A Systematic Study for Learning-Based Software Defect Prediction

Han Cao[1]

[1] Faculty of Applied Sciences, Simon Fraser University, Burnaby, BC, V5A 1S6, Canada

han_cao@sfu.ca

Abstract. Software defect refers to the code error in the process of software development, which could cause execution fault under specific conditions, resulting in failure, collapse, and high cost of the target software. Traditional detection techniques for software defect contain static and dynamic analysis, both of which require a great deal of workforce and time. With the development of machine learning and deep learning, software defect prediction has opened a new avenue to circumvent the drawbacks of traditional analysis approaches. Although various learning-based techniques in the prediction field have been developed, there is a lack of systematic summary and classification from the technical point of view. This paper studies the problem from the three aspects: traditional machine learning, deep learning, and hybrid learning. Moreover, the predicted performance is discussed in detail, especially in cross-project and just-in-time, to understand current research status thoroughly. This paper also provides a useful guide for further research, particularly for the potential usage of deep learning in semantic defect prediction.

1. Introduction

Software defect refers to code errors (commonly known as bugs) in the process of software development. In most cases, such defects will cause software faults under certain conditions and lead to application failure. In the age of information, software defects are able to lead to information security problems. Such type of software defects, called software vulnerability, is prone to raising severe resource loss and poor user experience. The earliest known example of system defect was the Mariner 1 launch failure in 1962 when NASA had to destroy an $80 million spacecraft because there was a hyphen missed in the FORTRAN program [1]. The "Morris Worm" of cybersecurity forced people to focus on software vulnerability more than three decades ago and provided an impetus for the US to create the CERT (National Internet Emergency Response Center) [2].

Software defect analysis is of great importance in checking software defects. The most direct analysis way is static analysis, which can find defects by using various techniques. A large number of tools such as Coverity Scan [3] and Fortify SCA (Static Code Analyzer) [4] have been developed to perform such static analysis. Dynamic analysis is another defect analysis approach which tracks the execution of programs as well as analyzes the memory information and function calls. In this way, it can find exceptions and use techniques such as ambiguity test and stain analysis to identify defects. Many tools are relying on dynamic analysis, such as Valgrind [5] and Ftrace [6]. Although traditional software defect analyses become sophisticated, it is evident that static analysis is labor-intensive and performs poorly in semantic analysis. The resources and time expended are usually unacceptable for dynamic analysis, since it has to test every line of code. Thus, dynamic analysis is seldom used in practice.

With the development of machine learning and statistical theory, it is more likely to introduce such techniques to the timely detection of defects in the software development phase. As a result, software defect prediction with learning-based methods gradually becomes an emerging research field. In the early years, some researchers exploited statistical and probabilistic approaches to predict the defect time distribution of the software's life cycle [7], which belongs to the research field of reliability engineering. After the year 2000, many researchers began to use machine learning to predict defects in programs [8].

In recent years, with the development of artificial neural network (ANN) and natural language processing (NLP), deep learning has become popular in many researchers [9-12]. Compared with traditional machine learning, which is based on strict metrics, deep learning tends to convert source code into vectors to learn directly. Vectors can preserve code information thoroughly; they even include the annotation information [13]. Deep learning has excellent performance in both syntax learning and semantic learning.

At present, although lots of researches are on software defect prediction, there lacks a systematic summary from the technical perspective. Therefore, this paper aims to address this technical gap by investigating as well as classifying the related work of software defect prediction. Also, this paper summarizes the advantages and disadvantages of the related techniques which provide support for the future work of software defect prediction.

2. Techniques

Software defect prediction refers to the use of machine learning in the early stage of software development. It can save development costs and improve development speed. The core of its performance is machine learning algorithms. Technically, learning-based software defect prediction includes three main branches: traditional machine learning, deep learning, and transfer learning.

2.1. Traditional Machine Learning

Traditional machine learning has been explored for a long time, and its algorithms are relatively straightforward. Here are three typical traditional machine learning techniques that are widely used.

