

Deep-Learning Approach with DeepXplore for Software Defect Severity Level Prediction

Lov Kumar[1(✉)], Triyasha Ghosh Dastidar[1], Lalitha Bhanu Murthy Neti[1], Shashank Mouli Satapathy[2], Sanjay Misra[3], Vipul Kocher[4], and Srinivas Padmanabhuni[4]

[1] BITS -Pilani Hyderabad, Hyderabad, India
{lovkumar,f20170829,bhanu}@hyderabad.bits-pilani.ac.in
[2] VIT Vellore, Vellor, India
[3] Covenant University, Ota, Nigeria
sanjay.misra@covenantuniversity.edu.ng
[4] Testaing.Com, Bengaluru, India
{vipulkocher,srinivas}@testAing.com

Abstract. Fixing the defects of earlier releases and working on fast and efficient fixing of those software defects is detrimental for the release of further versions. Bug tracking systems like Bugzilla get thousands of software defect reports every day. Manually handling those report to assign severity to the defects is not feasible. Earlier traditional Machine Learning methods have been used to predict the severity level from the defect description. This paper presents different deep learning models to predict defect severity level. Furthermore, the deep neural network was tested using a framework developed similar to that DeepXplore. Different word-embedding techniques, feature-selection techniques, sampling techniques and deep learning models are analyzed and compared for this study. In this paper, we have considered Descriptive statistics, Box-plot, and Significant tests to compare the developed models for defect severity level prediction. The three performance metrics used for testing the models are AUC, Accuracy and Neuron Coverage. This is a preliminary study on DNN testing on this dataset. Thus, the paper focuses on DeepXplore DNN testing technique. However further studies would be undertaken on comparative analysis of different DNN testing techniques on this dataset.

Keywords: Severity prediction · Imbalance handling · Feature selection · Deep learning · Word embedding

1 Introduction

Software defect resolution is one of the core activities of software maintenance. A huge number of software bugs are reported every day by different bug tracking systems used by various developers. In the present era, it is extremely difficult

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12955, pp. 398–410, 2021.
https://doi.org/10.1007/978-3-030-87007-2_28

to develop defect-free software considering its immense complexity, expending size and fluctuating customer requirements. Manual handling of defects is not a feasible task. Software defect severity level prediction is one of those areas of research, which has emerged very rapidly in the recent past. However, past studies have revealed that it is a time-consuming as well as expensive activity [1,2]. Also this process largely depends on the experience of the person handling the activity.

The Deep Learning (DL) approach helps the machines to process complex tasks with higher precision. With the growing use of DL systems, more uncertainty is prevailing towards the use of unaccounted test cases, which may lead to disastrous results. To solve this issue, DeepXplore was developed as a white-box testing framework for systematically testing real-world DL systems. In this paper, different DL techniques are being used to predict the software defect severity level. These deep neural networks were tested with DeepXplorer in order to check their neuron coverage using multiple DL systems with similar functionality for cross referencing. Three different performance metrics are also being used to test the DL models in order to assess their performance.

2 Related Work

When new software releases are created, it is necessary to fix errors in the older version with the inclusion of new features. Bug tracking systems are essential for any creation of software. The vulnerabilities and the definition of the bug are recorded on the bug monitoring systems. More bugs fixed help to create enhanced applications. Thousands of bugs are identified every day in certain well-known error monitoring software's like Bugzilla. This bug monitoring systems delete duplicate errors, assigns of bug severity and predicts bug severity level.

In the year 2006, **A. Sheta and D. Rine**, [3] in order to predict faulty modules, suggested using Linear Auto-regression models. To foresee possible errors, historical records on software defects have been used. The model was contrasted with the known Power Model (POWM) and the Root-Middle Square Error (RMSE) calculated efficiency, showing positive results.

