Understanding the Bug Characteristics and Fix Strategies of Federated Learning Systems

Anonymous Author(s)

1 THREATS TO VALIDITY

This study suffers from the following main threats to validity.

Selection of labels and keywords. We use labels and keywords to match bug instances during data collection on both *GitHub* and *StackOverflow*, which might contain potential bias. To mitigate this threat, we select those keywords that are widely used in data collection by existing related works [7, 8, 20, 21]. These keywords are proven to be effective for bug collection. However, automatically filtering data by labels or keywords may result in the loss of data without explicit keywords and labels. Unfortunately, such bias is hard for us as well as existing studies to mitigate.

Selection of data The credibility of data collection may lead to external threats. To mitigate this threat, we took the following measures. First, we select two typical bug sources that are widely used in empirical research: GitHub and Stack Overflow. To make our collected bugs highly relevant to FL, our identification is based on relevant frameworks [7, 20, 21]. To mitigate the bias in framework selection, we select six representative frameworks based on Forks and Stars. Meanwhile, these frameworks are widely used in academia and industry, and cover two typical scenarios of horizontal FL and vertical FL, which can ensure the extensiveness and comprehensiveness of the data we collected. Then, our automatic identification of bugs is based on keywords and labels. Finally, we use manual identification as the last step of data collection to ensure the validity of the data. To ensure the correctness of our conclusions, we only study fixed bugs, as unfixed or unconfirmed cases may not be real ones. Despite our best efforts above, our study still cannot cover all bugs because not all bugs of FL software are posted online. In the future, we plan to conduct developer interviews to validate our findings following existing work [20].

Subjectivity of manual labeling. The classification process may be biased since it involves human labeling. To mitigate this threat, firstly we classify bugs based on the well-validated taxonomy. On the other hand, we ensure that all data are independently labeled by two authors with federated learning background, and conflicting instances are resolved by other two experienced authors as arbitrators until agreement is reached. We performed multiple rounds of classification. During each round, we continuously adjust and update the categories. Multiple iterations of the categories can make the result more reliable, and the results of each iteration can be found on our online repository.

2 RELATED WORK

Empirical Study on DL Bugs. No existing work is found to characterize bugs in FL systems, so the most related work are those empirical studies focusing on DL systems. Zhang *et al.* [21] investigated the symptoms and root causes of 175 bugs in DL applications built on *TensorFlow*. They further summarized challenges and strategies for bug detection and localization. Islam *et al.* [8] investigated the impacts, types, root causes and bug-prone stages of bugs in DL softwares collected from *Github* and *StackOverflow* across five popular DL frameworks. They further studied the relationship among bugs in different DL frameworks and summarized the changing trend of two types of bugs. Humbatova *et al.* [7] studied three DL frameworks (*TensorFlow*, *Keras* and *PyTorch*) and collected 375 bugs. Different from previous works, they enriched the taxonomy by interviewing 20 researchers and practitioners in structured interviews, which help them find a variety of additional faults that did not emerge from *StackOverflow* and *GitHub*. Instead of collecting DL framework bugs, Zhang *et al.* [20] collected bugs from DL applications deployment to the DL platform *Philly* and their execution, which allows developers to submit and run DL applications in a shared manner. They collected 4960 bugs and classified them into 20 categories. Among them, 400 bugs were used to analyze the root cause. Chen *et al.* [5] studied the deployment bugs of DL models on mobile devices. They collected 304 deployment bugs from *StackOverflow* and *GitHub*, which were eventually classified into 23 categories regarding to symptoms and further summarized several fix strategies. These existing studies have summarized the symptoms [21], bug taxonomy [7, 19], bug characteristics [8, 19], root causes [1, 21], and fix patterns [9, 19]. We followed their methodology and workflow (e.g., collecting data from *StackOverflow* and *GitHub*, labeling data manually, etc.). However, our work targets at FL systems instead of ML/DL systems. Our findings show both great resemblance between the two and unique nature of FL systems.

Distributed/IoT System. Various definitions of distributed systems have been provided in previous work. Specifically, Chandy *et al.* [4] defined it as a system consists of a finite set of processes and a finite set of channels. It is described by a labeled, directed graph in which the vertices represent processes and the edges represent channels. Steen *et al.* [16] defined it as a collection of self-aware computational elements that appear to the user as a single coherent system. Naik [13] regarded it as a set of autonomous components located on different machines that communicate with each other through messages to achieve a common goal. The basic essence of these definitions is similar, that is, distributed systems are coordinated by multiple nodes to perform certain tasks. They also summarize the characteristics of distributed systems, and we compare them with the characteristics of FL systems to distill the unique characteristics of FL systems. (1) The distributed system supports resource sharing while the FL system strictly requires the data of each client to be independent. (2) Different nodes of a distributed system can coexist and work together while all clients of the FL system work independently. (3) Distributed system is fault-tolerant, that is, the failure of one node does not affect other nodes and guarantees the expected results. Since the FL system has aggregation

ESEC/FSE 2022, 14 - 18 November, 2022, Singapore
2022. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

operations, an error in a node may lead to an abnormal aggregation result. Based on the above differences, even if the two systems share similar root causes, such as node interaction, their essence is different. For example, the bugs in the interaction of shared data in distributed systems are rare, while the interaction of data in the FL system is a serious type of bug. In addition, if the interactive information is not data, such as the gradients of DL model, since FL system requires strict encryption, their fix strategies are also different.

