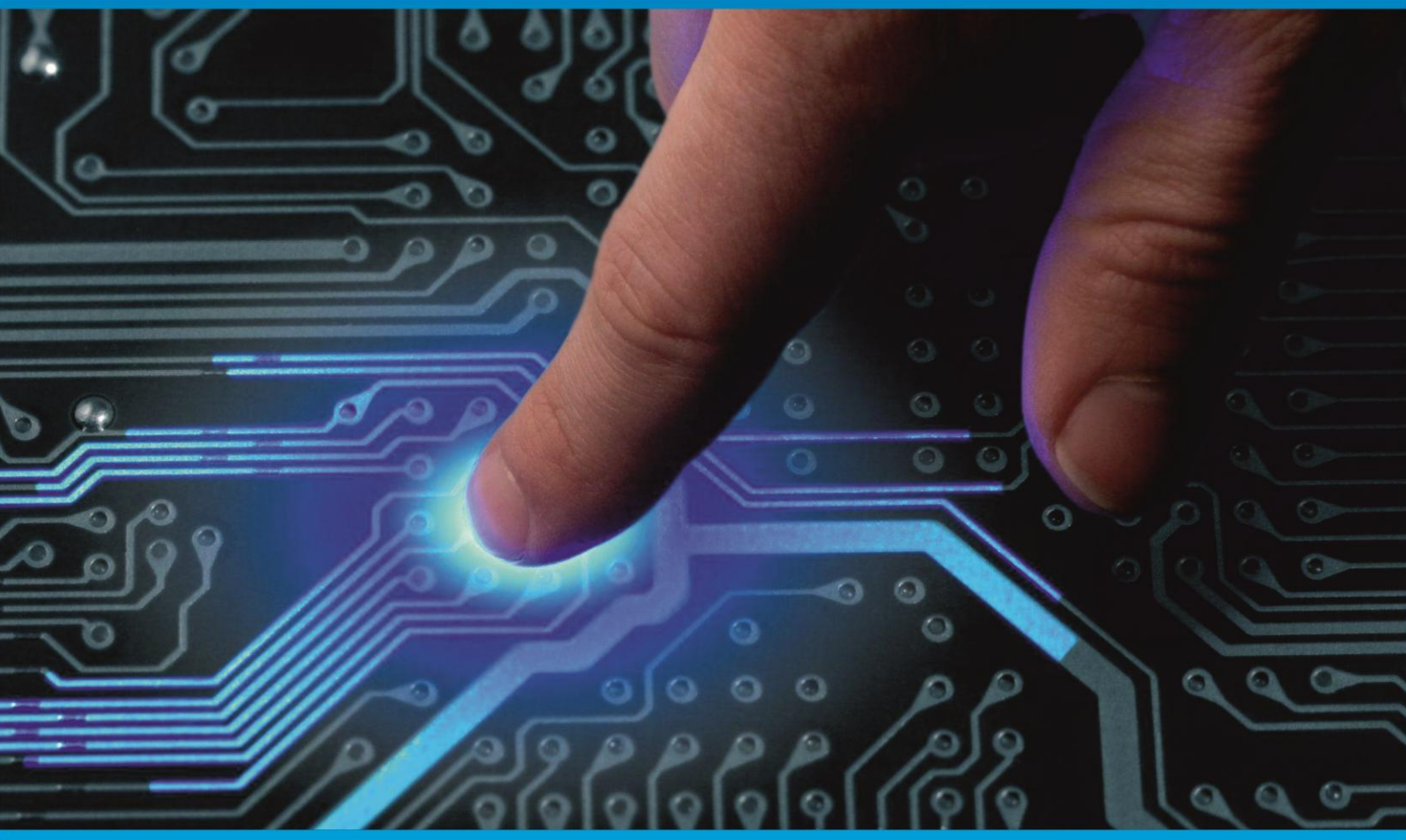
IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 9, Issue 5, May 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.488

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 5, May 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0905249 |

The Influence of Deep Learning Algorithms Factors in Software Fault Prediction

Mohd Asim Arzan[*1], Mohammed Mujeeb[*2], Syed Rayyan Ullah Hussaini[*3], Prof Sheena Mohammed[*4].