In 2010, **Lamkanfi** [4] used text mining techniques to estimate the fault severity levels from the fault summaries. It was tested on three data sets, Mozilla, Eclipse and GNOME, open source. For both Mozilla and Eclipse it was between 0.65 and 0.75 based on the recall and precision values, while it was between 0.70 and 0.85 for GNOME. In 2011 [5], Lamkanfi continued the research to use the Naive Bayes, Naive Bayes Multinomial, KNN and SVM to compare different mined algorithms to estimate grade levels. Their success was compared using accuracy, recall and AUC. The operation of NB on both of these was also used in three selection techniques. For Naive Bayes Multinomial, the models' efficiency was better with a maximum range of 0.93 AUC. But regular NB also performed well when at least 250 bug reports were in the dataset package.

All the works in this field, to the best of our knowledge, have used only traditional machine learning methods for detecting software defect severity level. Through our work, deep learning methods have been used to detect fault severity levels.

3 Study Design

This section presents the details regarding various design settings used for this research.

3.1 Experimental Dataset

In this study, six different software datasets have been used, which are referred to as CDT, JDT, PDE, Platform, Bugzilla, and Thunderbird to validate our proposed models. These datasets have been collected from msr2013-bug_dataset-master [1]. Mining Software Repositories (MSR) conducted Challenge every year by provides software-related data and motivate participate to apply data mining techniques for finding important patterns. The datasets are the collection of defect reports wherein each defect report contains the defect ID, defect description, and severity level of the defects. Table 1 shows the details of the dataset used for this study. As shown in Table 1, the CDT software defect report consists of 2220 normal defects, 146 minor defects, 288 major defects, 42 trivial defects, 58 blocker defects, 106 critical defects.

Table 1. Experimental data set description

	Normal	% Normal	Minor	% Minor	Major	% Major	Trivial	% Trivial	Blocker	% Blocker	Critical	% Critical
CDT	2220	77.62	146	5.10	288	10.07	42	1.47	58	2.03	106	3.71
JDT	1906	66.71	261	9.14	430	15.05	104	3.64	50	1.75	106	3.71
PDE	2380	81.65	91	3.12	295	10.12	52	1.78	27	0.93	70	2.40
Platform	1485	52.07	215	7.54	715	25.07	145	5.08	77	2.70	215	7.54
Bugzilla	1342	46.68	598	20.80	352	12.24	302	10.50	167	5.81	114	3.97
Thunderbird	1100	37.72	387	13.27	655	22.46	91	3.12	25	0.86	658	22.57

3.2 Word Embedding:

The objective of this study is to develop defect severity level prediction models using defect reports. So, the performance of severity level prediction models is heavily dependent on the choice of features extracted from the defect reports. In this work, we utilized 7-word embedding techniques such as Continuous Bag of Words Model (CBOW), Skip-gram (SKG), Global Vectors for Word Representation (GLOVE), Google news word to vector (w2v), fasttext (FST), Bidirectional Encoder Representations from Transformers (BERT), and generative

[1] http://2013.msrconf.org/.

pre-training model (GPT) to convert the description text to an n-dimensional vector. These extracted vectors have been used an input to develop the models for predicting appropriate level of severity for defects. The performance of the models developed using these word-embedding are also compared with most frequently used technique i.e., term frequency-inverse document frequency (TF-IDF) to extract feature from text.

3.3 Feature Selection Techniques

The defect severity level prediction models are developed by taking extracted vector using word-embedding techniques for defect report as an input. So, the predictive ability of the models also depends upon the selection of important features vectors. In this paper, three different features selection techniques have been used to remove irrelevant sets of features and select sets of un-correlated sets of relevant features. Initially, the rank-sum test is applied on individual feature to find significant sets of features that have the ability to predict the level of severity for defects [6]. Next, we have used the cross-correlation analysis to compute uncorrelated sets of features. Finally, principal component analysis (PCA) [7] has been used to find new sets of values for features that have uncorrelated with others.

