RESEARCH STATEMENT

December 18, 2024
School of Computing and Information Systems, Singapore Management University
Tel: (65) 6826-4861; Email: xfxie@smu.edu.sg

1 Summary

Today, software is ubiquitous, powering everything from mobile devices and web applications to advanced computer systems. The rise of intelligent systems such as healthcare, self-driving cars, and robotics has made us increasingly reliant on the reliable and secure functioning of software. As these systems continue to grow in size and complexity, ensuring their correctness and security has become a significant challenge. This challenge is further amplified by the evolving nature of software. Traditional code-based software is now complemented by AI-driven systems, such as large language models, which bring unprecedented capabilities but also introduce new dimensions of complexity and unpredictability. Given the increasing complexity and heterogeneity of modern software systems, the fundamental methodology for quality assurance and trustworthiness of software systems is necessary.

Motivated by these challenges, my research aims to develop automated techniques and practical tools to enhance the quality, reliability, and security of modern software systems, encompassing both traditional and AI-driven software. To date, I have established foundational methods for software analysis and testing, which have been successfully applied across diverse software systems, including open-source projects, autonomous driving platforms, game software, web applications, and multi-agent systems.

Research Overview Software systems are often vulnerable to manipulation by malicious users due to inherent quality and security issues, which can lead to severe consequences, particularly in environments where safety and security are critical. To address these challenges, I am dedicated to building trustworthy and secure systems. *My research philosophy is to identify and tackle problems that are both **fundamental** and **practical**, ensuring that my research work has significant impact.* Figure 1 provides an overview of my research roadmap, highlighting my contributions to quality assurance of diverse software systems.

For traditional software, my primary focus is on the fundamental research of program correctness:

- Fundamental Program Analysis [47, 42, 44]
- Program Termination Analysis [43, 53, 33].
- General-Purpose Software Testing [35, 36, 22, 1]
- Domain-Specific Software Testing [58, 40, 49, 24, 39, 50, 14, 27, 57, 51, 55, 56, 9, 32, 12, 54]
- Data-Driven Methods for Program Analysis [20, 25, 11, 28, 29, 37, 10]

For intelligent software, my research targets quality assurance across the entire machine learning lifecycle:

- Data Analysis and Selection [52, 18, 16, 17, 13, 19]
- Fundamental Abstraction and Model-based Analysis for Blackbox Deep Neural Networks [6, 5, 7, 26]
- General-Purpose Model Testing [48, 6, 59, 38, 46]
- Domain-Specific Intelligent System Testing [8, 3, 23, 30, 2, 15, 41]
- Model Repair [45, 34, 21]

