

How to Cite:

Balamurugan, R., & Ravichandran, M. (2022). Enhanced convolutional neural network based android software maintainability prediction. *International Journal of Health Sciences*, 6(S4), 6391–6403. <https://doi.org/10.53730/ijhs.v6nS4.9616>

Enhanced convolutional neural network based android software maintainability prediction

Balamurugan. R

Department of Computer Science, Sri Ramakrishna Mission vidyalaya College of Arts and Science, Coimbatore, Tamilnadu

Ravichandran. M

Department of Computer Science, Sri Ramakrishna Mission vidyalaya College of Arts and Science, Coimbatore, Tamilnadu

Abstract---Advancement in mobile technology, necessitates quality in development of android applications. To improve the quality of android application, evaluation of its maintenance has to be started from the initial stage of development. There are several existing models involved in android application maintenance evaluation, but most of them suffers from the class imbalance problem and its performance depends on the size of the dataset. This paper focuses on overcoming the class imbalance while using convolutional neural network for prediction the android application maintainability. In this work both android metrics and object-oriented metrics are used for evaluating the android application maintainability. The standard convolutional neural network is enhanced by pretraining it using evolutionary algorithm known as moth flame optimization (MFO). The MFO involves in optimizing the learning parameters of convolutional layer and after the training process is completed, with the acquired knowledge it handles the small size dataset very effectively during the testing phase. Thus, the simulation results of maintainability prediction in android application development using proposed enhanced CNN achieves better result while comparing with other standard classification models.

Keywords---android application maintainability, convolutional neural network, moth flame optimization, learning parameter, pretraining.

Introduction

The ease with which a software system or component may be adjusted to rectify flaws, improve performance or other features, or adapt to a changing environment is referred to as software maintainability [1]. The software must be updated to

satisfy the changing needs of consumers, which may happen for a variety of reasons, including technological advancements, the introduction of new hardware, or the development of existing functions [2]. It's not only unfeasible, but also incredibly expensive to create software that never has to be updated. Android application maintenance is the process of changing software that has already been delivered. Software maintenance consumes far more resources, effort, and time than its development.

The amount of money, effort, and time invested on software maintenance far outweighs the amount spent on its creation. As a result, designing android application that is simple to maintain might potentially save a lot of money and time [3]. Monitoring android metrics throughout the development process is one of the key techniques to controlling maintenance costs. Researchers are interested in measuring many aspects of software architecture, such as inheritance, coupling, cohesiveness, and predicting maintenance activity beliefs [4].

The challenge of anticipating android application maintainability is well-known in the industry, and much has been published on how maintainability may be anticipated using various tools and methods throughout the design phase utilizing software design metrics. In this paper an enhanced model of convolutional neural network is developed to optimize the process of android application maintainability prediction, even when the size of dataset is small. The pretrained model is introduced in this work which fine tunes the hyperparameters of the convolutional neural network by adopting moth flame optimization algorithm, which well balances the training phase learning rate more precisely in android application maintainability prediction.

Related Work

Ahmad et al [5] conducted a depth investigation on android mobile applications maintainability detection using android and object-oriented metrics. For calculating the maintainability measure of each application regression algorithm is applied. This work discovers the impact of each metrics in both objectoriented and android measures. The highly informative metrics alone are used for predicting maintainability. Parita Jain et al [6] in their work construed a portable application to ensure the quality of mobile application and the factors which highly influence the process of maintainability evaluation is detected. They applied elimination and choosing exploring reality as strategy for assessing the importance of the maintainability metrics using multi-criteria decision making.