2.1.1. Random Forest. The random forest is initially proposed by Breiman L et al. [14], which works as a classifier with multiple decision trees that collaborate as a forest for decision. The multiple decision trees vote their result, and RF employs a bootstrap sample to ensure the independence of multiple decision trees as well as increases the correlation between decision trees. It is capable of handling multi-variable inputs and large datasets, with relatively high accuracy, especially in handling unbalanced datasets. The forest decision mechanism can estimate the generalization error without bias.

2.1.2. Bayesian Network. The Bayesian network, also known as the belief network or directed acyclic graphical model is a probabilistic graphical model. Every node in the graph represents a random variable, while every edge represents a conditional dependency. Also, the Bayesian network is efficient at dealing with uncertain problems. Bayesian network expresses the correlation between information elements using conditional probability and is able to learn and reason with limited, incomplete, or uncertain information.

2.1.3. Support Vector Machine. Support Vector machine is first proposed as a generalized linear classifier for binary classification whose decision boundary is the maximum-margin hyperplane of the input learning samples. SVM was introduced as early as 1964. However, it did not until the 1990s [15]. Besides performing linear classification, SVMs can also perform a non-linear classification with the kernel trick which maps their inputs into high-dimensional feature spaces implicitly.

2.2. Deep Learning

Deep learning originates from the ANN. As early as 1943, psychologist W. S. Mcculloch and mathematical logician W. Pitts built its neural networks and mathematical models, and by the 1980s, ANN had become a hot topic in artificial intelligence [16]. The concept of deep learning was not put forward until 2006 by Hinton G et al. [17]. Deep Learning transforms initial "low-level" feature representation into "high-level" through multi-layer processing; it can complete complex classification tasks with simple models. Also, it has the capabilities of feature learning and representational learning. Therefore, it can achieve better performance while being flexible. All these attributes contribute to the rapid development of machine learning.

2.2.1. Deep Belief Network. DBN is the earliest deep learning method, proposed by Hinton in 2006 [17]. DBN is a generative graphical model. Compared with traditional neural network model, it can establish a joint distribution between the observation data and labels which evaluates both P(observation | label) and P'(label | observation) while the discriminant model only evaluated the P(label | observation).

DBN consists of multiple restricted Boltzmann machines (RBM) layers which contain a visible layer and multiple hidden layers. There are connections between the layers, but the cells within a layer are not interconnected. Elements inside the hidden layer are trained to capture the correlation of high-order data in the visual layer, which is also known as feature extraction. The training process is carried out layer by layer according to the RBM.

2.2.2. Convolutional Neural network. CNN is a class of convolutional feedforward neural network with a depth structure and is capable of shift-invariant classification. It is constructed to mimic the visual perception of biological processes and can be used for both supervised learning and unsupervised learning. The convolution kernel parameter sharing in the hidden layer and the sparsity of the connection between layers make it possible for CNN to handle high-dimensional data properly. Also, it is unnecessary to select the features manually to make the classification effective. Essentially, it is a mapping between inputs and outputs. With an adequate amount of known patterns about the mapping between inputs and outputs, there is no need for any exact mathematical expressions.

2.2.3. RNN/LSTM/GRU. Recurrent Neural Network (RNN) is a kind of artificial neural network where nodes and their connections form a directed graph along a temporal sequence [11]. RNN was proposed in the 1980s and developed rapidly in the past ten years. RNN can deal with time series not only spatially, but also temporally, i.e., memorization. It is Turing complete and has parameter sharing. As a result, RNN can efficiently acquire the nonlinear features that are in order.

With the increasing application of RNN, it is found that there is a long-term dependencies problem, that is, there are gradient vanishing and gradient explosion phenomena when learning sequence. This problem means RNN cannot guarantee the long-term nonlinear relationship. For this reason, a large number of optimization theories were introduced, and many improved algorithms are derived. To name a few, long short-term memory networks (LSTM) [11], gated recurrent unit networks (GRU) [12], echo state networks (ECHO state networks) and independent recurrent neural networks (Independent RNN) [18].