UG Scholar, Dept. of IT, ISL Engineering College, Bandlaguda, Hyderabad, Telangana, India[*1,2,3]

HOD, Dept. of IT, ISL Engineering College, Bandlaguda, Hyderabad, Telangana, India[*4]

ABSTRACT: The discovery of software faults at early stages plays an important role in improving software quality; reduce the costs, time, and effort that should be spent on software development. Machine learning (ML) have been widely used in the software faults prediction (SFP), ML algorithms provide varying results in terms of predicting software fault. Deep learning achieves remarkable performance in various areas such as computer vision, natural language processing, speech recognition, and other fields. In this study, two deep learning algorithms are studied, Multi-layer perceptron's (MLPs) and Convolutional Neural Network (CNN) to address the factors that might have an influence on the performance of both algorithms. The experiment results show how modifying parameters is directly affecting the resulting improvement, these parameters are manipulated until the optimal number for each of them is reached. Moreover, the experiments show that the effect of modifying parameters had an important role in prediction performance, which reached a high rate in comparison with the traditional ML algorithm. To validate our assumptions, the experiments are conducted on four common NASA datasets. The result shows how the addressed factors might increase or decrease the fault detection rate measurement. The improvement rate was as follows up to 43.5% for PC1, 8% for KC1, 18% for KC2 and 76.5% for CM1

I. INTRODUCTION

Developing high-quality software is one of the most challenges for software engineers. For that, software development should pass through a sequence of activities under certain constraints to come up with reliable and high quality software. A major drawback to having good quality and reliablesoftware is the occurrences of faults, where faults degraded the software quality and become unreliable end products also not acquire customer satisfaction. Reference [1] In order to achieve high-quality software, suitable planning, and control of software development cycle measures must be followed. The existence of faults is inevitable and it might occur in various phases of software development. One of the quality models that help to reduce software failure is Software fault prediction that also helps to avoid learning may provide valuable improvement in software fault prediction.

II. LITERATURE REVIEW

In this section, we review the most important relevant studiesthat focused on SFP and discussed the previous results for the stat-of-art, which used ML, NN, and Deep Learning. There are many studies improved software quality, and making better use of resources, and reducing or eliminating fault.Therefore, it is necessary to explore these researches to ensure understanding of the aspects of SFP.

A. MACHINE LEARNING

The following techniques use a clustering model to predict the fault for the unsupervised data. They suggest and evaluate new algorithms, such as K-Sorensen-means clustering, which is a new SFP clustering algorithm for K-means, using Sorensen distance to calculate cluster distance. JM1, PC1,
and CM1 are three datasets subject to the proposed approach.

B. NEURAL NETWORK APPROACH

Usually, neural networks consist of three components. The first is neurons. It is simple computing cells. Each neuron can receive input signals, process the signals and finally produce an output signal. Neural networks employ a massive interconnection between them to achieve good performance.shows the model of a neuron that has a set connecting links, each of which is characterized by weight, an adder for summing the input and an activation function.The second

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 5, May 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0905249 |

component is network architecture. The feedforward network is the most common type of neural network architecture that is composed of an input layer, a hidden layer, and an output layer. In a Feed-forward network the information moves forward in the neural network from leftto right, from the input nodes, through the hidden layers to the output nodes as shown in Figure 3. The third is a learning algorithm that is used during the learning processes to describe a process that adjusts the weights of the network to reduce the errors of the outputs.

C. DEEP LEARNING

propose Defect Prediction via Convolutional Neural Network (DP-CNN). CNN is used to automatically learn the semantic and structural features of programs.The approach contains four phases, starting with Abstract Syntax Trees (ASTs) which is used to extract tokens thatare encoded into numerical vectors. Then it employs CNN and combines it with traditional defect prediction features. Finally, it uses the Logistic Regression to decide if the
code files are having buggies or not. The experiments were made over seven open source projects and show that the DP-CNN improves the state-of-the-art method on average 12%. Dam et al. [19] develop a prediction model able to learn features automatically for representing source codethat has been used for defect prediction. They use a treestructured Long Short- Term Memory network (LSTM) that matches directly with the Abstract Syntax Tree representation of source code (ASTs). The model is built as a treestructured network of LSTM units to reflect better syntacticand many levels of semantics in source code

III. PROPOSED METHODOLOGY

In this section, we started to review the basic design steps,which required using MLPs and CNN for software faults prediction using the NASA datasets. The methodology steps,we started with selected four datasets with various fault percentage, then normalized for these datasets as preprocessing steps. The MLP is applied initially, and modify its parameters to measure the performance for it.Finally, we compared the results to find the best results achieved and applied the same steps for CNN. The MLP and CNN algorithms are implemented on python 3.6 languagebased on many libraries (such as Keras, Numpy, Panda and Sklearn, Matplotlib) to perform the experiments. Thereupon, each step is discussed in greater details in the subsequentsubsections. An overview of our proposed methodology and the pseudo code are presented in in Figure 5 and 6 respectively.