Jain et al [7] in tier work suggested a method that combines the Random Forest with three popular feature selection techniques namely Chi-Squared, RF, and Linear Correlation Filter, as well as a re-sampling strategy to increase the fundamental RF algorithm's prediction accuracy. The performance of Enhanced-RFA is assessed using R on two commercially accessible datasets, QUES and UIMS. Sharma et al [8] constructed techniques to determine important factors and its subfactors which supports the reliability of mobile application. With the selected factors and sub-factors, the versatility of mobile applications can be assessed. By predicting dependability of factors, the quality of mobile applications

can be easily evaluated. Song et al [9] devised a fault detection algorithm using uncertainty model. The Non-homogeneous Poisson process is applied on real time software failure dataset to predict the reliability of fault detection. The newly developed algorithm is compared with other state of arts.

Malavolta et al [10] analyzed about different android application maintainability process and detecting life time of it. They evaluated android apps with more than 9,945 weekly snapshots for understanding and categorizing the level of maintainability. NarimaneZighed et al [11] gave a complete taxonomy of Object-Oriented software maintainability prediction models and conduct a comparison study in this work. In addition, they focused on three different angles: architecture, design, and code. Furthermore, this research points to some fundamental principles for measuring maintainability, with the understanding that maintainability would be measured differently at each level.

Nanda et al [12] in their work gathered different object-oriented metrics and then the most significant metrics/features that impact maintainability in response to Changes were discovered using five attribute selection strategies and three machine learning techniques. Furthermore, the findings were compared to previous studies, and it was discovered that picking only the important characteristics for prediction improved the outcomes. be assessed in a different way

Jain et al [13] designed an evolutionary model using genetic algorithm to predict the software maintainability and compared its performance with many other existing classification algorithms. The open source software models are involved in collection of datasets and the effort of maintenance is computed based on the number of changes done on each module or software. Zhu et al [14] developed a software reliability model using novel non-homogenous process, its functionality is to predict the fault dependency and independency. The debugging process works according to the type of fault presented in the software.