2.3. Transfer Learning

In traditional classification learning, in order to guarantee accuracy and reliability, two fundamental assumptions are followed. The first one is that the training samples and the test samples are independent and identical distribution. Secondly, there should be enough training samples to learn and get a satisfactory classification model. However, these two conditions cannot be well met in practical. Besides, building a new model is complicated and time-consuming. Transfer learning solves these problems by making use of the existing learning achievements to speed up the learning progress.

Transfer learning usually can be divided into feature-based learning and instance-based learning [8]. Feature-based transfer learning is mainly used to identify the common features between the source domain and the target domain, and then transfer knowledge using these features. Instance-based transfer learning is also commonly used to solve the domain adaptation problem by selecting the training samples which are beneficial to the classification of the target domain.

3. Software Defect Prediction

Software defect prediction is a critical part of software quality assurance. With the rapid increase of software size and quantity, it becomes increasingly difficult to check the software defects by manual checking or basic tools. Especially when it comes to cross-project or cross-version situations, there is very few knowledge to be transferred. Most current editors give proper just-in-time notifications for some syntax errors, but not for complex syntax errors, and not to mention semantic errors.

In order to perform the software defect prediction automatically, machine learning is introduced and has an excellent effect. The traditional machine learning mainly applies the probability statistics method, using extracted features and software measure metrics to learn classification and predict defect. With the development of deep learning and NLP, it brings the evolution of software defect prediction. There is a tendency for a source code based software defect prediction research, and the method is gradually applied in practice. In recent years, Hybrid Learning has been applied in many defect prediction scenarios. It integrates a variety of learning models and algorithms to improve the prediction accuracy.

3.1. Machine Learning

Luca et al.[19] test different classifiers, i.e., binary logistic regression (BLR), J48, ADTree, multilayer perceptron (MLP), naive Bayes (NB), and random forest (RF). They use the RF algorithm, which has a higher performance to 10 data sets and conduct the training by selecting more predictive variables. The result of short-term defect prediction was proved to be effective.

Yeongjun C et al.[18] use three machine learning classification methods: RF, logistic regression (LR) and NB on five different programming languages and projects from various application areas. The datasets not only cover sub-projects but also cross-sub-projects in the machine learning training. All three learning methods have achieved satisfying results.

Osman H et al. [20] apply two commonly used machine learning algorithms: k-nearest neighbors (KNN) and SVM, which improve the prediction accuracy by hyperparameter optimization. The experiment result of 5 open-source Java programs shows that the defect prediction accuracy has increased by 20% and 10%, respectively, for the two learning algorithms.

Sokratis t et al. [21] use four conventional machine learning methods: C4.5 decision trees, NB, Bayesian networks (BN) and LR to predict the performance defects of commercial real-time systems (RTS) applications. They also perform necessary preprocessing such as cleansing and re-balancing for the datasets before learning. The result of their experiment is promising, especially for the C4.5 decision trees and BN.

Puja A H et al. [22] use NB algorithm to compute the Chidamber and Kemerer object-oriented metric (CK OO) as the feature parameters. They select the petstore web application and other project datasets to run cross-project defect prediction (CPDP) experiments. The results indicate relatively high accuracy of 72.30% - 89.30% and slightly low false alarm of 5% - 26.67%, which can predict more defect modules than traditional code review. However, it has low precision and recall score, around 12.5% - 25% and 20%-60%.

Yanhong Yang et al. [23] use the cluster ensemble method to run CPDP and use unsupervised learning to deal with unlabeled datasets directly, thus avoiding different data distribution problems. The experiment is carried out on 15 open source projects. Compared with the three most used supervised learning methods: LR, decision tree (DT), and RF, it produces better prediction results.

Kishan K G et al. [24] use the well-received NASA defect datasets in defect prediction research and preprocess the datasets with repeated attribute removal and noise removal using machine learning algorithms such as NB, LR, DT, RF, and MLP. The learning performance is improved by cleaning the data, and the results show that these learning algorithms have agreeable performance.