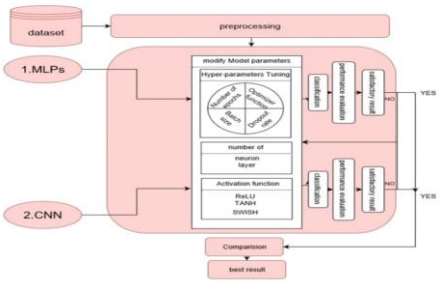


FIGURE 5. Research framework.

IV. DATA COLLECTION

A. SELECT DATASET

The datasets are selected from the NASA Metrics Data Program (MDP), include software measurement data and associated error data collected. NASA MDP dataset is made publicly available in order to encourage repeatable, verifiable, refutable, and/or improvable predictive models of software engineering. The datasets have been heavily used in software defect prediction experiments. The data sets' characteristics are presented in table 1. The attributes of the datasets are shown in Table 2.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 5, May 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0905249 |

TABLE 2. Attributes of datasets.

Attribute ID	Attribute	Definition
1	LOC	McCabe's line count of code
2	V(g)	McCabe "cyclomatic complexity"
3	EV(g)	McCabe "essential complexity"
4	IV(g)	McCabe "design complexity"
5		Halstead total operators + operands
6	V	Halstead "volume"
7	L	Halstead "program length"
8	D	Halstead "difficulty"
9	I	Halstead "intelligence"
10	E	Halstead "effort"
11		Halstead
12	T	Halstead's time estimator
13	lOCode	Halstead's line count
14	lOComment	Halstead's count of lines of comments
15	lOBlank	Halstead's count of blank lines
16	lOCodeAndCommente	Numeric
17	uniq_Op	unique operators
18	uniq_Opnd	unique operands
19	total_Op	total operators
20	total_Opnd	total operands
21	branchCount	of the flow graph
22	Problems	(no,yes)

B. NORMALIZATION

Normalization is used with numerical attributes that can find new ranges from an existing range based on an equation. It is performed during the preprocessing step, useful for classification algorithms as NN, and distance measurements as (KNN, clustering). This study applied the standardization method such that the attributes preserve the normal distribution. Standardization is a useful technique to transform attributes with a Gaussian Distribution and differing means and distribution. In Python, we use a Standard Scaler from scikit-learn library.

C. APPLYING DEEP LEARNING ALGORITHMS

In this phase, we applied two algorithms (MLPs and CNN) to address their abilities in enhancing the accuracy of SFP and determined the factor affecting the performance. In this step, we will select the various numbers of layers, different functions and different hyperparameters on each experiment.
Then we repeat the process until we get a satisfactory result.

D. MODIFY MODEL PARAMETERS

The settings which have to be defined for the network include hyper-parameter activation function, number of layers in each layer and hyper-parameter.

E. COMPARISON

A comparison will be conducted in order to determine the effectiveness of manipulating the studied parameters (hyperparameter, number of layers and neurons, activation , function).

F. DATA ANALYSIS & INTERPRETATION

This study performs a comparison of deep learning algorithms in terms of classification accuracy. In addition to use, Detection rate and TNR, that is computed by considering the positive and negative prediction of objects. The performance measures are computed through the following equations

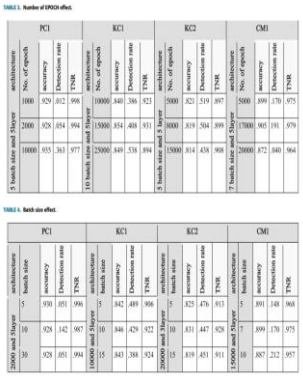
G. HARDWARE SPECIFICATION

While working on the implementation, two machines were used in accordance to the time needed for testing. The first one, a personal laptop, was used primarily for conducting small tests that take short time. The second machine, a virtual machine loaned from the Computer Centre, was used primarily for long testing purposes.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