3.4 Deep Learning Architecture

With the increase in the amount of data, traditional machine learning models seem to fail after a certain amount of time. Deep learning means the machine can grasp complex perception tasks with great precision. Deep learning is also known as deep structured learning with multiple layers composed of non-linear processing units for conversion and extraction of features. The findings from the previous layer are used as the input in any subsequent layer. The learning process is performed by means of distinctive stages of abstraction and various layers of representation, whether supervised or unsupervised. The basic computing device is a deep learning system or a deep neural network is neuron which takes several data as an input. It combines these signals linearly and transforms the combined signals into the non-linear tasks in order to generate outputs. In this study, three different architecture of deep learning models have been used to train the models for predicting defect levels of severity.

3.5 SMOTE

The information present in experimental data sets description of Table 1 suggests that the considered data does not have an equal number of the severity level of the defects; as for example, only 1.46 % of trivial defects present in the case of CDT. So, the models trained using these data without applying any sampling techniques may lead to an adverse effect on the actual performance. In this work, the Synthetic Minority Oversampling Technique (SMOTE) has been used to balance the data. SMOTE technique is identified as a very popular technique by different researches that helps to improve the predictive ability of the models.

3.6 DeepXplore

Deep Learning has come a long way in recent years. It is being used in all domains. There has been a growing use of DL systems in security-driven systems like self-driving cars, malware detection. In all those cases, there is a case importance of making sure we take into account all the test cases. Any sort of test case that has not been accounted for can lead to disastrous results in DL systems where security is of utmost importance. Earlier mostly manually labeled data was used for testing but that often led to the negligence of effects of rare test cases.

To solve this issue, DeepXplore was developed in 2019 as a white-box testing framework for systematically testing real-world DL systems. The main features taken into consideration are neuron coverage, using multiple DL systems with similar functionality for cross-referencing. These techniques reduce manual checking. This white-box testing method was tested on five very famous real-world datasets like Udacity, ImageNet, MNIST, Drebin and PDF.

To understand DL testing, it is necessary to understand DNN architecture first. DNN architecture is inspired by the human brain. It is used for extracting high-level features from raw inputs without any human intervention. The neuron is the smallest computing unit in the DNN. Each neuron has an activation function working on the raw input coming into it, after which the result is passed on to the neighbouring connected neurons. A typical DNN is made up of multiple layers where each layer consists of multiple neurons. There are three types of layers in the network: input, output and one or more hidden layers. Each neuron is directed towards the neurons in the next layer. Each connected neuron has an attached weight parameter which elaborates the strength with which they are connected. The weights of the network are learned during training to minimise the cost function using gradient descent. In each layer of this architecture, the DNN learns high-level representations of the raw-input which gradually gives us the highly meaningful outputs.

The first aim of the algorithm is to generate test inputs that can induce different behaviours in the tested DNNs, i.e. different DNNs will classify the same input into different classes. Next comes the objective to generate inputs which will increase the neuron coverage significantly. This is achieved through the selection of inactivated neurons in the process and modification of the input so that the neuron output exceeds the neuron activation threshold.

4 Results and Analysis

This section presents the results and analysis of the prediction models developed using word-embedding techniques, feature-selection techniques, sampling techniques, and deep learning techniques. In this work, seven word embedding techniques and Tf-IDF is applied on defect report of 6 data sets along 3 deep learning models with 5-fold cross-validations, 4 sets of features. Thus, in this research, we assessed the performance of 2880 (8*6*3*5*4) defect severity level

prediction models with the help of Box-plot, and Significant tests. In this study, we have also considered Neuron Coverage to evaluate the feasibility of the developed models.

4.1 Word Embedding

This work makes the use of different word-embedding techniques to convert the text into a vector of numbers which could be fed in as input of the prediction models. In this work, we have used Accuracy, AUC and Neuron Coverage as performance parameters to compare the ability to predict the level of severity using models developed by considering word-embedding extracted vectors as an input.