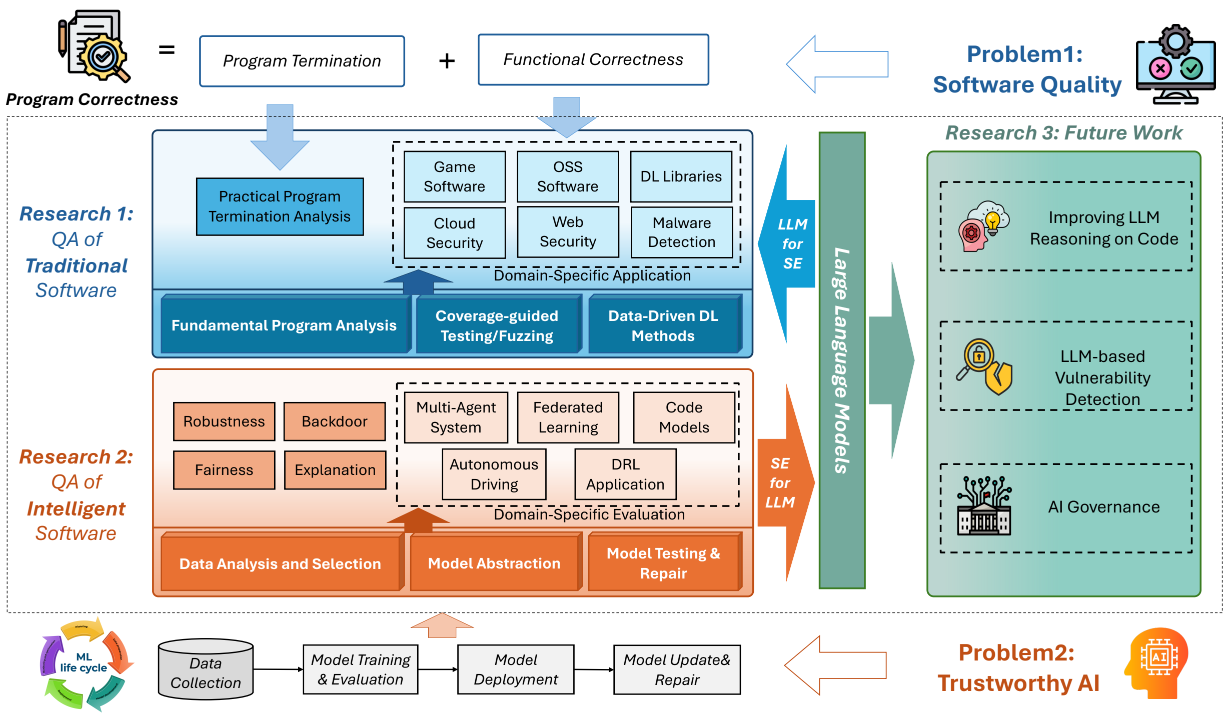

Figure 1: Overview of My Research

In what follows, I list some of my most representative lines of works.

2 Quality Assurance of Traditional Software

Program correctness is a critical factor that directly influences the quality, reliability, and security of software systems. In theoretical computer science, establishing a program's *total correctness* requires proving both its *functional correctness* and its *termination*[1]. To address these challenges, I have developed theoretical methodologies in the areas of static program analysis, software testing, and AI-driven techniques.

2.1 Theoretical Loop Analysis and Practical Termination Analysis

As part of my earlier research, I focused on foundational loop analysis, which is one of the most challenging problems in program analysis. My primary contribution has been in static loop summarization, which aims to compute the relationship between loop inputs and outputs without executing the loops. Given the undecidability of this problem, I first conducted comprehensive study [42] to systematically assess the complexity of various loop types. Then I developed a series of specialized summarization techniques tailored for different types of loops, including: 1) summarization for string traversal loops [47] and 2) summarization for inductive integer loops [42] and 3) extensions to non-inductive integer loops and nested loops [44]. These techniques have been instrumental in enhancing the performance of symbolic execution, vulnerability detection, and program verification. This *fundamental* research was recognized with the ***ACM SIGSOFT Distinguished Paper Award*** (FSE 2016) for the integer loop summarization work [42].

[1] https://en.wikipedia.org/wiki/Correctness_(computer_science)

Termination analysis, a crucial aspect of program correctness, is a classical and challenging problem (i.e., *the halting problem*). Despite extensive research over the years, existing methods have primarily focused on theoretical aspects, often lacking practical applicability. Consequently, these techniques are typically effective only on simplified toy examples and struggle to handle real-world programs. My research addresses this gap by developing *practical* approaches for termination analysis. I began by developing a lightweight static loop termination analysis method [43], which significantly improves the performance of existing tools by over 20 times. To move one step further, my most recent work further tackles the complexities of real-world software. We conducted an in-depth study to characterize real-world non-termination bugs [33], gaining insights into the root causes of infinite loops and recursions, as well as identifying the limitations of state-of-the-art tools. Based on these insights, we developed the *first* practical non-termination detection method [53] capable of identifying infinite loops and recursion issues in real-world applications. The key innovation is the development of two novel non-termination oracles, which are integrated into dynamic fuzzing, allowing for the effective detection of non-termination bugs in real-world software systems. This work has been recognized for its *practical* impact, wining the ***ACM SIGSOFT Distinguished Paper Award*** (ASE 2023) and successfully identifying 8 previously unknown non-termination bugs in OSS projects.

