A Multiple-Criteria Ensemble Weight Strategy to Increase the Effectiveness of Deep Learning-based Fault Localization

Chih-Chiang Fang and Chin-Yu Huang*
Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
neilfang112113@gmail.com, cyhuang@cs.nthu.edu.tw
*corresponding author

Abstract—Fault localization (FL) is an essential phase in software debugging and is used to detect the possible faulty location. Nowadays, deep learning technique has a great advantage to extract semantic features of program and hidden data information for coverage data effectively. Therefore, deep learning-based FL is regarded as the latest solution. However, it is well known that none of existing methods is suitable to all scenarios including single fault and multiple faults. Furthermore, the optimal hyperparameters of deep learning model for FL are hard to be determined in different types of programs. According to past researches, Multicriteria decision-Making (MCDM) can provide the best or trade-off solutions to multiple method combinations. In this paper, we combined multiple-criteria ensemble weight strategy with deep learning-based FL to improve the effectiveness of program. Overall, our proposed method has also a good scalability and run some experimental results to obtain significant performance improvement comparisons among the past methods.

Keywords—Fault Localization; Deep Learning; Hyperparamters; Multicriteria Decision-Making; Software Debugging

1. Introduction

Software debugging is very difficult task and how to find the location of faulty statement is critical topic. That is called fault localization (FL) problem[1][2][3]. Recently, there are many fault localization methods to be proposed and classified them into three most popular types: 1) Spectrum-Based Fault Localization (SBFL) techniques, 2) Mutation-Based Fault Localization (MBFL) techniques, and 3) Deep Learning-Based Fault Localization [4][5] techniques that are used to identify the faulty location accurately and effectively.

In general, faulty program can be classified into single fault and multiple faults in real world. We also do not know how many faults are in this program. According to some investigations [1][5][6][7], most FLs are not suitable to multiple faults directly and need to perform additional operations such as running FL process for individual failed test case and sufficient passed test cases combination together, various distance matrix, clustering, iteration and so on. Furthermore, none of existing methods is suitable to all cases. Actually, each program has different characteristics from different software developers and programming language aspects. To effectively extract semantic feature of program, selecting deep learning-based FL method is the best solution among these methods. Based on the stated reasons, we must design a novel approach to meet these requirements and can reflect the results quickly. In this paper, we combined multiple-criteria ensemble weight strategy with deep learning-based FL to improve the effectiveness of program. On the other hand, we analyzed the execution time of multiple-criteria ensemble weight added into deep learning-based FL. The preliminary experiment has shown that our proposed method is effective and can improve the overall performance.

2. Our Propsed Approach

The detailed processing diagram of our proposed method is illustrated in Figure 1.

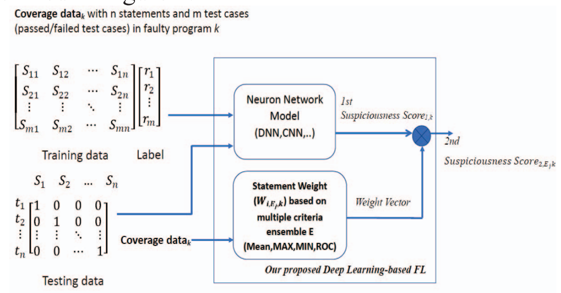
Coverage data$_k$ with n statements and m test cases (passed/failed test cases) in faulty program k
S_{11} S_{12} … S_{1n}; S_{21} S_{22} … S_{2n}; S_{m1} S_{m2} … S_{mn}
r_1 r_2 … r_m
Training data
Label
S_1 S_2 … S_n
t_1 t_2 … t_n
Testing data
Coverage data$_k$
Neuron Network Model (DNN,CNN,..)
1st Suspiciousness Score$_{1,k}$
2nd Suspiciousness Score$_{2,E_jk}$
Statement Weight ($W_{i,E_j,k}$) based on multiple criteria ensemble E (Mean,MAX,MIN,ROC)
Weight Vector
Our proposed Deep Learning-based FL

Figure 1. Combining multiple criteria ensemble wight with deep learning-based FL

Our proposed approach can be divided into the following steps:

Step 1. Collect the coverage data of program k using a given number of test cases (fail or pass).