Box-Plot: Word-Embedding Techniques. We have used seven different word embedding techniques such as Continuous Bag of Words Model(CBOW), Skip-gram(SKG), Global Vectors for Word Representation (GLOVE), Google news word to vector(w2v), fastText (FST), BERT, and generative pre-training model (GPT) to convert the defect descriptions into numerical vector representations. The performance of the models developed using above techniques are also compared using term frequency-inverse document frequency(TFIDF). Figures 1 and 2 show the box-plot diagram for Accuracy, AUC and Neuron Coverage of the models developed using different word-embedding techniques. While checking the box-plots for different word embedding in terms of AUC, we see that both w2v and GLOVE has average AUC value of 0.78, maximum value of 0.99 AUC for GLOVE and 0.98 AUC for w2v and the Q3 values for GLOVE and w2v are also comparable at 0.91 and 0.92 AUC respectively. So it can be summarized that GLOVE and w2v both are the contenders for the best fit for word embedding for predicting defect severity levels in terms of performance. Figure 2 compares the models using neuron coverage. The value of neuron coverage helps to predict the amount of logic explored by a set of test inputs. The high value of neuron coverage represents that the neurons of deep learning models are successfully tested using test inputs. It can be seen that the median value of neuron coverage for the models trained on TF-IDF relatively higher and also closely by the GLOVE word embedding.

Comparison Using Significant Test. We have also used significant test with the help of rank-sum test to find out whether the models that are developed using 7 different word embedding provide a significant improvement or not. The test is used to validate the considered null hypothesis like "The models developed on extracted vectors using different word-embedding have not significant improvement in ability to predict level of severity". The above null-hypothesis is accepted if the p-value is more than 0.05. The results obtained after applying rank-sum test are listed in Table 2. While checking the results of the significant tests in terms of AUC for different word embedding, we observed the above considered hypothesis is rejected for most of the pairs i.e., the performance of the

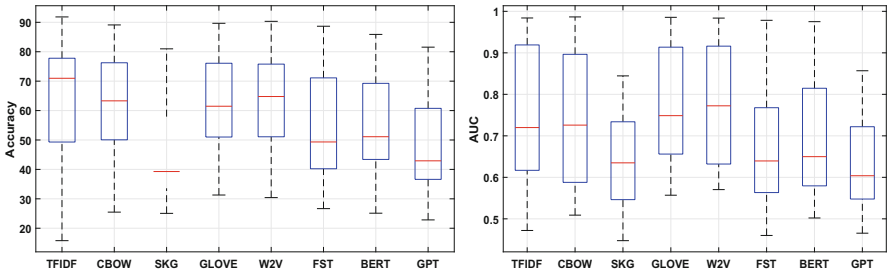

Fig. 1. Performance Box-Plot diagram: performance of different word embedding

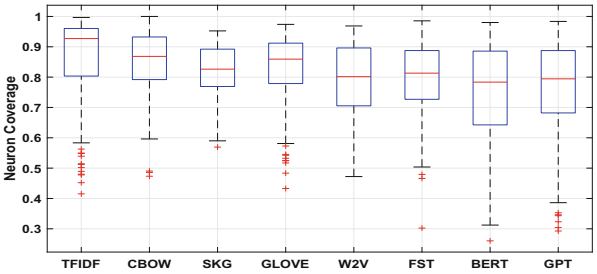

Fig. 2. Neuron coverage : different word embedding

models depended on the word-embedding techniques that used to extract features from defect reports. Similar, significant improvement in Neuron Coverage for models trained on different word-embedding techniques.

Table 2. Significant tests: different word embedding

	AUC								Neuron coverage							
	TFIDF	CBOW	SKG	GLOVE	W2V	FST	BERT	GPT	TFIDF	CBOW	SKG	GLOVE	W2V	FST	BERT	GPT
TFIDF	1.00	0.03	0.00	0.16	0.31	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CBOW	0.03	1.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	1.00	0.00	0.10	0.00	0.00	0.00	0.00
SKG	0.00	0.00	1.00	0.00	0.00	0.04	0.00	0.49	0.00	0.00	1.00	0.04	0.14	0.27	0.00	0.06
GLOVE	0.16	0.00	0.00	1.00	0.73	0.00	0.00	0.00	0.00	0.10	0.04	1.00	0.01	0.01	0.00	0.00
W2V	0.31	0.00	0.00	0.73	1.00	0.00	0.00	0.00	0.00	0.00	0.14	0.01	1.00	0.65	0.12	0.60
FST	0.00	0.00	0.04	0.00	0.00	1.00	0.12	0.01	0.00	0.00	0.27	0.01	0.65	1.00	0.05	0.33
BERT	0.00	0.01	0.00	0.00	0.00	0.12	1.00	0.00	0.00	0.00	0.00	0.00	0.12	0.05	1.00	0.34
GPT	0.00	0.00	0.49	0.00	0.00	0.01	0.00	1.00	0.00	0.00	0.06	0.00	0.60	0.33	0.34	1.00