Lipika G et al. [25] focus on the severe imbalance problem of data sources from different projects, which have a significant impact on the performance of CPDP. They use the synthetic minority over-c technique (SMOTE) to preprocess the datasets, and run experiment on the open PROMISE repository using gradient boosting, compared with another kind of ensemble learning algorithm like RF, and achieve better prediction performance.

3.2. Deep Learning

Wang et al. [9] propose to leverage the DBN based representation-learning algorithm to learn the semantic representation of programs automatically from source code. To this end, they extracted the abstract syntax trees (ASTs) from a targeted app and generated corresponding token vectors. The experiments on ten open source projects demonstrate that the proposed approach significantly improves the performance of both within-project defect prediction (WPDP) and CPDP compared to traditional features.

Thong et al. [10] introduce two software projects (QT and OPENSTACK), which were first collected by Shane and Yasutaka and used for relevant research. They propose a CNN based method and train the model on the two software projects. In this process, the researchers keep optimizing the CNN parameters in three ways (cross-validation, short-period, and long-period). Compared with the state-of-the-art approach, the result achieves improvements of 10.36-11.02% for the QT and 9.51-13.69% for the OPENSTACK in terms of the area under the curve (AUC).

Hoa K D et al. [11] use a famous timing-related learning algorithm LSTM, a variant of the RNN, to make a thorough study on 18 Android applications. These applications involve in fields of the economy, education, books, and network, which have comprehensive representations. Using the time series characteristic of the LSTM processing not only has an outstanding effect on the syntax study training but also has an excellent effect on the semantic study training. As a result, the WPDP was improved by 3%-58%, and CPDP was improved by 85%.

Zhen L et al. [12] use another important algorithm of RNN: Bidirectional LSTM (BiLSTM), on 19 popular C/C++ open-source programs to predict several critical security vulnerabilities. These programs include the Linux kernel, Firefox, Thunderbird, Apache Http Server, and the vulnerabilities include two typical types: Buffer Error (i.e., CWE-119) and Resource Management Error (i.e., CWE-399). The BiLSTM has been trained with optimized parameters and achieved impressive results.

Anh V P et al. [26] use multi-layer directed graph-based CNN to predict the defects on four real-world datasets written by C, C++, Java, and Python. The method improves the defects prediction accuracy from 4.08% to 15.49% in comparison with the feature-based approaches and increased the accuracy from 1.2% to 12.39% in comparison with the tree-based approaches.

Xuan H et al. [13] make full use of the annotation information, as well as comments augmented from the annotation programs, to enhance the learning effects. They also use CNN to process open-source programs from PROMISE, which improves defect prediction performance.

Anh V P et al. [27] choose four open-source datasets and convert them into assembly code after necessary preprocessing. Then they use CNN to learn the assembly code. When compared with the traditional machine learning algorithms such as SVM, there is a significant improvement.

3.3. Hybrid Learning

Yun Z et al. [14] combine six standard machine learning algorithms, LR, radial basis function network (RBF Network), MLP, BN, decision and trees and decision tables (DTS) to make decisions based on average voting and maximum voting methods. They conduct experiments on ten open-source cross-project software systems from the PROMISE repository and achieve outstanding results, in which the maximum voting is specifically prominent.

In Jian L et al. [28]'s research, seven open-source Java programs from the PROMISE repository were processed by the convolution neural network based on ASTs and combined with traditional prediction learning methods for deep semantic and syntactic learning. The result shows a 12% improvement over the state-of-the-art method. It is a successful attempt to combine deep learning with traditional machine learning.

Pushphavathi T P et al. [29] use the genetic algorithm (GA) in combination with the fuzzy c-means (FCM) classifier and RF classifier to learn datasets from NASA. When only FCM classifier is used, there is an accuracy of 76.2%, and the combination of GA in the FCM classifier achieves 90.12% accuracy. A combination of GA, FCM, and the RF can achieve an astounding 98.23% accuracy.

Zhou X et al. [30] apply the hybrid active learning and kernel PCA (HALKP) based cross-version defect prediction on 31 versions of 10 software projects taken from the open MORPH datasets. This method leverages hybrid active learning to obtain some unlabeled modules from the current version, then merges them into the labeled modules of the prior version， thus acquires an enhanced training set. Then a non-linear mapping method, HALKP, is utilized to extract representative features by embedding the original data of two versions into a high-dimension space. Compared with the traditional methods, the learning result of the hybrid method is remarkably improved.