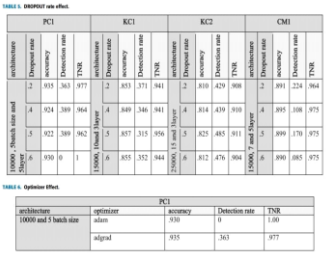
|| Volume 9, Issue 5, May 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0905249 |



V. EXPERIMENTAL RESULTS

The implementation of the MLPs and CNN is done in Python 3.6.5 using Keras Frameworks. In addition, Numpy, Panda and Sklearn libraries were used. Visualization is done using the Matplotlib library and Spyder served as the development environment. This work has developed without separation between the modification of the network and the implementation of the network. We had tried more than two hundred experiments with different (hyper-parameter, activation function and a number of layers). Experiments cover the modification of the model parameters used to improve the results. The following sub-sections present the results obtained



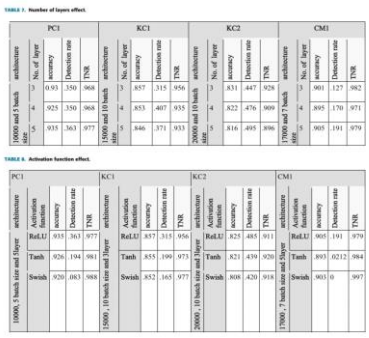
A. MLPS RESULT

To address the effect of (number of epoch, batch size, dropout rate, Optimizer, number of layers, activation function) the proposed approach achieved the following results by MLPs
algorithms as described in tables 3 - 8.
NUMBER OF EPOCH In order to examine the effect of a number of the epoch, we performed the following experiment in MLPs as described in table 3.
BATCH SIZE In order to examine the effect of batch size, we performed the following experiments as describedin table 4.
DROPOUT RATE In order to examine the effect of the dropout rate, we performed the following experiments as described in table 5.
OPTIMIZER In order to examine the effect of Optimizer, we performed the following experiments as described in table 6.
adgrad gave better accuracy and Detection rate values. NUMBER OF LAYERS In order to examine the effect of a number of layers, we performed the following experiments
as described in table 7.
ACTIVATION FUNCTION In order to examine the effect
of activation function, we performed the following experiments as described in table 8.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 5, May 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0905249 |

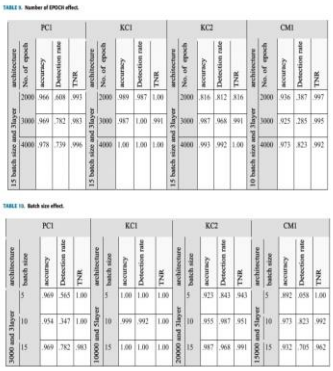


architecture	No. of layer	accuracy	Detection rate	TNR
PC1				
10000 and 5 batch size	3	0.93	.350	.968
	4	.925	.350	.968
	5	.935	.363	.977
KC1				
15000 and 10 batch size	3	.857	.315	.956
	4	.853	.407	.935
	5	.846	.371	.933
KC2				
20000 and 10 batch size	3	.831	.447	.928
	4	.822	.476	.909
	5	.816	.495	.896
CM1				
17000 and 7 batch size	3	.901	.127	.982
	4	.895	.170	.971
	5	.905	.191	.979

architecture	Activation function	accuracy	Detection rate	TNR
PC1				
10000, 5 batch size and 3layer	ReLU	.935	.363	.977
	Tanh	.926	.194	.981
	Swish	.920	.083	.988
KC1				
15000 , 10 batch size and 3layer	ReLU	.857	.315	.956
	Tanh	.855	.199	.973
	Swish	.852	.165	.977
KC2				
20000 , 10 batch size and 3layer	ReLU	.825	.485	.911
	Tanh	.821	.439	.920
	Swish	.808	.420	.918
CM1				
17000 , 7 batch size and 3layer	ReLU	.905	.191	.979
	Tanh	.893	.0212	.984
	Swish	.903	0	.997

B. CNN RESULT

To address the effect of (number of epoch, batch size, number of layers, activation function) the proposed approach achieved the following results by MLPs algorithms as
described in table 9, 10, 11. NUMBER OF EPOCH In order to examine the effect of the number of the epoch, we performed the following experiment on CNN as described in table 9.
BATCH SIZE In order to examine the effect of batch size, we performed the following experiment on CNN as described in table 10.
NUMBER OF LAYERS In order to examine the effect of a number of layers, we performed the following experiment on CNN as described in table 11.



