International Journal of Mathematical, Engineering and Management Sciences
Vol. 8, No. 4, 632-639, 2023
https://doi.org/10.33889/IJMEMS.2023.8.4.036

Deep Learning Based on Fine Tuning with Application to the Reliability Assessment of Similar Open Source Software

Yoshinobu Tamura
Graduate School of Sciences and Technology for Innovation,
Yamaguchi University, Ube, Yamaguchi, Japan.
Corresponding author: tamuray@yamaguchi-u.ac.jp

Shigeru Yamada
Graduate School of Engineering,
Tottori University, Tottori, Tottori, Japan.
E-mail: bex2yama@muse.ocn.ne.jp

(Received on September 28, 2022; Accepted on April 18, 2023)

Abstract
Recently, many open-source products have been used under the situations of general software development, because the cost saving and standardization. Therefore, many open-source products are gathering attention from many software development companies. Then, the reliability/quality of open-source products becomes very important factor for the software development. This paper focuses on the reliability/quality evaluation of open-source products. In particular, the large quantity fault data sets recorded on Bugzilla of open-source products is used in many open-source development projects. Then, the large amount of data sets of software faults is recorded on the Bugzilla. This paper proposes the reliability/quality evaluation approach based on the deep machine learning by using the large quantity fault data on the Bugzilla. Moreover, the large quantity fault data sets are analyzed by the deep machine learning based on the fine-tuning.

Keywords- Open-source software, Deep learning, Fine tuning, Similar open-source software.

1. Introduction

Many open-source products are used in many software development companies. The open-source product is useful for many software companies to make a standardization and cost reduction of development. On the other hand, the open-source product has several quality problems, because there is no specific testing phase (Li et al., 2005; Anbalagan and Vouk, 2008; Lee et al., 2008; Syed-Mohamad and McBride, 2008; Yadav et al., 2011; Razzaq et al., 2018; Rammohan and Rajesh, 2020).

Especially, Bugzilla's used in many open-source projects. The Bugzilla's useful for software developers to assess the reliability/quality of open-source products. However, the large quantity of software fault data recorded on the Bugzilla is not utilized by the developers and users. The problem of reliability/quality for open-source products will be resolved if the large quantity database of Bugzilla is used efficiently (Kaur, and Jindal, 2017; Gopal et al., 2022, Tuna et al., 2022).

Especially, there is no research paper in terms of the reliability/quality evaluation approach of open-source products based on deep machine learning by using the large quantity fault data sets obtained from Bugzilla's. Our research group has proposed several reliability/quality evaluation approaches based on deep machine learning. In particular, this paper newly applies fine-tuning to our deep machine learning approach. Thereby, this approach of the proposed approach will be able to reduce the calculation cost by deep machine learning. Then, the estimated accuracy keeps the same value.

The structure of this paper is shown as follows. Section 2 shows the literature review. Section 3 introduces our deep machine learning modeling based on fine-tuning by using the model structures. Section 4 shows several numerical illustrations based on the proposed model by using the actual data sets. Section 5 discusses the results of the proposed approach.

This paper discusses the methodology of open-source product reliability/quality evaluation by using several open-source products. In particular, this paper discusses the applicability of transfer-learning and fine-tuning for the open-source software reliability/quality evaluation by using the deep machine learning model. Furthermore, this paper shows several numerical illustrations based on the proposed deep machine learning model.

2. Literature Review

Several research papers on reliability/quality evaluation approaches for open-source products have been proposed in the past (Li et al., 2005; Anbalagan and Vouk, 2008; Lee et al., 2008; Syed-Mohamad and McBride, 2008; Yadav et al., 2011; Razzaq et al., 2018; Rammohan and Rajesh, 2020). These papers are the reliability/quality evaluation approach by using. However, there is no proposed a part of the data sets obtained from the Bugzilla. There is no proposed the reliability/quality evaluation approach by using the large quantity fault data. We have proposed several reliability/quality evaluation approaches based on the deep machine learning in the past (Tamura and Yamada, 2021, 2022; Yanagisawa et al., 2021). Actually, there is no research paper in terms of the reliability/quality evaluation based on the deep machine learning. In particular, the reliability/quality evaluation approach based on deep machine learning by using the large quantity fault data sets obtained from the Bugzilla's has no proposed in the past. The characteristic of this paper is the fully connected deep neural network. In the research area of OSS reliability/quality evaluation, it will not be able to find the paper in terms of the OSS reliability/quality evaluation approach based on the fully connected deep neural network by using the OSS large quantity fault data.