2.2 Software Testing for Bug Detection

To ensure functional correctness, I have focused on developing testing methods for detecting bugs that violate specifications, including both general-purpose fuzz testing and domain-specific testing.

General-Purpose Fuzz Testing. In my earlier works, I developed automated fuzzing techniques [35, 1, 22, 36] for bug detection and analysis. My primary focus has been on enhancing the testing optimization through effective guidance [1, 35] and seed prioritization [22]. A representative project is the development of a technique [35] for detecting use-after-free (UAF) vulnerabilities. In this approach, we modeled UAF as an automaton, which is then used to guide test generation, effectively covering the abnormal states within the automaton. Beyond detecting vulnerabilities, I also developed a root cause analysis technique [36] to facilitate more efficient debugging and root cause localization. These efforts have led to the identification of over 50 critical security vulnerabilities in open-source projects. For instance, the PHP project recognized my contributions with a USD 1,500 bug bounty award.

Domain-Specific Software Testing. Due to the diverse nature of software applications, such as mobile apps, web applications and cloud platforms, general-purpose testing methods often fall short in addressing domain-specific challenges. My recent work has focused on tackling these challenges and developing specialized testing algorithms to ensure quality and security in various domains. Key contributions include:

- *Game Software*. Ensuring the quality of game software is crucial, as bugs can significantly impact user experience (with over 3.3 billion players in 2023), result in financial losses, and even pose security risks. However, existing game software testing largely relies on manual efforts and simple scripts, which are both costly and inefficient. To address this, we developed the first scalable and effective technique for testing large-scale game software [58]. Our method combines evolutionary algorithms and deep reinforcement learning to generate diverse and intelligent policies for playing games, thereby increasing test coverage. Additionally, we addressed the challenge of game regression testing, where frequent software updates (e.g., up to three versions per day in NetEase) significantly increase testing requirements. We proposed a differential testing technique [40] and regression test selection method [50] to identify regression bugs. Specifically, GameRTS [50] is the first work on game regression test selection, modeling the game as a transition state graph to select regression tests by identifying changed states and actions. We also explored methods for testing mobile game software [49, 39] by detecting widget-related issues. To further advance game testing research, we released the first comprehensive game bug dataset [24] derived from real-world games. The method [58] received the ***ACM SIGSOFT Distinguished Paper Award*** (ASE 2019) for its pioneering approach to testing real-world games.

- *Cloud Security*. As many services are now deployed in the cloud, ensuring their quality and security is critical. To address this, I have contributed methods from various perspectives, including bug detection in cloud infrastructure and network security. We first proposed a fuzzing method [31] to identify vulnerabilities in SSL/TLS implementations by a syntax-aware certificate mutations. We systematically investigated bugs in container runtime systems [51], providing valuable insights for developing new detection methods. To detect cloud-based attacks not related to software vulnerabilities, we further developed intrusion detection systems [56, 55] and malware detection methods [9], which are designed to identify attacks stemming from malicious behaviors. Our methods have successfully identified over 16 real-world bugs in SSL/TLS implementations and container systems, contributing to the security and stability of cloud services.
- *Web Security*. Web applications are among the most widely used software types. I have developed several testing methods to enhance web security. We proposed an effective method [57] that combines automata-guided exploitation with curiosity-driven exploration to effectively test web applications. To address the oracle problem in detecting logic bugs, we developed a method [27] that leverages large language models (LLMs) to infer invariants within the context of web applications, allowing the detection of complex logic errors. We also introduced a fuzzing method [14] for identifying vulnerabilities in JavaScript engines, a critical component in web browsers. By designing reflection-based mutation, it generates high-quality test cases for testing. These tools have resulted in the identification of over 12 previously unknown bugs in web applications and 51 vulnerabilities in popular JavaScript engines.
- *Deep Learning Libraries*. Deep learning libraries, such as TensorFlow and PyTorch, are fundamental to the development of AI applications. To ensure their quality and security, I have conducted extensive research into bugs present in these libraries. For example, we investigated bugs in TensorFlow.js [32] and developed detection methods [12] that have successfully identified over 100 bugs across TensorFlow and PyTorch. In our most recent work, we have discovered very novel code injection attacks [60] through abusing TensorFlow APIs, posing significant security risks to large language models.