Xinli Yang et al. [31] use the two-layer ensemble learning method whose inner layer is an RF that is made up of decision tree and bagging with adjustable weight. The outer layer is stacking, which is made up of multiple equal weights random under-sampling and is tested with 137,417 changes from six open large source projects. When compared with other just-in-time studies, the cost-effectiveness of the proposed method has achieved a 20% improvement.

4. Discussion

Defect prediction using machine learning includes various machine learning techniques that are based on probability, geometry, and logic model with distinct characteristics.

4.1. Overview

Table 1. Research work overview.

Research	Technique	Achievement	Prediction type
Luca P et al.[19]	RF	F-measure and AUC-ROC	Just-in-time
Yeongjun C et al.[18]	RF LR NB	Cost effectiveness	CPDP and just-in-time
Osman H et al. [20]	IBK SVM	Accuracy	Hyperparameter Optimization
Sokratis T et al.[21]	C4.5 NB BN LR	TPR ACC and PPV	Real time system
Puja A H et al. [22]	NB	Accuracy and low FA	CPDP
Yanhong Y et al.[23]	Cluster Ensembles	Recall and F-measure	CPDP
Kishan K G et al. [24]	NB LR DT RF MP	F-measure	
Lipika G et al. [25]	Gradient Boosting	Accuracy, Recall, and F1-score	CPDP
Wang S et al.[9]	DBN	precision, Recall, and F1	WPDP and CPDP
Thong H et al.[10]	CNN	AUC	Just-in-time
Hoa K D et al.[11]	RNN	Precision, Recall, and F1	WPDP and CPDP
Zhen L et al.[12]	RNN	Fewer false negatives and false positives	
Anh V P et al.[26]	CNN	Accuracy	
Xuan H et al.[13]	CNN	F-measure	WPDP and CPDP
Anh V P et al. [27]	CNN	Accuracy and F1	
Yun Z et al.[14]	LR RBF MLP BN	Combined classifier	CPDP

	DT DTs	average F-measure	
Jian L et al.[28]	CNN LR	F-measure	
Pushphavathi T P [29]	GA FCM RF	Accuracy: GA+FCM+RF > GA+FCM > FCM	
Zhou X et al.[30]	HALKP	F-measure, g-mean balance	Cross-version
Xinli Y et al.[31]	RF stacking	Cost effectiveness.	Just-in-time

4.2. Work Analysis

As can be seen from Table 1, naive Bayes [18][21-23] is the most widely used method. Naïve Bayes is a classification method based on the Bayes' Theorem and the characteristic independent hypothesis, which is derived from the classical mathematical theory, has a solid mathematical foundation and is the most typical probability model. It has good cost-effectiveness, which is suitable for the development of defect prediction.

Bayesian network [14][21] is an extension of the Bayesian method with the advantage of the probability model and is quite useful in the defect prediction. It is one of the most effective theoretical models in the field of uncertain knowledge expression and reasoning. This structure is similar to that of the software's control flow graph and data flow graph so that it can perform the defect prediction better. However, the network construction and operation of the Bayesian network are relatively complex, and it has increased uncertainty, which limits the application of the Bayesian network.

SVM [20] has many unique advantages in the small sample, nonlinear and high dimension processing, especially the character and ability of high dimension processing, which promotes the development of metrics in defect prediction field and gets better prediction performance.

RF [18][19][24] is the most commonly used ensemble learning method, the core of which is decision trees. Decision trees are widely used in defect prediction [21][24]. The edge of the random forest includes easy to understand and small computation. RF adds a voting mechanism that leverages the advantage of decision trees so that it can obtain an unbiased estimation of internal errors and improve the accuracy of the algorithm. RF also has a significant improvement in the learning ability of a limited sample, with a distinct advantage in just-in-time defect prediction.