3. Deep Machine Learning Modeling Based on Fine-Tuning

Many approaches of deep machine learning are frequently used in the area of image recognition. In particular, many researchers have proposed many approaches based on the fine-tuning. Various existing learning models such as InceptionResNetV2, VGG16, and VGG19 (Tan et al., 2018; Vrbančič and Podgorelec, 2020; Ohira et al., 2021). These existing learning models such as InceptionResNetV2, VGG16, and VGG19 are used for the image recognition area, respectively. The image recognition area focuses on the similar figures. However, we focus on the large sized fault data on the Bugzilla of OSS. There is no existing learned model in case of the fault data. Therefore, we need to re-make the re-transferred model of deep machine learning. Then, we make the re-learned model from scratch as follows:

(i) All large quantity data is divided into two data sets based on a certain version.
(ii) The data preprocessing is made for two data sets.
(iii) The learned model is made from 1st certain version data.
(iv) The learned model is saved for using in the next model.
(v) The weight parameter of learned model is added to new layers of the existing learned model.
(vi) New model from added learned model is structured by using 2nd certain version of data.
(vii) The estimate is provided by using new learned-model.

This paper shows two approaches consists of the fine-tuning and the transfer-learning in terms of the deep machine learning approach.

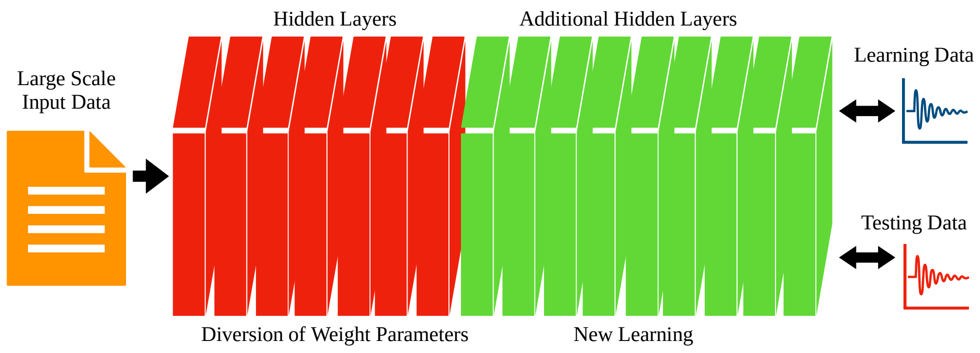

Figure 1. The structure of deep machine learning using the fine-tuning.