architecture	No. of epoch	accuracy	Detection rate	TNR
PC1				
15 batch size and 3layer	2000	.966	.608	.993
	3000	.969	.782	.983
	4000	.978	.739	.996
KC1				
15 batch size and 3layer	2000	.989	.987	1.00
	3000	.987	1.00	.991
	4000	1.00	1.00	1.00
KC2				
15 batch size and 3layer	2000	.816	.812	.816
	3000	.987	.968	.991
	4000	.993	.992	1.00
CM1				
10 batch size and 3layer	2000	.936	.387	.997
	3000	.925	.285	.995
	4000	.973	.823	.992

architecture	batch size	accuracy	Detection rate	TNR
PC1				
3000 and 3layer	5	.969	.565	1.00
	10	.954	.347	1.00
	15	.969	.782	.983
KC1				
10000 and 3layer	5	1.00	1.00	1.00
	10	.999	.992	1.00
	15	1.00	1.00	1.00
KC2				
20000 and 3layer	5	.923	.843	.943
	10	.955	.987	.951
	15	.987	.968	.991
CM1				
15000 and 3layer	5	.892	.058	1.00
	10	.973	.823	.992
	15	.932	.705	.962

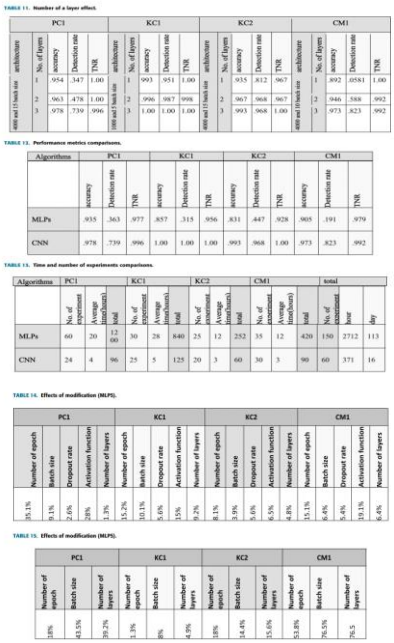
C. COMPARISON

COMPARISON IN TERMS OF PERFORMANCE Metrics Table 12 shows the best results we obtained from both algorithms. The results show a clear advantage for CNN. COMPARISON IN TERM OF TIME AND NUMBER OF EXPERIMENTS Table 13 shows the difference in effort in
terms of the number and time of experiments to obtain the satisfactory results of both algorithms.
As a summary of MLPs and CNN experiments, Tables 14 and 15 present improvements on results as parameters changed based on the detection rate.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 5, May 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0905249 |



VI. CONCLUSION AND FUTURE WORK

Machine learning is widely used in the area of prediction, one of the most promising subset is deep learning, the researchers prove that how deep learning achieves tangible performance in terms of prediction in various fields as computer vision, natural language processing, bioinformatics and software engineering etc. In this article, authors aimed to concentrate on answering two main research questions, Does the manipulating algorithms parameter could lead to introduce any performance enhancement in terms of accuracy?, Which of the studied deep learning algorithms provide the best SFP performance? The main essence of this study is to investigate the factors that have a tangible effect on the performance of the studied deep learning algorithms in the field of the SFP.

REFERENCES

[1] E. Erturk and E. A. Sezer, ‘‘Iterative software fault prediction with a hybridapproach,’’ Appl. Soft Comput., vol. 49, pp. 1020–1033, Dec. 2016.
[2] R. Kumar and D. Gupta, ‘‘Software Bug Prediction System Using Neural Network,’’ Eur. J. Adv. Eng. Technol., vol. 3, no. 7, pp. 78–84, 2016.
[3] I. B. Y. Goodfellow and A. Courville, Deep Learning, 1st ed. Cambridge, U.K.: MIT Press, 2016.
[4] S. Haykin, Networks and Learning Machines. London, U.K.: Pearson, 2009.
[5] Y.-S. Su and C.-Y. Huang, ‘‘Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models,’’ J. Syst. Softw., vol. 80, no. 4, pp. 606–615, Apr. 2007

INNO SPACE

SJIF Scientific Journal Impact Factor

Impact Factor:
7.488

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details