DBN [9] is the earliest deep learning method, which is based on neural networks. Traditional machine learning's feature to describe sample is usually manually selected; thus, the performance of generalization is greatly affected by the quality of features. Deep learning is the process of generating useful features by itself through machine learning techniques. Consequently, it can also be understood as "feature learning." DBN is a probability generation model and consist of multiple RBM layers that contain a visible layer and multiple hidden layers. There are connections between the layers without connections between the cells within the layers. Therefore, through the training process carried out in the invisible layers, DBN can capture the correlation of high-order data. Therefore, it has the advantage of the probability model in defect prediction.

CNN [10][13][26-27] is the most commonly used method of deep learning. The concept behind CNN is an imitation of the biological visual perception system, which is closer to the real nervous system. By learning to generate input-output mappings, CNN can produce high performance and excellent generalization performance with enough training samples. To get more sample sets, Xuan H et al. [9] combine the annotation information with the defect prediction and significantly improve the prediction performance.

RNN [11-12] is adept at dealing with the features of time sequences, which is consistent with the time characteristics of software. The program code not only has the space character of memory allocation but also has the time character of operation processing. This approach is highly capable of predicting both syntax defects as well as semantic defects.

It is not difficult to see that every learning algorithm has an advantage. Much research has made meaningful attempts to combine different methods' advantages, and this is the basic idea of hybrid learning [14][30][28][29][31]. An easy way in hybrid learning is to average the votes with multiple algorithms or select the maximum. For instance, Yun Z et al. [14] carry out this kind of research and obtain satisfying results. Xinli Y et al. [31] use decision trees to construct RF, and then apply random under-sampling to improve the processing speed. Finally, they use multiple RFs stacking to adjust the weight and increase accuracy. As a result, they reach excellent just-in-time performance. Jian L et al. [28] use CNN to generate a new feature, and combining it with traditional features, then use LR classifier to predict defects. Compared with the sole DBN method and methods using traditional features, the prediction performance has been improved dramatically.

5. Conclusion

In this paper, we summarize the research works of defect prediction in recent years, mainly, in the perspective of the techniques they utilized, by which the research works are divided into three categories: traditional machine learning-based, deep learning-based and hybrid learning-based. The contents and characteristics of the works are discussed in detail. Also, the advantages of different techniques based works are correspondingly analyzed. To sum up, with 20 years' development of defect prediction, the traditional machine learning method has been prevalently used in defect prediction as well as location. In the field of network development and application, the requirement of real-time defect prediction is more and more demanding. As a result, the research on just-in-time is valued. Deep learning and hybrid learning have produced numbers of state-of-art methods that can significantly improve prediction performance, aiming to predict defects of both cross-project and within-project.

References

[1] Clarke A C 1985 *The Promise of space*（New York: Berkley Books）

[2] Dressler J 2007 *Cases and Materials on Criminal Law*（St. Paul MN: Thomson/West）

[3] Coverity Scan *https://scan.coverity.com/*

[4] Fortify Static Code Analysis Tool: Static Application Security Testing *https://www.microfocus.com/en-us/products/static-code-analysis-sast/overview*

[5] Valgrind: Instrumentation framework for building dynamic analysis tools *https://sourceware.org/git/?p=valgrind.git*

[6] Ftrace-Function Tracer *https://www.kernel.org/doc/Documentation/trace/ftrace.txt*

[7] Keller T and Schneidewind N 1997 *Successful Application of Software Reliability Engineering for the NASA Space Shuttle Computer Standards & Interfaces* **21(2)** pp 71-82

[8] Venkata U B C, Farokh B B, I-Ling Y and Raymond A P 2005 Empirical assessment of machine learning based software defect prediction techniques *10th IEEE Int. Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'05)*

[9] Wang S, Liu T and Tan L 2016 Automatically learning semantic features for defect prediction *38th Int. Conf. on Software Engineering (ICSE)* pp 297-308

[10] Thong H, Hoa K D, Yasutaka K, David L and Naoyasu U 2019 DeepJIT: an end-to-end deep learning framework for just-in-time defect prediction *MSR 2019 Technical Papers*