4.2 Sampling

The considered dataset has highly imbalanced that may give very poor performance in predicting the minority class. Thus the performance of the models can be improved by oversampling the minority class. This work makes the use of SMOTE technique to handle class imbalance. The work also make the box-plot and significant test analysis to compare the performance of the models trained on SMOTE data with original data.

Box-Plot: SMOTE. Figure 3 compares the performance of the trained models on SMOTE and original data with respect to Accuracy and AUC. The accuracy value data in Fig. 3 shows that the models trained on SMOTE data have similar ability to assign level of severity based on defect report with models train on original data. However, AUC is considered to be a better metric in case of class imbalance problem. So, based on AUC values, we observed that models trained on SMOTE data have better ability to assign level of severity based on defect report. Figure 4 compares the models using neuron coverage. It can be seen that the median value of neuron coverage for the models trained on SMOTE data relatively higher with models trained on original data.

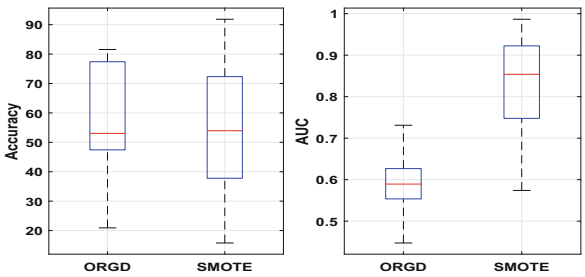

Fig. 3. Performance Box-Plot diagram: performance of original data and SMOTE

Comparison Using Significant Test. The main idea behind this test is to find out whether the models that are developed using SMOTE and original data provide a significant improvement or not. The result present in Table 3 confirmed that the models trained on SMOTE data have significant improvement in both AUC and Neuron Coverage.

4.3 Feature Selection

Feature Selection is the technique for selecting related features from the provided set of data and thus removing the other unrelated, irrelevant features that does not affect our target variable. This technique helps in improving the performance of the model. Irrelevant and lowly related features sometimes hinder the performance of the model. In this study, different varieties of feature selection techniques such as significant sets of features using the rank-sum test (SIGF), uncorrelated sets of features using cross-correlation analysis(UCRF), and principal component analysis(PCA) have been applied on datasets to find right combination of relevant features. The performance of these techniques is derived in terms of AUC and accuracy, and compared with box-plot and significant test.

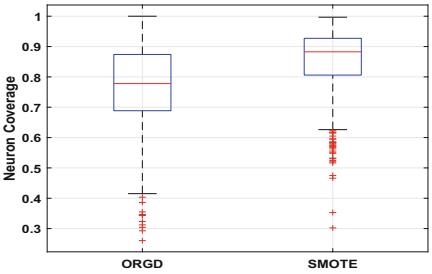

Fig. 4. Neuron coverage : original data and SMOTE

Table 3. Significant tests: original and SMOTE data

AUC		
	ORGD	SMOTE
ORGD	1.00	0.00
SMOTE	0.00	1.00
Neuron coverage		
	ORGD	SMOTE
ORGD	1.00	0.00
SMOTE	0.00	1.00

Box-Plot: Feature Selection Techniques. Figure 5 compares the performance of the different feature selection techniques using box-plot with respect to Accuracy and AUC. The box-plot of the different feature selection techniques in terms of accuracy shows the highest average value for the model using all the features but the interquartile ranges for ALLF, SIGF and UCRF are almost similar. However, AUC is considered to be a better metric in case of class imbalance problem. So, based on AUC values, we observed that the performance of models remain same after removing features using rank-sum test and cross-correlation analysis(UCRF) i.e., it is possible to develop good models with less numbers of features. Figure 6 compares the models using neuron coverage. It can be seen that the median value of neuron coverage for the models by taking all feature or significant features is almost similar.