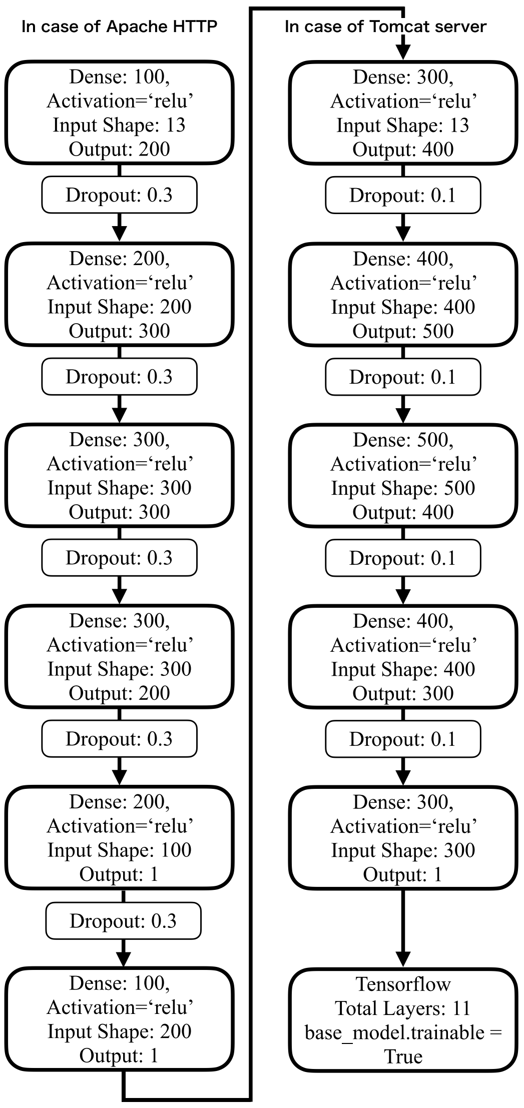
In case of Apache HTTP
Dense: 100, Activation='relu' Input Shape: 13 Output: 200
Dropout: 0.3
Dense: 200, Activation='relu' Input Shape: 200 Output: 300
Dropout: 0.3
Dense: 300, Activation='relu' Input Shape: 300 Output: 300
Dropout: 0.3
Dense: 300, Activation='relu' Input Shape: 300 Output: 200
Dropout: 0.3
Dense: 200, Activation='relu' Input Shape: 100 Output: 1
Dropout: 0.3
Dense: 100, Activation='relu' Input Shape: 200 Output: 1
In case of Tomcat server
Dense: 300, Activation='relu' Input Shape: 13 Output: 400
Dropout: 0.1
Dense: 400, Activation='relu' Input Shape: 400 Output: 500
Dropout: 0.1
Dense: 500, Activation='relu' Input Shape: 500 Output: 400
Dropout: 0.1
Dense: 400, Activation='relu' Input Shape: 400 Output: 300
Dropout: 0.1
Dense: 300, Activation='relu' Input Shape: 300 Output: 1
Tensorflow Total Layers: 11 base_model.trainable = True

Figure 2. The structure of our model using the fine-tuning.

Figure 1 is the structure of deep machine learning based on fine-tuning. Then, this paper shows the structure of our model using fine-tuning in Figure 2. Figure 2 is empirically structured by trial and error from the large quantity fault data set. Our deep machine learning model has two learning stage structured from Apache HTTP server and Tomcat open-source product.

4. Numerical Examples

This paper focuses on Apache HTTP server (The Apache Software Foundation, The Apache HTTP Server Project) and Tomcat server (The Apache Software Foundation, The Apache Tomcat Project) of open-source product. This paper shows numerical examples by using the data sets on the Bugzilla of Apache HTTP server and Tomcat. These data sets in this paper are collected in the Bugzilla. This open-source product is similar to software in terms of Web-network-oriented open-source product. Thereby, it can prepare two data sets for the transfer-learning and fine-tuning. Thus, the estimation result based on the transfer-learning and the estimation results based on the fine-tuning is compared by using data sets of the Apache HTTP server and Tomcat server.

This paper shows the training and validation errors by the deep machine learning in transfer-learning in Figure 3. Also, the training and validation errors by the deep machine learning in fine-tuning is shown in Figure 4. Figures 3 and 4 show that the error in the case of fine-tuning is smaller than one of transfer learning. Therefore, the learning condition of fine-tuning is good. Generally, it is well known that the results fine-tuning become good than the conventional transfer learning. Moreover, several numerical examples in case of the fine-tuning are shown in this paper. Also, Figure 5 is the estimated instantaneous times for failures by the deep machine learning in fine-tuning. Furthermore, the scatter plot for actual testing data and estimate by the deep machine learning in fine-tuning is shown in Figure 6. The estimate becomes smaller than the actual data from Figure 6. Figure 7 is the estimated cumulative time of software failure by the deep machine learning based on fine-tuning. Similarly, Figure 8 is the estimated cumulative time of software failure by the deep machine learning based on fine-tuning. From Figures 7 and 8, we found that the proposed approach can exactly estimate until 500 faults in the long time future.

As the results of all figures, Figure 3 and 4 mean that the results of the error in fine-tuning are better than the transfer-learning. In particular, Figure 8 shows that the estimation results are good until the point by 750 faults. The approaches of neural networks and deep machine learning could not estimate the long terms period such as 750 faults. This paper enables us to estimate the long term prediction.