2.3 AI-based Methods for Software Analysis

While traditional static analysis and dynamic testing methods have proven effective, they still face significant challenges when applied to large-scale software systems. With the advent of AI, data-driven approaches are increasingly being used in software engineering tasks. My contributions in this area include AI-based methods such as code search [29], code summarization [28], code review [10], type inference [25], and code completion [11]. One representative work addresses the cross-lingual problem by proposing a transfer learning-based method that transfers knowledge from one programming language to another. This innovative approach won the ***ACM SIGSOFT Distinguished Paper Award*** (ISSTA 2022). Another key contribution is the development of a novel retrieval-augmented generation framework for code completion [11]. This work tackles the critical problem of determining *what* information to retrieve and *how* to augment model outputs based on retrieved data, derived from a *theoretical analysis* of the fine-tuning process. Our method achieved a over 2× increase in Exact Match performance compared to the state-of-the-art methods.

3 Quality Assurance of Intelligent Software

Machine learning (ML) has been widely applied in many applications. However, ML models (e.g., DNNs) remain vulnerable to various attacks. My research focuses on the quality assurance of ML software throughout the entire lifecycle, including data collection, model training, deployment, and updates.

3.1 Data Analysis and Selection

Data collection and labeling are crucial steps in the machine learning pipeline, as the quality of the data significantly impacts model performance. However, most real-world data is unlabeled, and manual labeling is both time-consuming and costly. My research focuses on selecting a minimal subset of data for labeling, optimizing it for tasks such as accuracy estimation [13, 19], data distribution analysis [4], robustness evaluation [18], and robustness enhancement [18, 17]. One of my representative works proposes an unsupervised method to estimate the accuracy of an unlabeled dataset based solely on the original test data. This unsupervised method utilizes the relationship between the distance from the data to the decision boundary and the model's capability for these data to perform accuracy estimation on new data. Additionally, we analyzed the characteristics of benign and adversarial samples based on the uncertainty. A method has been developed to generate robust adversarial examples [52] that are difficult to detect.

3.2 Model-based Analysis for Deep Neural Networks

Compared to traditional software, the primary challenge in analyzing DNNs is their black-box nature. To address this, I developed fundamental state abstraction and model-based analysis methods that aim to extract an abstract model from black-box DNN. This enables us to improve further analysis such as testing, explanation, attack, and repair. Specifically, we propose the methods [6, 45, 7, 5] that model a Recurrent Neural Network (RNN) as an abstract state transition system, effectively characterizing its internal behaviors. Using this abstract model, we have conducted further analysis, including: Effective techniques to test and defense defects in RNNs [6]; A method for repairing RNNs to improve performance and reliability [45]; A framework for analyzing and enhancing the robustness of RNNs [5]; Techniques to detect backdoor attacks in DNNs [7]. Furthermore, we extend model-based analysis to enhance deep reinforcement learning [26] by calculating more accurate rewards based on abstract state measurements.

3.3 Testing and Repair of Intelligent Systems

General-Purpose Model Testing To address the unique challenges of testing deep learning models, I have contributed to the development of novel testing methods specifically designed for DNNs. For the new software, I designed new testing coverage criteria [46, 6] that analyze the decision logic of neural networks, including neuron activations and state transitions, to effectively measure test adequacy. In addition, I developed several general-purpose testing methods [59, 48, 6, 38] to detect defects in DNNs. A notable example is DeepHunter [48], the first general fuzzing framework for DNNs. Our recent work, DistXplore [38], reconsiders existing issues in DNN testing and redefines testing objectives to focus on generating robust defects and improving model robustness. We further proposed a distribution-guided approach to effectively test and enhance model robustness. By comparing DistXplore to 14 state-of-the-art methods, we demonstrated its effectiveness and set a new direction for future DNN testing research.

