen., Baesens, Bart., Mues, Christopher., & Pietsch, Swantje (2008) "Benchmarking classification models for software defect prediction: A proposed framework and novel finding", IEEE Transaction on Software Engineering, Vol. 34, Issue 4, pp. 485-496.

[7] R. Chillarege and D. P. Siewiorek, "Experimental evaluation of computer systems reliability,'' IEEE Trans. Reliability, pp. 403-408, vol. 39, Oct. 1990.

[8] Chillarege, Ram & Bhandari, I.S. & Chaar, Jarir & Halliday, M.J. & Moebus, D.S. & Ray, Bonnie & Wong, M.-Y. (1992). Orthogonal Defect Classification - A Concept for In-Process Measurements. Software Engineering, IEEE Transactions on. 18. 943 - 956. 10.1109/32.177364.

[9] H. Wei and M. Li, "Supervised deep features for software functional clone detection by exploiting lexical and syntactical information in source code," in Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, pp. 3034–3040, Melbourne, Australia, August 2017.

[10] M. White, M. Tufano, C. Vendome et al., "Deep learning code fragments for code clone detection," in Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, pp. 87–98, ACM, Singapore, September 2016.

[11] N. Marastoni, R. Giacobazzi, and M. Dalla Preda, "A deep learning approach to program similarity," in Proceedings of the 1st International Workshop on Machine Learning and Software Engineering in Symbiosis, pp. 26–35, ACM, Montpellier, France, September 2018.

[12] Soumi Ghosha, Ajay Ranab, Vineet Kansal. "A Nonlinear Manifold Detection based Model for Software Defect Prediction". International Conference on Computational Intelligence and Data Science (ICCIDS 2018).

[13] Asplund, F.. Exploratory testing: Do contextual factors influence software fault identification? Information and Software Technology, 107, 101-111, 2019.

[14] Tanujit Chakraborty and Ashis Kumar Chakraborty. "Hellinger Net: A Hybrid Imbalance Learning Model to Improve Software Defect Prediction". IEEE Transactions On Reliability, 2020.

[15] Lech Madeyski, Marian Jureczko, "Which Process Metrics Can Significantly Improve Defect Prediction Models? An Empirical Study.", Software Quality Journal, vol. 23, no. 3, pp. 393–422, 2015.

[16] Saket Jain, Dr. Rajendra Gupta. (2020). Relevant Image Fetching Using IWD Algorithm with Co-occurrence Matrix and Annotation Features. International Journal of Advanced Science and Technology, 29(7), 12521 - 12535.

[17] NJ Mohammed. "Neural Network Training by Selected Fish Schooling Genetic Algorithm Feature for Intrusion Detection". International Journal of Computer Applications, Nov, 2020.

[18] Ashwani Mathur *"Hybrid Combination of Error Back Propagation and Genetic Algorithm for Text Document Clustering"* International Journal of Computer Trends and Technology 68.11 (2020):64-68.

[19] Bharot, N., Verma, P., Sharma, S. et al. Distributed Denial-of-Service Attack Detection and Mitigation Using Feature Selection and Intensive Care Request Processing Unit. Arab J Sci Eng 43, 959–967 (2018).

[20] Arora and A. Saha, "Comparison of back propagation training algorithms for software defect prediction," *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, Noida, 2016, pp. 51-58,