Step 2. The coverage data$_k$ can be regarded as training data to deep learning-based FL. Also, the virtual test cases [4] are regarded as testing data.

Step 3. It is noted that each statement i itself should be different weight value and can be confirmed effectively in this step. Here, we use multi-criteria decision-making concepts[8][9], ensemble classifier approach [10] and the lower EXAM/AVG EXAM score to derive it. Furthermore, we discuss four common cases that are Mean, MAX, MIN, and ROC (Rank-Order Centroid) value respectively. Assume that there are p methods (M_1,M_2,..M_p) to find the location of faulty statement separately. Each method has its advantage and disadvantage for the certain situation. Combining these methods with deep learning-based FL is an alternative solution to derive some possible confident weight values of each statement that are defined as (1)-(4).

$$\boldsymbol{w_{i,E_1,k}} = \left(1 + \mathrm{M}EAN\,(M_{1,i}, M_{2,i}, \dots, M_{p,i})\right) \quad (1)$$

$$\boldsymbol{w_{i,E_2,k}} = \left(1 + \mathrm{MAX}\,(M_{1,i}, M_{2,i}, \dots, M_{p,i})\right) \quad (2)$$

$$\boldsymbol{w_{i,E_3,k}} = \left(1 + \mathrm{MIN}\,(M_{1,i}, M_{2,i}, \dots, M_{p,i})\right) \quad (3)$$

$$\boldsymbol{w}_{\boldsymbol{i},\boldsymbol{E_4},\boldsymbol{k}} = \left(\ 1 + \mathrm{ROC}\,(M_{1,i}, M_{2,i}, \dots, M_{p,i})\right) \qquad (4)$$

where $\boldsymbol{w}_{\boldsymbol{i},\boldsymbol{E_1},\boldsymbol{k}}$ is ensemble weight 1 that takes average value of these methods for statement *i* in faulty program k, $\boldsymbol{w}_{\boldsymbol{i},\boldsymbol{E_2},\boldsymbol{k}}$ is ensemble weight 2 that takes maximum value of these methods, $\boldsymbol{w}_{\boldsymbol{i},\boldsymbol{E_3},\boldsymbol{k}}$ is ensemble weight 3 that takes minimum of these methods, $\boldsymbol{w}_{\boldsymbol{i},\boldsymbol{E_4},\boldsymbol{k}}$ is ensemble weight 4 that takes ROC of these methods, and initial value of statement *i* is 1. If statement *i* is only visited by some failed test cases or failed test cases and passed test cases together, its weight value must be re-adjusted. Conversely, it is not visited by any test cases and keep original value. The maximum value of adjusted weight is 2. Equation (4), we also have to know these estimated method orders in advance. Select one of these methods with the lowest EXAM/AVG EXAM score as M_1, the second lower score as M_2 and so on. Then we use ROC to derive the confident weight value of statement *i.*

Step 4. Select a suitable deep learning model to predict the output value of each statement *i in program k*. In this case, the first suspiciousness scores of faulty programs are generated by selected deep learning model.

Step 5. The second suspiciousness score of statement *i* in *program k* is the product of the output of selected deep learning model and different ensemble weight vector *j* from step 3. This step can re-order the rank of each statement to locate possible faulty statements quickly. If the hyperparameters of deep learning model cannot be found, the predicted output may be worse. At the time, this step can alleviate the hyperparameter effects. When this step is completed, it is obvious that the highest suspiciousness score is possible faulty statement.

3. EXPERIMENTAL EVALUATION

In this section, our experimental environments are listed below: 1) AMD Ryzen 7 5000 series, 2) 16GB of RAM, 3) NVIDIA GPU RTX 3060, and 4) two basic deep learning models (DNN, CNN) are implemented by using TensorFlow API 2.5 in python programming language. We selected two published datasets that are Gzip and Grep from SIR.

3.1. Average Processing Time of Multiple Criteria Ensemble Weight Analysis

While considering ensemble weight to deep learning-based FL, analyzing its execution time distribution is necessary. Table 1 shows the average execution time of each component for ensemble weight approach. As can be seen that the average processing time of multiple criteria ensemble weights was close to **0.91s/1.4s** for Gzip and Grep dataset.