Proposed Methodology

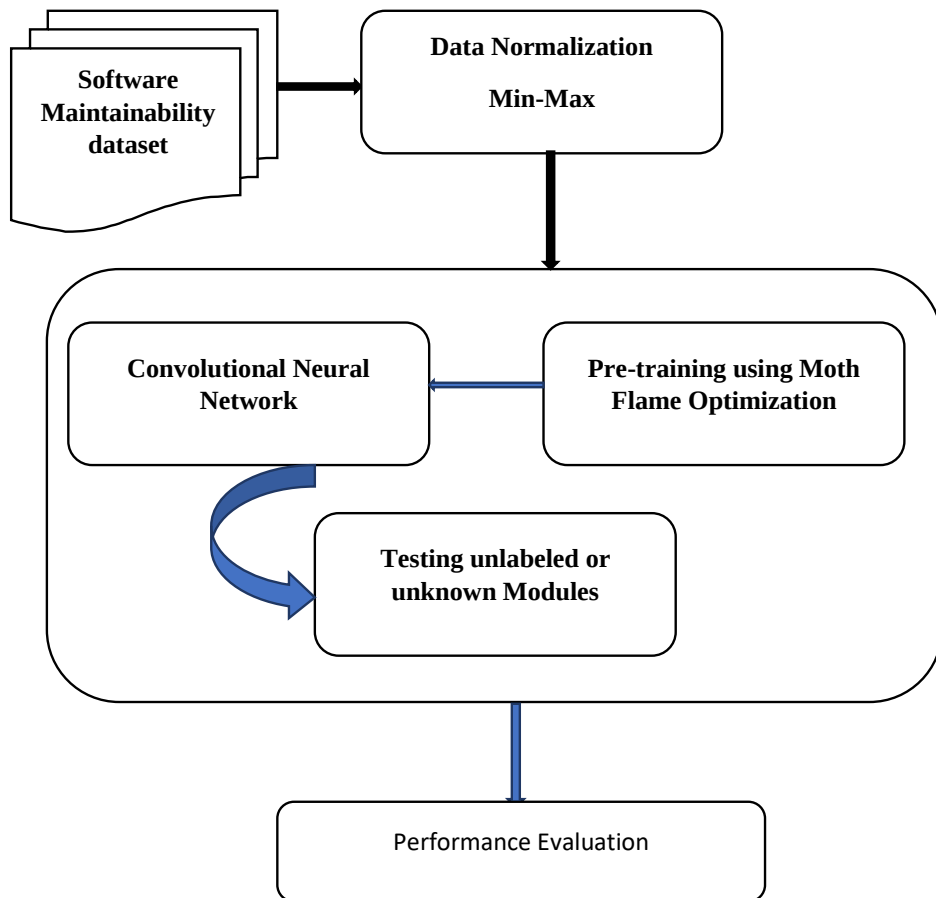


Figure 1: Workflow of the enhanced CNN based Android application maintainability prediction

Maintainability is a critical quality feature that must be considered while developing android applications. This phase focuses on developing a maintainability predictive model which integrates the convolutional neural network and moth flame optimization for improving the quality of the android applications as shown in the figure 1. The convolutional neural networks lack its performance when the class imbalance is very during the training phase of maintainability prediction. Hence, in this work moth flame optimization as a pretrained model which improves the learning accuracy of CNN to achieve better result in android maintainability prediction. This enhanced CNN focuses on analyzing and measuring the maintainability of Android mobile applications using Object Oriented metrics and Android Metrics on the Android application's maintainability. It assesses the impact all of these metrics on the maintainability of Android applications and choose the metrics that has the highest impact. The dataset used for evaluating the maintainability of android application is collected from 776 class hierarchies from 20 different Java open source projects [15] is

considered with 12 different design level metrics as the independent variables and modifiability and understandability metrics as the dependent variables.

Min-Max Normalization

In this work for predicting the android software maintainability QUES dataset is used for object-oriented metrics and android metrics. The dataset comprised of 12 different metrics with 20 different java source code projects. The range of value of all metrics is converted to same range to treat with equal consideration. The formula used for normalization is min-max which is represented as shown in the below equation

$$MN - MX = \frac{att - min}{max - min}$$

Where att is the metric value of class and min and max are minimum and maximum value of the concern metric.

Convolutional Neural Network for maintainability prediction of Android applications

A Convolutional Neural Network (CNN) is a Deep Learning algorithm that can feed in data, assign significance to different characteristics in the android app maintainability dataset, and differentiate between them [16]. The amount of pre-processing required by a ConvNet is much less than that required by other classification techniques. ConvNets can learn these filters/characteristics with adequate training, whereas simple techniques need hand-engineering of filters. A convolutional layer, a pooling layer, and a fully connected layer are the three layers that make up a CNN.

- Convolutional Layer: The CNN's main building component is the convolution layer. It is
- responsible for the majority of the network's computational requirements. This layer does a dot product over two matrices, one of which is a collection of trainable parameters termed as a kernel, while the other is the confined part of the receptive field.
- Pooling Layer: This layer uses a reduction in the quantity of neighboring outputs to substitute the network's output at specific spots. This reduces the representation's spatial size, which reduces the amount of computation and weights necessary. Every chunk of the representation is independently treated throughout the pooling procedure.
- Fully Connected Layer: As in conventional FCNN, neurons in this layer have complete connection with all neurons in the previous and following layers. It seems that a matrix multiplication followed by a bias effect would be used to compute it. It aids in the mapping of the input and output representations.

While using standard Convolutional Neural Network, it requires huge volume of dataset during training process to understand the pattern of input in depth. It is not suited for small size dataset and for unknown patterns of instances its performance is not satisfactory. To overcome this problem, the CNN is pretrained in this proposed work by adopting the moth flame optimization algorithm, for

optimizing the learning parameters of CNN and making it suited even for small datasets of android application maintainability prediction. The detailed working principle of moth flame optimization is explained in the following section

Moth flame optimization Algorithm for Pre-Training CNN

The nightly flying behaviour of moths was the inspiration to develop moth flame optimization algorithm [17]. They learnt to fly by moonlight at night. For navigation, moths developed a transverse orientation mechanism. For long-distance travel, they maintained a steady angle with the moon. Because the moon is so far away from the moth, this process assures that the moth will travel in a straight line. Moths, on the other hand, have been seen to fly in a spiral around lights in general. This is due to the fact that they are fooled by man-made artificial lights that attempt to fly in a straight line at a similar inclination. The Moth Flame may be seen searching is depicted in figure 2.