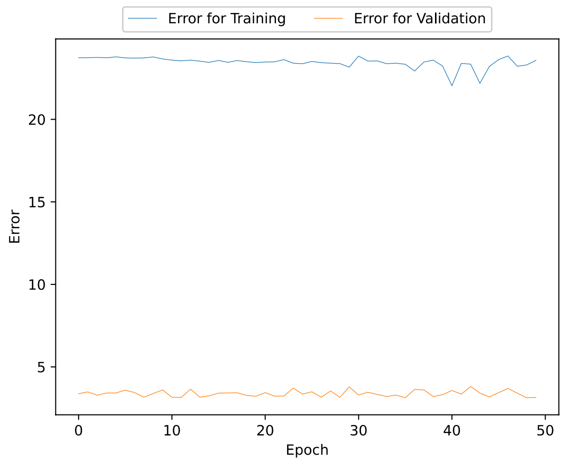

Figure 3. The training and validation errors by deep machine learning in transfer-learning.

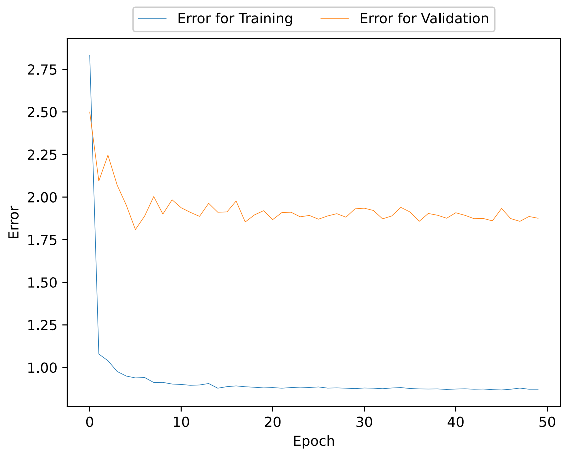

Figure 4. The training and validation errors by deep machine learning in fine-tuning.

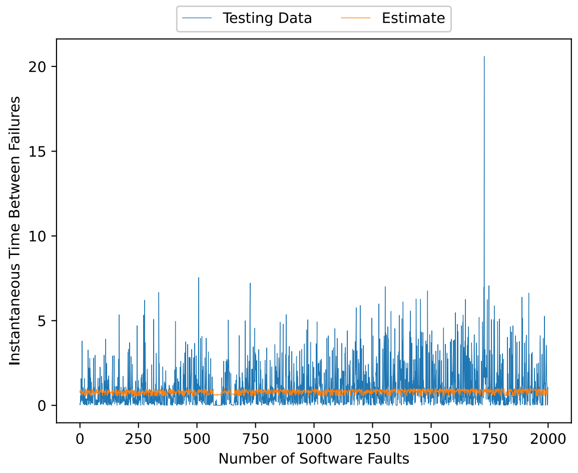

Figure 5. The estimated instantaneous times for failures by deep machine learning in fine-tuning.

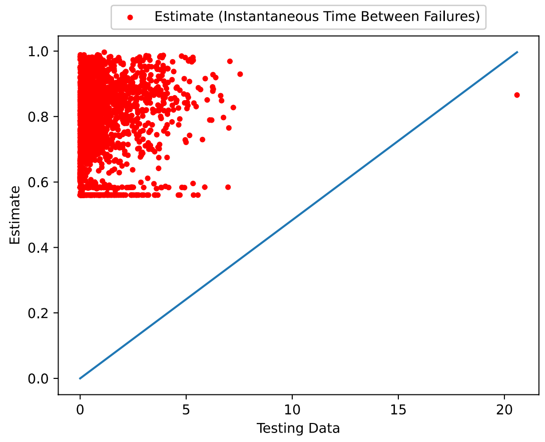

Figure 6. The scatter plot for actual testing data and estimate by the deep machine learning in fine-tuning.