The basic concept of Internet of Things (IoT) is the detection of a wide variety of objects so that users can query and manipulate them directly on the Internet, or through programs that encapsulate their behavior and goals [18]. Makhshari *et al.* [12] categorized 323 IoT bugs based on the observed failures, root causes, and the locations of the faulty components from 91 representative IoT project repositories. From this study, IoT system enables communication between devices through cloud servers, where each device is connected to the Internet like a regular network device. The FL system needs to initialize the server/client first, where the wrong order of initialization can lead to bugs, which makes FL and IOT bugs different in terms of fixing strategies for configuration bugs. Also in the node interaction, IoT messages are sent to IoT devices through the cloud, and the rate and order error of these messages will lead to bugs, which is also different from the bug of FL system.

Attacks in FL Systems. In ML-based systems, poisoning attacks aim to undermine the integrity of training process, which can be divided into random poisoning attacks [3, 10, 17] and targeted poisoning attacks [2, 11, 15]. There are two mainstream poisoning methods, data poisoning [6, 14] during training dataset collection and model poisoning [2] impairing learning process. Another type of attack methods are inference attacks. Model updates contain private information related to local data, while adversaries can conduct inferences from such intermediate results. For instance, Zhu *et al.* [22] proposed an algorithm which can infer training inputs and labels from gradients. Our work focuses on the quality of FL systems, and provides actionable guidelines for bug detection and localization, including bugs introducing high security risks.

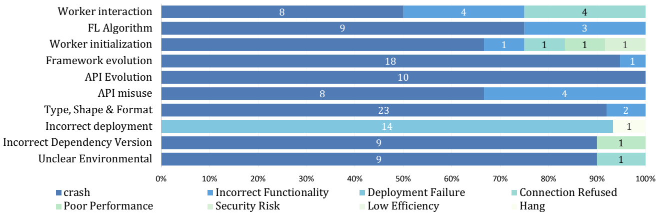

Figure 1: Bug Distribution by Root Causes for Each Symptom.

3 DISCUSSION

Correlation between Root Causes and Symptoms. Figure 1 presents the number of the top 10 root causes for bugs with each symptom. Except for *Incorrect Deployment*, *Crash* is the most common symptom for all these root causes. We also find that all bugs exhibiting the symptom of *Incorrect Deployment* are produced by *Incorrect Deployment*. On the other hand, 83.33% of bugs exhibiting symptom *Connection Refused* are produced by *Interaction Issues*, which indicates that developers can be suggested to check the implementation of interaction between different parties when a *Connection Refused* bug occurs.

REFERENCES

[1] 2020. Taxonomy of Real Faults in Deep Learning Systems. Retrieved March 10, 2022 from https://github.com/dlfaults/dl_faults

[2] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2938–2948.

[3] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*. 1467–1474.

[4] K Mani Chandy and Leslie Lamport. 1985. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems (TOCS)* 3, 1 (1985), 63–75.

[5] Zhenpeng Chen, Huihan Yao, Yiling Lou, Yanbin Cao, Yuanqiang Liu, Haoyu Wang, and Xuanzhe Liu. 2021. An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 674–685. https://doi.org/10.1109/ICSE43902.2021.00068

[6] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).

[7] Nargiz Humbatova, Gunel Jahangirova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, and Paolo Tonella. 2020. Taxonomy of Real Faults in Deep Learning Systems. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (Seoul, South Korea) *(ICSE '20)*. Association for Computing Machinery, New York, NY, USA, 1110–1121. https://doi.org/10.1145/3377811.3380395

[8] Md Johirul Islam, Giang Nguyen, Rangeet Pan, and Hridesh Rajan. 2019. A Comprehensive Study on Deep Learning Bug Characteristics. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Tallinn, Estonia) *(ESEC/FSE 2019)*. Association for Computing Machinery, New York, NY, USA, 510–520. https://doi.org/10.1145/3338906.3338955

[9] Md Johirul Islam, Rangeet Pan, Giang Nguyen, and Hridesh Rajan. 2020. Repairing deep neural networks: Fix patterns and challenges. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 1135–1146.

[10] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. In *2018 IEEE Symposium on Security and Privacy (SP)*. 19–35. https://doi.org/10.1109/SP.2018.00057

[11] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2017. Trojaning attack on neural networks. (2017).

[12] Amir Makhshari and Ali Mesbah. 2021. IoT Bugs and Development Challenges. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 460–472. https://doi.org/10.1109/ICSE43902.2021.00051

[13] Nitin Naik. 2021. Demystifying properties of distributed systems. In *2021 IEEE International Symposium on Systems Engineering (ISSE)*. IEEE, 1–8.

[14] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems* 31 (2018).

[15] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. 2018. When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks. In *27th USENIX Security Symposium (USENIX Security 18)*. 1299–1316.

[16] Maarten van Steen and Andrew S Tanenbaum. 2016. A brief introduction to distributed systems. *Computing* 98, 10 (2016), 967–1009.

[17] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. 2015. Is feature selection secure against training data poisoning?. In *international conference on machine learning*. PMLR, 1689–1698.

[18] Teng Xu, James B. Wendt, and Miodrag Potkonjak. 2014. Security of IoT systems: Design challenges and opportunities. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 417–423. https://doi.org/10.1109/ICCAD.2014.7001385

[19] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. 2020. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering* (2020).

[20] Ru Zhang, Wencong Xiao, Hongyu Zhang, Yu Liu, Haoxiang Lin, and Mao Yang. 2020. An Empirical Study on Program Failures of Deep Learning Jobs. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. 1159–1170. https://doi.org/10.1145/3377811.3380362

[21] Yuhao Zhang, Yifan Chen, Shing-Chi Cheung, Yingfei Xiong, and Lu Zhang. 2018. An Empirical Study on TensorFlow Program Bugs. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis* (Amsterdam, Netherlands) *(ISSTA 2018)*. Association for Computing Machinery, New York, NY, USA, 129–140. https://doi.org/10.1145/3213846.3213866

[22] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in Neural Information Processing Systems* 32 (2019).