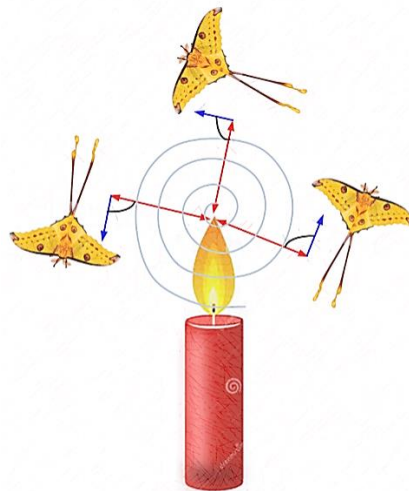


Figure 2: Moth Flame Optimization

In the Moth-Flame Optimization method, each candidate solution is supposed to be a moth, and the problem variables are represented as moth locations in space. Moths and flames are both considered solutions in the MFO algorithm because it is a population-based method [18]. Flames are seen as the best position of moths found thus far by the moths, who operate as a search agent that travels throughout the search space. As a consequence, every Moth search for a flame and updates it whenever a better alternative becomes available, guaranteeing that it never misses out on optimization. Moths are employed as a matrix in this research to identify the best values for fine tweaking the parameters in the convolution layer during the training phase. The number of moths is denoted by n , while the control variables are denoted by d . Artificial moths' matrices are expressed mathematically as shown in the equation below :

$$T = \begin{pmatrix} t_{1,1} & \cdots & t_{1,d} \\ \vdots & \ddots & \vdots \\ t_{n,1} & \cdots & t_{n,d} \end{pmatrix}$$

Each moth fitness value is shown in the below equation.

$$V = \begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{pmatrix}$$

The moth adjusts its location in relation to the flame during the search process to prevent local optimization and to revise the optimal position using its global exploring capacity. Because moths and flames share the same search space, the flame matrix is as follows.

$$F = \begin{pmatrix} f_{1,1} & \cdots & f_{1,d} \\ \vdots & \ddots & \vdots \\ f_{n,1} & \cdots & f_{n,d} \end{pmatrix}$$

The flames' fitness value is theoretically represented as

$$O = \begin{pmatrix} o_1 \\ o_2 \\ \vdots \\ o_n \end{pmatrix}$$

Because the moth searches among individuals inside their search space, the flame is the optimal position, which is iteratively adjusted and updated by the moth. Based on the position of the moth, the mathematical depiction of flame flying behaviour is updated.

$$T_i = H(T_i - F_j)$$

where T_i means i^{th} moth, F_j denotes of the j^{th} flame and the helical purpose is symbolized as H . The flight path of the moth to search the best parameter values which influence the prediction process of CNN based on the pattern of object oriented and android metrics is optimized using the equations

$$\begin{aligned} H(T_i, F_j) &= s_i * e^{hd} * \cos(2\pi d) + F_j \\ d &= (k - 1) * rn + 1 \\ k &= -1 + it * \left(-\frac{1}{T_{mx}}\right) \end{aligned}$$

Where linear distance among moth i and flame j is denoted using ' s_i ', helix shape is signified as h and path coefficient is represented using d . The moth is considered as closest to the frame when if the value of $v = -1$ and if $v = 1$ then it is believed that moth is far away from the flame. The distance p_i is calculated as shown as $s_i = F_j - H_i$. As a result, throughout the CNN learning process, the moth seeking behaviour towards flame light fine-tunes the learning parameters in an optimal way.

The figure 3 illustrates the process of pretrained CNN enhanced by moth flame searching behavior which optimizes the process of android application maintainability prediction.