Domain-Specific Testing of Intelligent Software Given the diverse applications of DNNs, we have developed specialized testing methods for various domains, such as autonomous driving systems (ADS) [3, 23], multi-agent systems (MAS) [30, 2], federated learning enhancement [15, 41]. For ADS testing, we introduced the first diversity-driven testing method by designing an abstraction-based approach to measure vehicle driving diversity and generate failure scenarios with high coverage. While existing methods primarily focus on maximizing the number of failures, we emphasize the need to improve the diversity of failure cases to ensure comprehensive testing. For MAS testing, we developed MASTest [30], a novel framework that innovatively considers both individual agent behaviors and group dynamics to characterize MAS behavior diversity, providing a more holistic evaluation of multi-agent systems. In addition, we have developed repair methods to enhance model performance in terms of robustness and fairness [45, 21, 34].

4 Future Work

My future research will primarily focus on leveraging LLMs to advance program analysis and software testing. LLMs have shown significant potential in various software engineering tasks, but their current limitations in understanding code semantics need to be addressed. 1) Given the current limitations of LLMs in semantic code understanding, I aim to enhance their reasoning capabilities, such as through fine-tuning and RAG. This will involve incorporating more reasoning-relevant information during the training phase, enabling LLMs to better comprehend and analyze complex code structures. 2) Building on these enhanced reasoning capabilities, I aim to develop the next generation of LLM-driven vulnerability detection frameworks. This approach will integrate LLMs with traditional program analysis, testing, and fuzzing techniques to boost the detection of vulnerabilities. 3) On the other hand, as LLMs become increasingly integrated into software systems, their ethical and security implications must be carefully managed. I will focus on AI governance, addressing the gap between government regulations and existing evaluation methods. My goal is to bridge this gap by creating comprehensive frameworks and tools that align AI development and deployment with governance standards, ensuring safe, secure, and ethical AI applications.

References

[1] H. Chen, Y. Xue, Y. Li, B. Chen, X. Xie, X. Wu, and Y. Liu. Hawkeye: Towards a desired directed grey-box fuzzer. In *CCS*, pages 2095–2108. ACM, 2018.

[2] J. Chen, Y. Wang, J. Wang, X. Xie, D. Wang, Q. Wang, and F. Xu. Demo2test: Transfer testing of agent in competitive environment with failure demonstrations. *ACM Transactions on Software Engineering and Methodology*, 2024.

[3] M. Cheng, Y. Zhou, and X. Xie. Behavexplor: Behavior diversity guided testing for autonomous driving systems. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 488–500, 2023.

[4] B. David, X. Xie, L. Ma, L. Zhou, Y. Liu, C. Xu, and J. Zhao. Cats are not fish: Deep learning testing calls for out-of-distribution awareness. In *ASE*, 2020.

[5] X. Du, Y. Li, X. Xie, L. Ma, Y. Liu, and J. Zhao. Marble: Model-based robustness analysis of stateful deep learning systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 423–435, 2020.

[6] X. Du, X. Xie, Y. Li, L. Ma, Y. Liu, and J. Zhao. Deepstellar: model-based quantitative analysis of stateful deep learning systems. In *ESEC/FSE*, pages 477–487. ACM, 2019.

[7] M. Fan, Z. Si, X. Xie, Y. Liu, and T. Liu. Text backdoor detection using an interpretable rnn abstract model. *IEEE Transactions on Information Forensics and Security*, 16:4117–4132, 2021.

[8] M. Fan, W. Wei, X. Xie, Y. Liu, X. Guan, and T. Liu. Can we trust your explanations? sanity checks for interpreters in android malware analysis. *IEEE Transactions on Information Forensics and Security*, 16:838–853, 2020.

[9] R. Feng, S. Chen, X. Xie, G. Meng, S.-W. Lin, and Y. Liu. A performance-sensitive malware detection system using deep learning on mobile devices. *IEEE Transactions on Information Forensics and Security*, 16:1563–1578, 2020.