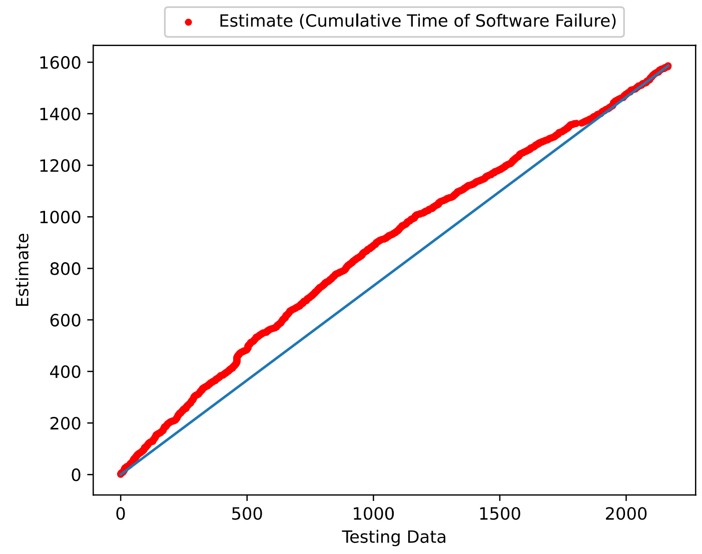

Figure 7. The estimated cumulative times of software failure by deep machine learning in fine-tuning.

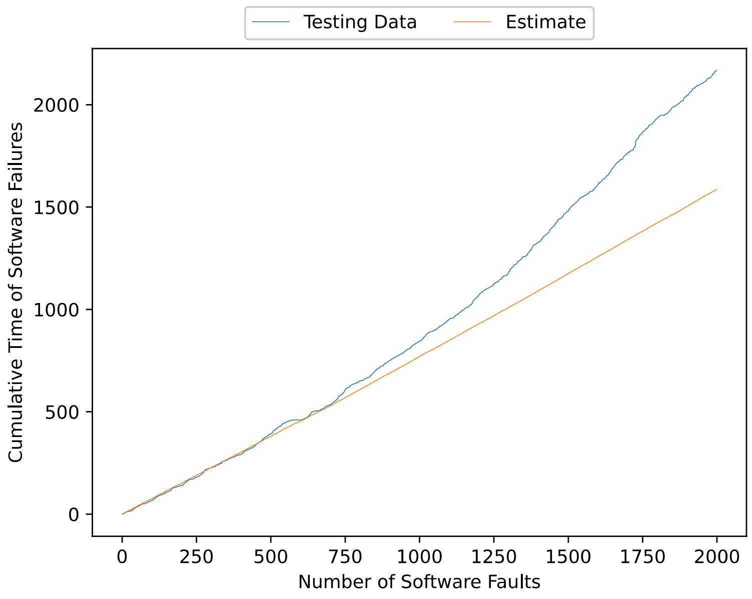

Figure 8. The estimated cumulative times of software failure by the deep machine learning in fine-tuning.

5. Conclusion

Many open-source products have been developed by many developers and users. Then, we consider that the similar software can use as a large quantity fault data by the fine-tuning technique. Also, the fine-tuning technique is helpful to assess the software reliability/quality, because the fine-tuning is widely used in the research area of image recognition. In particular, we focus on the characteristics of the similar open-source product. We have considered several cases of similar open-source products in numerical examples.

From Figure 3 and Figure 4, this paper found that the learning error becomes very small by using the fine-tuning approach. This means that it is useful to use the fine-tuning from the standpoint of cost reduction and time saving in deep machine learning. In the image recognition area, there are many packages for fine-tuning. However, there is no packages for the fine-tuning in the area of Bugzilla in open-source product. Especially, as the novelty of the work, the proposed approach can apply the fine-tuning to the large quantity of fault data obtained from the Bugzilla. Then, this paper has made the original learning weight data from the large quantity of fault data. In the past, our research group has proposed several reliability/quality evaluation approaches based on the deep machine learning in the past (Tamura and Yamada, 2021, 2022; Yanagisawa et al., 2021). However, there is the problem that the learning time is long. This paper has solved this problem by using the fine-tuning approach.

This paper has discussed an approach of reliability/quality evaluation by using the large quantity of fault data sets obtained from several open-source products. In particular, the transfer-learning and fine-tuning have been compared in order to assess the reliability/quality of the deep machine-learning model. Furthermore, we have shown several numerical examples based on our deep machine-learning model by using actual large-quantity fault data sets in this paper.

Conflict of Interest
The authors declare no conflict of interest.