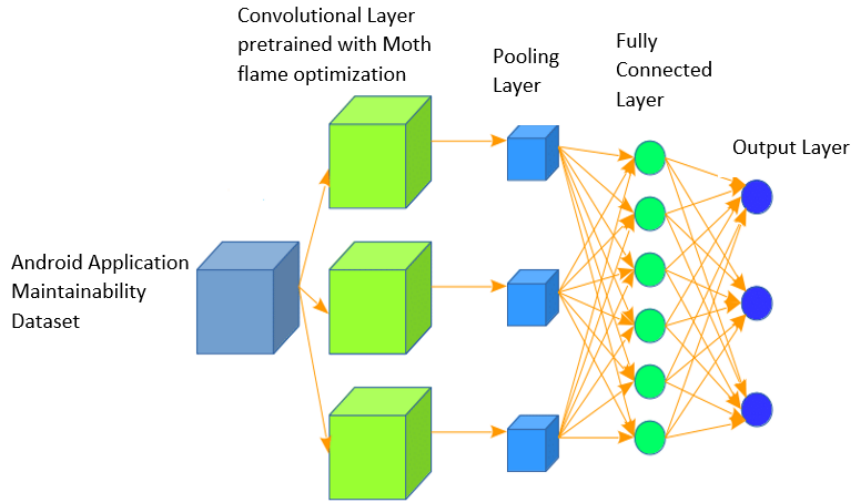


Figure 3: Moth flame optimization pretrained Convolutional Neural Network

Experimental Results

In this section the detailed description about the performance of the newly constructed Enhanced Convolutional neural network for android application software maintainability prediction. The dataset used to assess the maintainability of an Android application is made up of 776 class hierarchies from 20 distinct Java open source projects, with 12 different design level metrics as independent variables and modifiability and understandability metrics as dependent variables. The proposed ECNN is implemented using python software. The existing state of arts used for performance assessment are Artificial Neural Network, Multilayer perceptron and standard convolutional neural network.

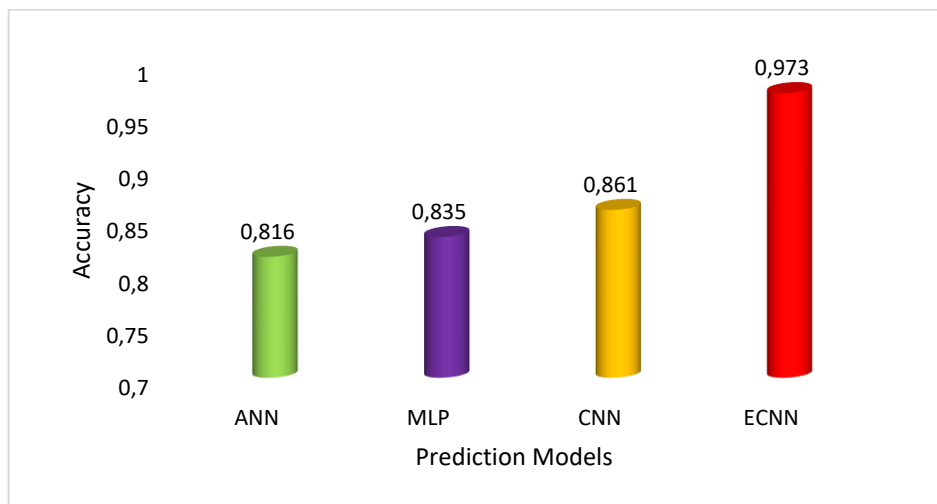


Figure 4: Comparison based on Accuracy

Figure 4 explores the efficiency of the newly devised enhanced convolutional neural network (ECNN) for predicting the maintainability of the android applications based on accuracy. Predicting the maintainability of android applications is accomplished prominently by the convolutional neural network by improving its learning rate. The accuracy of the ECNN has achieved highest value compared to other existing algorithms Artificial Neural Network, Multiple Layer Network and standard conventional neural network. The Moth Flame pretrained model is used in CNN training phase to understand in depth about the pattern of dataset and assess the maintainability level more successfully. The class imbalance problem is overcome by the proposed ECNN while comparing with ANN, MLP and standard CNN for android applications maintainability prediction.