Comparison Using Significant Test. The main idea behind this test is to find out whether the models that are developed using 3 different feature selection techniques provide a significant improvement or not. The result present in Table 4 confirmed that the models trained on all features, SIGF, and UCRF are significantly similar performance i.e., there exists small number of features that have better ability to predict level of severity. However, the models trained on different sets of features are significantly different in terms of Neuron Coverage.

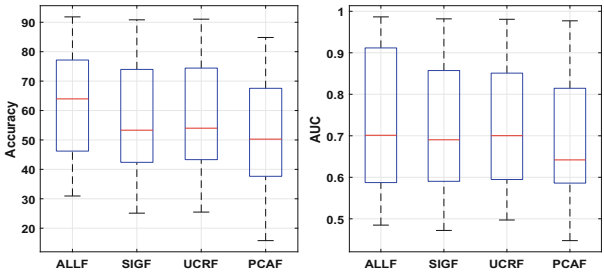

Fig. 5. Performance Box-Plot diagram: performance of different sets of features

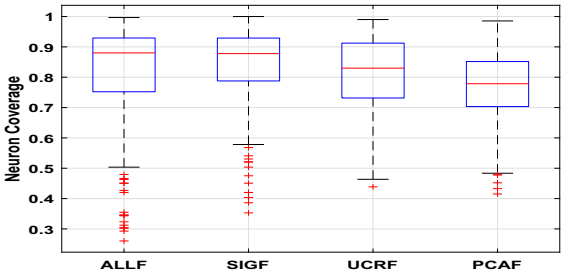

Fig. 6. Neuron coverage : different sets of features

4.4 Deep Learning Models

This study makes the use of three different variants of deep learning models by changing the numbers of hidden layers to train the models for predicting level of severity based on defect report. The activation function used for the hidden layers were ReLU (Rectified Linear Activation Function) and for the output layer softmax activation function was used. The optimization algorithm used for all the models were Adam which is used to update the network weights during training. Finally, the models ability to predict severity level trained using above deep learning techniques are compared with the help of AUC and accuracy and compared in terms of box-plot and significant test.

Box-Plot: Deep Learning Models. Fig. 7 compares the performance of the different models using box-plot with respect to Accuracy and AUC. The box-plot of the different deep learning models (DLM1, DLM2 and DLM3) in terms of accuracy shows the average value for all the models to be same and the interquartile ranges for DLM1, DLM2 and DLM3 are almost similar. However, AUC is considered to be a better metric in case of class imbalance problem. So, based on AUC values, we observed that the average AUC values for DLM2 and DLM3 are similar and is lesser than DLM1 by a very minuscule margin. Figure 8 compares the models using neuron coverage. It can be seen that the median value of neuron coverage for the all three models is almost similar.

Table 4. Significant tests: different sets of features

AUC				
	ALLF	SIGF	UCRF	PCAF
ALLF	1.00	0.16	0.23	0.00
SIGF	0.16	1.00	0.81	0.01
UCRF	0.23	0.81	1.00	0.01
PCAF	0.00	0.01	0.01	1.00
Neuron coverage				
	ALLF	SIGF	UCRF	PCAF
ALLF	1.00	0.00	0.00	0.00
SIGF	0.00	1.00	0.00	0.00
UCRF	0.00	0.00	1.00	0.10
PCAF	0.00	0.00	0.10	1.00

Comparison Using Significant Test. The main idea behind this test is to find out whether the models that are developed using 3 different deep learning techniques provide a significant improvement or not. The result present in Table 5 confirmed that the models trained using different techniques are significantly similar performance as well as similar Neuron Coverage.