Acknowledgments
This work was supported in part by the JSPS KAKENHI Grant No. 23K11066 in Japan.

References

Anbalagan, P., & Vouk, M. (2008). On reliability analysis of open source software-fedora. In *2008 19th International Symposium on Software Reliability Engineering (ISSRE)* (pp. 325-326). IEEE Computer Society.

Gopal, M.K., Govindaraj, M., Chandra, P., Shetty, P., & Raj, S. (2022). Bugtrac–a new improved bug tracking system. In *2022 IEEE Delhi Section Conference (DELCON)* (pp. 1-7). IEEE. New Delhi, India.

Kaur, A., & Jindal, S.G. (2017). Bug report collection system (BRCS). In *2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence* (pp. 697-701). IEEE. Noida, India.

Lee, W., Jung, B.G., & Baik, J. (2008). Early reliability prediction: An approach to software reliability assessment in open software adoption stage. In *2008 Second International Conference on Secure System Integration and Reliability Improvement* (pp. 226-227). IEEE. Yokohama, Japan.

Li, P.L., Herbsleb, J., & Shaw, M. (2005). Finding predictors of field defects for open source software systems in commonly available data sources: A case study of openbsd. In *11th IEEE International Software Metrics Symposium (METRICS'05)* (pp. 10-pp). IEEE. Como, Italy.

Ohira, M., Takano, K., & Ma, Z. (2021). A novel deep-Q-network-based fine-tuning approach for planar bandpass filter design. *IEEE Microwave and Wireless Components Letters*, *31*(6), 638-641.

Rammohan, D., & Rajesh, L. (2020). Modeling reliability of three open source software systems. In *2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)* (pp. 1-4). IEEE. Chennai, India.

Razzaq, S., Li, Y.F., Lin, C.T., & Xie, M. (2018). A study of the extraction of bug judgment and correction times from open source software bug logs. In *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)* (pp. 229-234). IEEE. Lisbon, Portugal.

Syed-Mohamad, S.M., & McBride, T. (2008). Reliability growth of open source software using defect analysis. In *2008 International Conference on Computer Science and Software Engineering* (Vol. 2, pp. 662-667). IEEE. Wuhan, China.

Tamura, Y., & Yamada, S. (2021). Performance assessment based on stochastic differential equation and effort data for edge computing. *Software Testing, Verification and Reliability*, *31*(6), e1766. https://doi.org/10.1002/stvr.1766.

Tamura, Y., & Yamada, S. (2022). Prototype of 3D reliability assessment tool based on deep learning for edge OSS computing. *Mathematics*, *10*(9), 1572. https://doi.org/10.3390/math10091572.

Tan, T., Li, Z., Liu, H., Zanjani, F.G., Ouyang, Q., Tang, Y., Hu, Z., & Li, Q. (2018). Optimize transfer learning for lung diseases in bronchoscopy using a new concept: Sequential fine-tuning. *IEEE Journal of Translational Engineering in Health and Medicine*, *6*, 1-8. https://doi.org/10.1109/JTEHM.2018.2865787.

The Apache Software Foundation, The Apache HTTP Server Project, http://httpd.apache.org/.

The Apache Software Foundation, The Apache Tomcat Project, http://tomcat.apache.org/.

Tuna, E., Kovalenko, V., & Tüzün, E. (2022). Bug tracking process smells in practice. In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice* (pp. 77-86). Association for Computing Machinery. New York. https://doi.org/10.1145/3510457.3513080.

Vrbančič, G., & Podgorelec, V. (2020). Transfer learning with adaptive fine-tuning. *IEEE Access*, *8*, 196197-196211. https://doi.org/10.1109/ACCESS.2020.3034343.

Yadav, K., Jain, E., & Bhatia, J. (2011). Comparative study of open source software reliability assessment models. In *2011 3rd International Conference on Computer Research and Development* (Vol. 1, pp. 384-388). IEEE. Shanghai, China.

Yanagisawa, T., Tamura, Y., Anand, A., & Yamada, S. (2021). Comparison of hazard-rates considering fault severity levels and imperfect debugging for OSS. *Journal of Software Engineering and Applications*, *14*(11), 591-606.