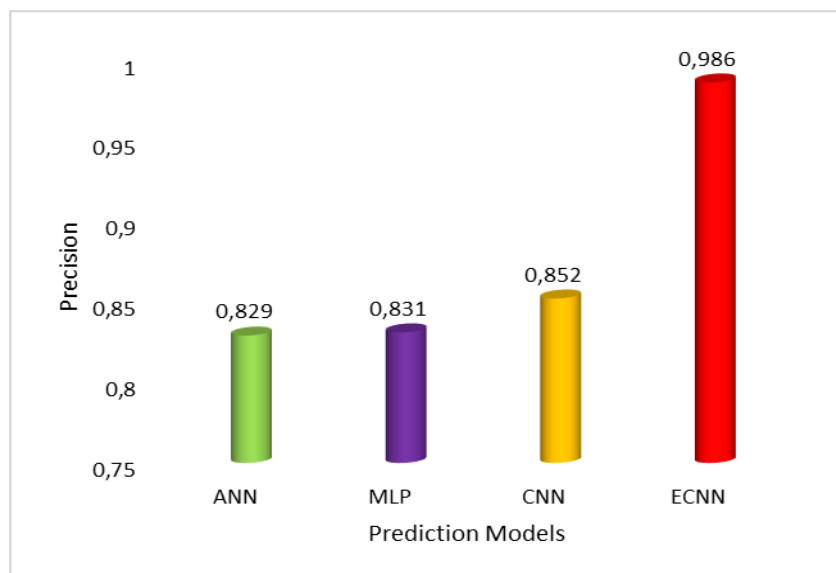


Figure 5 comparative analysis based on precision

Figure 5 demonstrates the importance of training phase for classification model which learns and understands the pattern of incoming records with labels by comparing four different prediction models used for android application maintainability prediction. The testing phase suffers from high class imbalance due to limited instance of low-level maintainability of android applications. The proposed Enhanced Convolutional Neural Network improves its learning process by adopting moth flame behavior, its light source searching behaviour is adapted to improving the learning rate of conventional CNN and well balances both training and testing datasets effectively.

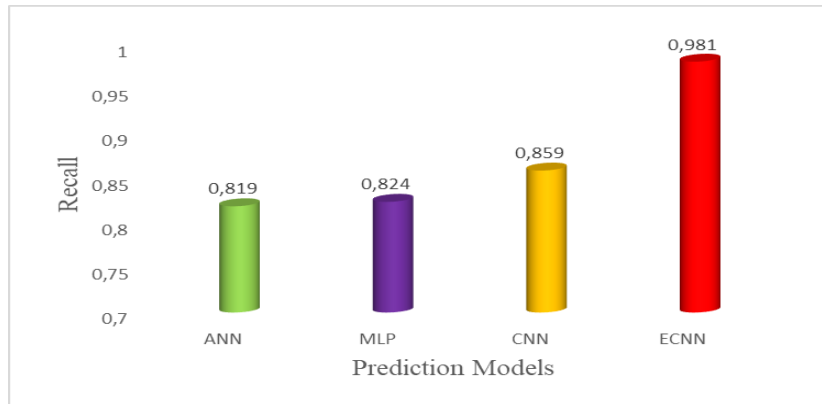


Figure 6: Comparative analysis based on Recall

The recall measure of four prediction models involved in android application maintainability prediction is shown in figure 6. Both object-oriented metrics and android metrics are used for maintainability prediction. The performance of ECNN is better than standard CNN, MLP and ANN because it perfectly tunes the assignment of parameters implicated in the classifier using moth flame optimization to optimize learning rate. Thus, the recall rate of ECNN is higher than the other existing models.