[10] Q. Guo, J. Cao, X. Xie, S. Liu, X. Li, B. Chen, and X. Peng. Exploring the potential of chatgpt in automated code refinement: An empirical study. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–13, 2024.

[11] Q. Guo, X. Li, X. Xie, S. Liu, Z. Tang, R. Feng, J. Wang, J. Ge, and L. Bu. Ft2ra: A fine-tuning-inspired approach to retrieval-augmented code completion. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 313–324, 2024.

[12] Q. Guo, X. Xie, Y. Li, X. Zhang, Y. Liu, X. Li, and C. Shen. Audee: Automated testing for deep learning frameworks. In *Proceedings of the 35th IEEE/ACM international conference on automated software engineering*, pages 486–498, 2020.

[13] Y. Guo, Q. Hu, X. Xie, M. Cordy, M. Papadakis, and Y. Le Traon. Kape: k nn-based performance testing for deep code search. *ACM Transactions on Software Engineering and Methodology*, 33(2):1–24, 2023.

[14] X. He, X. Xie, Y. Li, J. Sun, F. Li, W. Zou, Y. Liu, L. Yu, J. Zhou, and W. Shi. Sofi: Reflection-augmented fuzzing for javascript engines. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2229–2242, 2021.

[15] M. Hu, Z. Yue, X. Xie, C. Chen, Y. Huang, X. Wei, X. Lian, Y. Liu, and M. Chen. Is aggregation the only choice? federated learning via layer-wise model recombination. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1096–1107, 2024.

[16] Q. Hu, Y. Guo, X. Xie, M. Cordy, L. Ma, M. Papadakis, and Y. Le Traon. Active code learning: Benchmarking sample-efficient training of code models. *IEEE Transactions on Software Engineering*, 2024.

[17] Q. Hu, Y. Guo, X. Xie, M. Cordy, L. Ma, M. Papadakis, and Y. Le Traon. Test optimization in dnn testing: a survey. *ACM Transactions on Software Engineering and Methodology*, 33(4):1–42, 2024.

[18] Q. Hu, Y. Guo, X. Xie, M. Cordy, M. Papadakis, and Y. Le Traon. Laf: Labeling-free model selection for automated deep neural network reusing. *ACM Transactions on Software Engineering and Methodology*, 33(1):1–28, 2023.

[19] Q. Hu, Y. Guo, X. Xie, M. Cordy, M. Papadakis, L. Ma, and Y. Le Traon. Aries: Efficient testing of deep neural networks via labeling-free accuracy estimation. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1776–1787. IEEE, 2023.

[20] S. Li, X. Xie, Y. Lin, Y. Li, R. Feng, X. Li, W. Ge, and J. S. Dong. Deep learning for coverage-guided fuzzing: How far are we? *IEEE Transactions on Dependable and Secure Computing*, 2022.

[21] T. Li, X. Xie, J. Wang, Q. Guo, A. Liu, L. Ma, and Y. Liu. Faire: Repairing fairness of neural networks via neuron condition synthesis. *ACM Transactions on Software Engineering and Methodology*, 33(1):1–24, 2023.

[22] Y. Li, Y. Xue, H. Chen, X. Wu, C. Zhang, X. Xie, H. Wang, and Y. Liu. Cerebro: context-aware adaptive fuzzing for effective vulnerability detection. In *ESEC/FSE*, pages 533–544. ACM, 2019.

[23] Z. Li, X. Wu, D. Zhu, M. Cheng, S. Chen, F. Zhang, X. Xie, L. Ma, and J. Zhao. Generative model-based testing on decision-making policies. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 243–254. IEEE, 2023.

[24] Z. Li, Y. Wu, L. Ma, X. Xie, Y. Chen, and C. Fan. Gbgallery: A benchmark and framework for game testing. *Empirical Software Engineering*, 27(6), 2022.

[25] Z. Li, X. Xie, H. Li, Z. Xu, Y. Li, and Y. Liu. Retracted on march 14, 2023: Cross-lingual transfer learning for statistical type inference. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2022, page 239–250, New York, NY, USA, 2022. Association for Computing Machinery.