Table 5. Significant tests: classification techniques

AUC			
	DLM1	DLM2	DLM3
DLM1	1.00	0.94	0.67
DLM2	0.94	1.00	0.74
DLM3	0.67	0.74	1.00
Neuron coverage			
	DLM1	DLM2	DLM3
DLM1	1.00	0.02	0.35
DLM2	0.02	1.00	0.11
DLM3	0.35	0.11	1.00

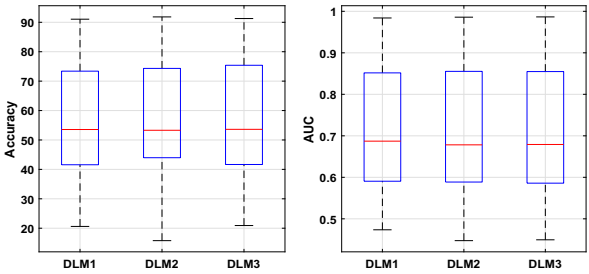

Fig. 7. Performance Box-Plot diagram: performance of different classification techniques

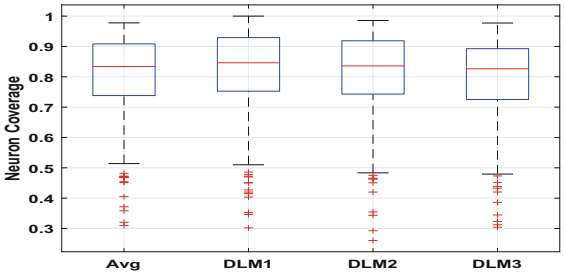

Fig. 8. Neuron coverage : different deep learning models

5 Conclusion

This research proposed models using word embedding, data sampling technique, deep learning, and feature selection to predict level of defect severity using defect reports. The proposed models have been validated with the help of six open source datasets and compared using AUC and Accuracy. This work also make use of DeepXplore to test the developed models. The quality of the models developed using the above techniques are compared with the help of neuron coverage. The value of neuron coverage helps to predict the amount of logic explored by a set of test inputs. The high value of neuron coverage represents that the neurons of deep learning models are successfully tested using test inputs. The experimental results suggested that the models trained on extracted vectors using word-embedding techniques have better ability to predict defect severity level. Similarly, the models trained on sampled data with relevant sets of features using deep learning technique have ability to assign right level of severity to the defects based on defect reports. More importantly, the high value of neuron coverage (Average 89%) confirms the better quality of defect severity level prediction models.

Acknowledgment. This research is funded by TestAIng Solutions Pvt. Ltd.

References

1. Singh, V.B., Misra, S., Sharma, M.: Bug severity assessment in cross project context and identifying training candidates. J. Inf. Knowl. Manage. **16**(01), 1750005 (2017)
2. Gupta, A., Suri, B., Misra, S.: A systematic literature review: code bad smells in java source code. In: Gervasi, O., et al. (eds.) ICCSA 2017. LNCS, vol. 10408, pp. 665–682. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62404-4_49
3. Sheta, A., Rine, D.: Modeling incremental faults of software testing process using AR models (01 2006)
4. Lamkanfi, A., Demeyer, S., Giger, E., Goethals, B.: Predicting the severity of a reported bug. In: 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), pp. 1–10, Los Alamitos, CA, USA, IEEE Computer Society (May 2010)
5. Lamkanfi, A., Demeyer, S., Soetens, Q.D., Verdonck, T.: Comparing mining algorithms for predicting the severity of a reported bug. In: 2011 15th European Conference on Software Maintenance and Reengineering, pp. 249–258 (2011)
6. Kumar, L., Misra, S., Rath, S.K.: An empirical analysis of the effectiveness of software metrics and fault prediction model for identifying faulty classes. Comput. Stan. Interfaces **53**, 1–32 (2017)
7. Kumar, L., Sripada, S.K., Sureka, A., Rath, S.K.: Effective fault prediction model developed using least square support vector machine (lssvm). J. Syst. Softw. **137**, 686–712 (2018)