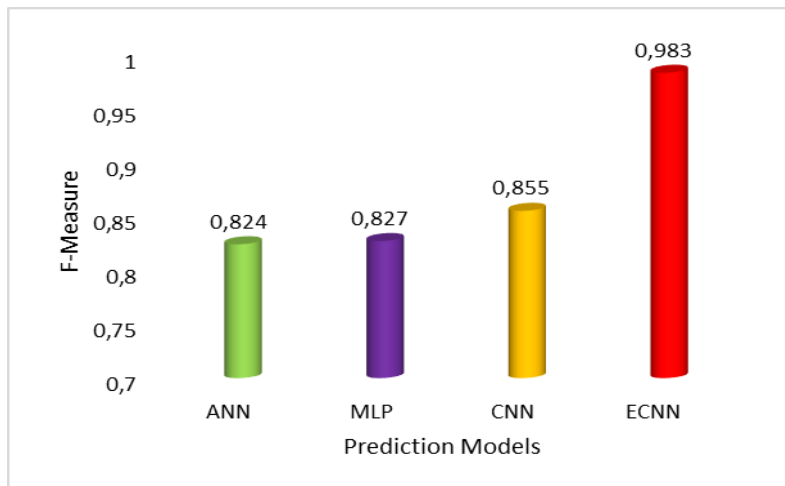


Figure 7: Comparison based on F-Measure

The F-measure of proposed ECNN in android application maintainability prediction is higher than the other state of arts ANN, MLP and CNN as illustrated in figure 7. The nature of moth flame is to find best positions to discover the variation in input patterns and stimulate the corresponding nodes in the convolutional neural network. The fully connected layer of CNN is improvised with the knowledge of Moth Flame optimization algorithm to produce better maintainability prediction on android applications. The moth flame work as a pretrained model that are trained on a large dataset by truncating existing softmax layer of CNN to handle the problem of class imbalance and handling small dataset to avoid overfitting.

Conclusion

In this work, the problem in optimized maintainability prediction of android application due to class imbalance is well handled by introducing the evolutionary algorithm pretrained convolutional neural network. For effective measure of maintainability, it is very essential to understand the effectiveness of each applications source code. In this work to evaluate the maintainability of android applications both object-oriented metrics and android metrics are used for evaluation. In conventional CNN, it needs huge volume of dataset to produce better result. But in real time, while developing software, it is not possible to always have historical information of previous information. Thus, using CNN for small size dataset leads to overfitting problem and its accuracy will not be satisfactory. The existing drawback in standard CNN is subdued by adopting moth flame optimization as the pretraining model, which involves in handing the assignment of learning parameter more appropriately. The searching behavior of moths toward the light source and selection of best position are the major factors involved in fine tuning the parameters of the CNN. During the testing phase, the fully connected layer, with the acquired knowledge it prominently detects the maintainability of android application. The experimental results proved that the enhanced CNN improves the quality of android application development by evaluating the maintainability during initial stage of android application engineering.

References

1. D. Bijlsma, M. A. Ferreira, B. Luijten, J. Visser, Faster issue resolution with higher technical quality of software, *Software quality journal*, vol. 20, no. 2, pp. 265–285, 2012.
2. Alannsay, M.O. and Tian, J., “Measurement and Prediction of SaaS Reliability in the Cloud”, 2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Vienna, Austria, pp. 123-130, 2016.
3. F. Hanandeh, A. A. Saifan, M.Akour, N. Al-Hussein, K. Shatnawi (2017). Evaluating Maintainability of Open Source Software: A Case Study, *International Journal of Computer Science and Information Security*. 15(2): 411-429.
4. Jorayeva, M.; Akbulut, A. Catal, C.; Mishra, A. Machine Learning-Based Software Defect