Table 1. Average execution time for multiple criteria ensemble weight

	Various weight calculations for effective statements (ms)	*lower EXAM score calculation (5 methods) find TOP 3 methods and orders x^a,y^b (ms) (a:1 fault, b:4 faults)*	*The product of weight vector and test predicted output*	*Various wight calculations for all statements (ms)*
Gzip	864.218	17.345,37.578	ignored	4389.491
Grep	1354.733	16.089,40.489	ignored	8852.6

3.2. Preliminary Result and Discussion

In this experiment, we compared the effectiveness of some SBFLs, CNN,DNN and our proposed method for distinct faulty versions of Gzip as illustrated Figure 2. As can be seen that our multiple criteria ensemble weight is significant improvement for the second suspiciousness score output. It is proven that the hyperparameter problems of two basic deep learning models can be alleviated and effectively find the possible faulty statement.

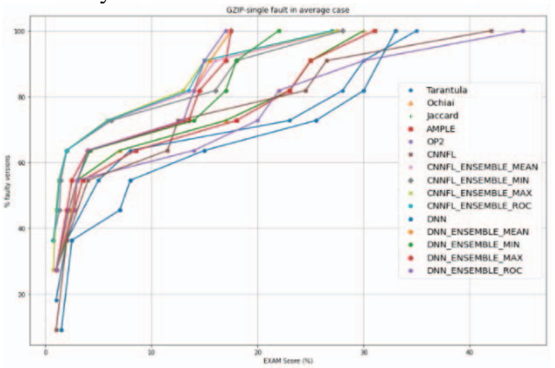

Figure 2. The effectiveness comparison among different methods for Gzip

4. CONCLUSION

In this paper, we tried to combine multiple criteria ensemble weight with deep learning-based FL to obtain the better suspiciousness score of faulty statement. The preliminary experiment has shown this approach is feasible. Furthermore, the overhead of our proposed approach is relatively small and effectively alleviate the hyperparameter problems. We are planning to perform more diverse datasets to validate the correctness and effectiveness. In the future, we will also investigate the effects from different kinds of deep learning models.

REFERENCES

[1] W. E. Wong, R. Gao, Y. Li, R. Abreu and F. Wotawa, "A Survey on Software Fault Localization," IEEE Transactions on Software Engineering, vol. 42, no. 8, pp. 707-740, 1 Aug. 2016

[2] W. E. Wong and V. Debroy, "Software Fault Localization," Encyclopedia of Software Engineering, vol. 1, pp. 1147-56, Sep. 2010

[3] W. E. Wong and T. H. Tse, "Handbook of Software Fault Localization: Foundations and Advances," Edition 1, Wiley-IEEE Computer Society Press, May 2023

[4] Z. Zhang,Y. Lei,X. Mao, and P. Li, "CNN-FL: An Effective Approach for Localizing Faults using Convolutional Neural Networks," *Proceedings of the 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering* (SANER), pp.445-455, Hangzhou, China, 2019.

[5] A. Dutta, R. Manral, P. Mitra, and R. Mall, "Hierarchically Localizing Software Faults Using DNN," *IEEE Transactions on Reliability*, Vol. 69, No. 4, pp. 1267-1292, Dec. 2020.

[6] Y. Li and P. Liu, "A Preliminary Investigation on the Performance of SBFL Techniques and Distance Metrics in Parallel Fault Localization," *IEEE Transactions on Reliability*, vol. 71, no. 2, pp. 803-817, June 2022.

[7] M. Zhang, S. Wang and W. Qiu, "A Software Multi-fault Locating Technique based on Space Shrinkage," *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*, Wulumuqi, China, pp. 853-858,2022.

[8] N. Aribi, N. Lazaar, Y. Lebbah, S. Loudni and M. Maamar, "A Multiple Fault Localization Approach Based on Multicriteria Analytical Hierarchy Process," *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, Newark, CA, USA, pp. 1-8, 2019.

[9] M. Danielson, and L. Ekenberg. "Trade-Offs for Ordinal Ranking Methods in Multi-criteria Decisions," *GDN 2016. Lecture Notes in Business Information Processing*. Vol.274. pp.16-27, 2017.

[10] A. Dutta, "Poster: EBFL-An Ensemble Classifier based Fault Localization," 2022 IEEE Conference on Software Testing, Verification and Validation (ICST), Valencia, Spain, pp. 473-476,2022.