[26] Z. LI, D. ZHU, Y. HU, X. XIE, L. MA, Y. ZHENG, Y. SONG, Y. CHEN, and J. ZHAO. Neural episodic control with state abstraction. *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[27] Y. Liao, M. Xu, Y. Lin, X. Teoh, X. Xie, R. Feng, F. Liauw, H. Zhang, and J. S. Dong. Detecting and explaining anomalies caused by web tamper attacks via building consistency-based normality. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 2024.

[28] S. LIU, Y. CHEN, X. XIE, J. SIOW, and Y. LIU. Retrieval-augmented generation for code summarization via hybrid gnn. *Proceedings of the Ninth International Conference on Learning Representations*, pages 1–16, 2021.

[29] S. Liu, X. Xie, J. Siow, L. Ma, G. Meng, and Y. Liu. Graphsearchnet: Enhancing gnns via capturing global dependencies for semantic code search. *IEEE Transactions on Software Engineering*, 49(4):2839–2855, 2023.

[30] X. Ma, Y. Wang, J. Wang, X. Xie, B. Wu, S. Li, F. Xu, and Q. Wang. Enhancing multi-agent system testing with diversity-guided exploration and adaptive critical state exploitation. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1491–1503, 2024.

[31] L. Quan, Q. Guo, H. Chen, X. Xie, X. Li, Y. Liu, and J. Hu. Sadt: syntax-aware differential testing of certificate validation in ssl/tls implementations. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 524–535, 2020.

[32] L. Quan, Q. Guo, X. Xie, S. Chen, X. Li, and Y. Liu. Towards understanding the faults of javascript-based deep learning systems. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–13, 2022.

[33] X. Shi, X. Xie, Y. Li, Y. Zhang, S. Chen, and X. Li. Large-scale analysis of non-termination bugs in real-world oss projects. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 256–268, 2022.

[34] C. X. Tian, H. Li, X. Xie, Y. Liu, and S. Wang. Neuron coverage-guided domain generalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1302–1311, 2022.

[35] H. Wang, X. Xie, Y. Li, C. Wen, Y. Li, Y. Liu, S. Qin, H. Chen, and Y. Sui. Typestate-guided fuzzer for discovering use-after-free vulnerabilities. In *ICSE, accepted*. ACM, 2020.

[36] H. Wang, X. Xie, S.-W. Lin, Y. Lin, Y. Li, S. Qin, Y. Liu, and T. Liu. Locating vulnerabilities in binaries via memory layout recovering. In *ESEC/FSE*, pages 718–728, New York, NY, USA, 2019. ACM.

[37] J. Wang, S. Liu, X. Xie, S. Kai, K. Liu, and Y. Li. Ratchet: Retrieval augmentedtransformer for program repair. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–13, 2024.

[38] L. Wang, X. Xie, X. Du, M. Tian, Q. Guo, Z. Yang, and C. Shen. Distxplore: Distribution-guided testing for evaluating and enhancing deep learning systems. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 68–80, 2023.

[39] X. Wu, J. Ye, K. Chen, X. Xie, Y. Hu, R. Huang, L. Ma, and J. Zhao. Widget detection-based testing for industrial mobile games. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 173–184. IEEE, 2023.

[40] Y. Wu, Y. Chen, X. Xie, B. Yu, C. Fan, and L. Ma. Regression testing of massively multiplayer online role-playing games. In *2020 IEEE international conference on software maintenance and evolution (ICSME)*, pages 692–696. IEEE, 2020.

[41] Z. Xia, M. Hu, D. Yan, X. Xie, T. Li, A. Li, J. Zhou, and M. Chen. Cabafl: Asynchronous federated learning via hierarchical cache and feature balance. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[42] X. Xie, B. Chen, Y. Liu, W. Le, and X. Li. Proteus: computing disjunctive loop summary via path dependency analysis. In *FSE*, pages 61–72. ACM, 2016.