5. Prediction for Mobile Applications: A Systematic Literature Review. *Sensors* 2022, 22, 2551.
6. Ahmad A. Saifan, Areej Al-Rabadi, Evaluating Maintainability of Android Applications, 2017 8th International Conference on Information Technology (ICIT, 978-1-5090-6332-1)
7. Parita Jain, Puneet Kumar Aggarwal, Mobile Phone Maintainability Prediction using MCDM Methodology, *International Journal of Recent Technology and Engineering (IJRTE)* ISSN: 2277-3878, Volume-8 Issue-3S2, October 2019
8. P. Jain, A. Sharma, and L. Ahuja, "Software Maintainability Estimation in Agile Software Development" *International Journal of Open Source Software and Processes (IJOSSP)*, Vol. 9, Issue 4, Article 4, pp. 65-78, 2018
9. A. Sharma, Tulika, Recognition of Distinctive Reliability Leading Factors for Mobile Applications, *Turkish Journal of Computer and Mathematics Education* Vol.12 No. 4 (2021), 488-493
10. Song, K.Y, ChangI.H, PhamH, NHPP Software Reliability Model with Inflection Factor of the Fault Detection Rate Considering the Uncertainty of Software Operating Environments and Predictive Analysis. *Symmetry* 2019, 11,521.
11. I. Malavolta, R. Verdecchia, B. Filipovic, M. Bruntink, P. Lago, How Maintainability Issues of Android Apps Evolve, *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2018, pp. 334-344.
12. NarimaneZighed, Nora Bounour, Abdelhak-Djamel Seriali, Comparative Analysis of Object-Oriented Software Maintainability Prediction Models, *Foundation of Computing and Decision Sciences*, Vol. 43 (2018) No. 4
13. Nanda S., Saxena S.G., Bala A.G.: Evaluation of Feature Selection Techniques for SoftwareMaintenance Prediction. Thapar University, India, (2017).
14. Jain A., Tarwani S., Chug A.: An Empirical Investigation of Evolutionary Algorithm forSoftware Maintainability Prediction. In *Students' Conference on Electrical, Electronicsand Computer Science (SCEECS)*, pp.1-6, IEEE, India, (2016).
15. Zhu M, Pham, H. A two-phase software reliability modeling involving with software fault dependency and imperfect fault removal. *Comput. Lang. Syst. Struct.* 2018, 53, 27–42
16. A. Albeladi, R. Abdalkareem, F. Agwaeten, K. Altoum, Y.Bennis, Z. Nasereldine., Toward software measurement and quality analysis of MARF and GIPSY case studies, 2014.
17. Valueva, M.V.,Nagornov, N.N.,Lyakhov, P.A.; Valuev, G.V.,Chervyakov, N.I. (2020). Application of the residue number system to reduce hardware costs of the convolutional neural network implementation". *Mathematics and Computers in Simulation*. Elsevier BV. 177: 232–243.
18. S. Gope, S. Dawn, A. K. Goswami and P. K. Tiwari, "Moth Flame Optimization based optimal bidding strategy under transmission congestion in deregulated power market," *2016 IEEE Region 10 Conference (TENCON)*, 2016, pp. 617-621,
19. . Mirjalili, 'Moth-Flame optimization algorithm: A novel nature-inspired heuristic paradigm,' *Knowledge Based System.*, vol. 89, pp. 228_249, Nov. 2015.
20. Rinaritha, K., & Suryasa, W. (2017). Comparative study for better result on query suggestion of article searching with MySQL pattern matching and

- Jaccard similarity. In *2017 5th International Conference on Cyber and IT Service Management (CITSM)* (pp. 1-4). IEEE.
21. Rinarta, K., Suryasa, W., & Kartika, L. G. S. (2018). Comparative Analysis of String Similarity on Dynamic Query Suggestions. In *2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)* (pp. 399-404). IEEE.
 22. Halapiry, J., Ramadany, S., Sanusi B, Y., Made, S., Stang, S., & Syarif, S. (2020). Children's midwifery learning media application about early detection of android-based growth in improving midwifery students skills. *International Journal of Health & Medical Sciences*, 3(1), 153-159. <https://doi.org/10.31295/ijhms.v3n1.310>