[11] Hoa K D, Truyen T, Trang P, Shien W N, John G and Aditya G 2017 Automatic feature learning for vulnerability prediction *IEEE Transactions on Software Engineering*

[12] Zhen L, Deqing Z, Shouhuai X, Xinyu O, Hai J,Sujuan W, Zhijun D and Yuyi Z 2018 VulDeePecker: a deep learning-based system for vulnerability detection *2018 Network and Distributed Systems Security (NDSS) Symp.*

[13] Xuan H, Yang Y, Ming L and De-Chuan Z 2018 Learning semantic features for software defect prediction by code comments embedding *2018 IEEE Int. Conf. on Data Mining*

[14] Yun Z, David L, Xin X and Jianling S 2018 Combined classifer for cross-project defect prediction: an extended empirical study *Frontiers of Computer Science -Springer-.* **12(2)** pp 280-96

[15] 1992 *Proc. of the fifth annual workshop on Computational learning theory* (New York: ACM)

[16] Dreyfus and Stuart E 1990 Artificial neural networks, back propagation, and the Kelley-Bryson gradient procedure *Journal of Guidance, Control, and Dynamics*. **13 (5)**

[17] Hinton G E, Osindero S and Teh Y 2006 A fast learning algorithm for deep belief nets *Neural Computation* **18(7)**

[18] Yeongjun C, Jung-Hyun K and In-Young K 2018 Cross-sub-project just-in-time defect prediction on multi-repo projects *6th Int. Workshop on Quantitative Approaches to software Quality* pp 2-9

[19] Luca P, Fabio P and Alberto B 2019 Fine-grained just-in-time defect prediction *The Journal of Systems and Software* pp 22-36

[20] Osman H, Ghafari M and Nierstrasz O 2017 Hyperparameter optimization to improve bug prediction accuracy *MaLTeSQuE*

[21] Sokratis T, Andriy M and Elie M 2016 On automatic detection of performance bugs *27th Int. Symp. on Software Reliability Engineering Workshops*

[22] Puja A H, Victor A and Rizal B B 2018 Cross-project defect prediction for web application using naive Bayes *IWBIS*

[23] Yanhong Y, Jun Y and Hongbing Q 2018 Defect prediction by using cluster ensembles *10th Int. Conf. on Advanced Computational Intelligence (ICACI)* pp 29–31

[24] Kishan K G and B M Mainul H 2018 Evaluating the effectiveness of conventional machine learning techniques for defect prediction: a comparative study *Joint 7th Int. Conf. on Informatics, Electronics & Vision and 2nd Int. Conf. on Imaging, Vision & Pattern Recognition*

[25] Lipika G, Mayank S and Sunil K K 2018 Implementation of data sampling in class imbalance learning for cross project defect prediction: an empirical study *5th Int. Symp. on Innovation in Information and Comm. Technology (ISIICT)*

[26] Anh V P, Minh L N and Lam T B 2017 Convolutional neural networks over control flow graphs for software defect prediction *Int. Conf. on Tools with Artificial Intelligence*

[27] Anh V P and Minh L N 2017 Convolutional neural networks on assembly code for predicting software defects *21st Asia Pacific Symp. on Intelligent and Evolutionary Systems (IES)*

[28] Jian L, Pinjia H, Jieming Z and Michael R L 2017 Software defect prediction via convolutional neural network *Int. Conf. on Software Quality, Reliability and Security.*

[29] Pushphavathi T P 2017 An approach for software defect prediction by combined soft computing *Int. Conf. on Energy, Comm., Data Analytics and Soft Computing (ICECDS)*

[30] Zhou X, Jin L, Xiapu L and Tao Z 2018 Cross-version defect prediction via hybrid active learning with kernel principal component analysis *25th Int. Conf. on Software Analysis, Evolution and Reengineering (SANER) IEEE Computer Society*

[31] Xinli Y, David L, Xin X and Jianling S 2017 A two-layer ensemble learning approach for just-in-time defect prediction *Information and Software Technology* **87** pp 206-20