[43] X. Xie, B. Chen, L. Zou, S.-W. Lin, Y. Liu, and X. Li. Loopster: Static loop termination analysis. In *ESEC/FSE*, pages 84–94, New York, NY, USA, 2017. ACM.

[44] X. Xie, B. Chen, L. Zou, Y. Liu, W. Le, and X. Li. Automatic loop summarization via path dependency analysis. *IEEE Transactions on Software Engineering*, 45(6):537–557, June 2019.

[45] X. Xie, W. Guo, L. Ma, W. Le, J. Wang, L. Zhou, Y. Liu, and X. Xing. Rnnrepair: Automatic rnn repair via model-based analysis. In *International Conference on Machine Learning*, pages 11383–11392. PMLR, 2021.

[46] X. Xie, T. Li, J. Wang, L. Ma, Q. Guo, F. Juefei-Xu, and Y. Liu. Npc: N euron p ath c overage via characterizing decision logic of deep neural networks. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(3):1–27, 2022.

[47] X. Xie, Y. Liu, W. Le, X. Li, and H. Chen. S-looper: Automatic summarization for multipath string loops. In *ISSTA*, pages 188–198. ACM, 2015.

[48] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In *ISSTA*, pages 146–157. ACM, 2019.

[49] J. Ye, K. Chen, X. Xie, L. Ma, R. Huang, Y. Chen, Y. Xue, and J. Zhao. An empirical study of gui widget detection for industrial mobile games. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1427–1437, 2021.

[50] J. Yu, Y. Wu, X. Xie, W. Le, L. Ma, Y. Chen, J. Hu, and F. Zhang. Gamerts: A regression testing framework for video games. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1393–1404. IEEE, 2023.

[51] J. Yu, X. Xie, C. Zhang, S. Chen, Y. Li, and W. Shen. Bugs in pods: Understanding bugs in container runtime systems. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1364–1376, 2024.

[52] X. Zhang, X. Xie, L. Ma, X. Du, Q. Hu, Y. Liu, J. Zhao, and M. Sun. Towards characterizing adversarial defects of deep learning software from the lens of uncertainty. In *ICSE, accepted*. ACM, 2020.

[53] Y. Zhang, X. Xie, Y. Li, S. Chen, C. Zhang, and X. Li. Endwatch: A practical method for detecting non-termination in real-world software. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 686–697. IEEE, 2023.

[54] Y. Zhang, X. Xie, Y. Li, Y. Lin, S. Chen, Y. Liu, and X. Li. Demystifying performance regressions in string solvers. *IEEE Transactions on Software Engineering*, 49(3):947–961, 2022.

[55] Z. Zhao, Z. Li, X. Xie, J. Yu, F. Zhang, R. Zhang, B. Chen, X. Luo, M. Hu, and W. Ma. Towards fine-grained unknown class detection against the open-set attack spectrum with variable legitimate traffic. *IEEE/ACM Transactions on Networking*, 2024.

[56] Z. Zhao, Z. Li, J. Yu, F. Zhang, X. Xie, H. Xu, and B. Chen. Cmd: co-analyzed iot malware detection and forensics via network and hardware domains. *IEEE Transactions on Mobile Computing*, 2023.

[57] Y. Zheng, Y. Liu, X. Xie, Y. Liu, L. Ma, J. Hao, and Y. Liu. Automatic web testing using curiosity-driven reinforcement learning. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 423–435. IEEE, 2021.

[58] Y. Zheng, X. Xie, T. Su, L. Ma, J. Hao, Z. Meng, Y. Liu, R. Shen, Y. Chen, and C. Fan. Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning. In *ASE*. ACM, 2019.

[59] Y. Zhi, X. Xie, C. Shen, J. Sun, X. Zhang, and X. Guan. Seed selection for testing deep neural networks. *ACM Transactions on Software Engineering and Methodology*, 33(1):1–33, 2023.

[60] R. Zhu, G. Chen, W. Shen, X. Xie, and R. Chang. My model is malware to you: Transforming ai models into malware by abusing tensorflow apis. In *Proceedings of the 46th IEEE Symposium on Security and Privacy*, 2025.