Huanting Wang — Curriculum Vitae

Email: schwa@leeds.ac.uk Website: https://huantwang.github.io/

Education

10/2021 – Present	University of Leeds, United Kingdom	*Ph.D. in computer science* • **1 Publication (3 under review)** • **School of Computing Full Scholarship (2 places)**
09/2018 – 07/2021	Northwest University, China (Tier 1A)	*MSc in Software Engineering* (Core Major Cause Average: 91.14 / 100) • **5 Publications + 5 patent** • **3x First Class Scholarships (top 5% students), during 2018, 2019 and 2020**
09/2014 – 07/2018	Chang'an University, China (Tier 1A)	*BSc in Software Engineering* (Core Major Cause Average 90.33 / 100)

Experience

08/2021 – 11/2021	Alibaba DAMO Academic (Interns)	*Research And Development Engineer in NLP group*
07/2019 – 12/2019	Alipay Information & Technology Co., Ltd. (Part time)	*Development Engineer in Security group*

Publications

[1] *Automating reinforcement learning architecture design for code optimization,*

H. Wang, Z. Tang, C. Zhang, J. Zhao, C. Cummins, H. Leather, Z. Wang,

Proceedings of the 31st ACM SIGPLAN International Conference on Compiler Construction (CC), 2022.

*Premier ACM conference in parallel computing (**CORE A**)*

[2] *Combining Graph-based Learning with Automated Data Collection for Code Vulnerability Detection,*

H. Wang, G. Ye, Z. Tang, S.H. Tan, S. Huang, D. Fang, Y. Feng, L. Bian, Z. Wang**,**

IEEE Transactions on Information Forensics and Security, 2020.

Flagship journal in computer security (CORE A*), ranking #2 of the top publication list in the "Computer Security & Cryptography" category according to Google Scholar's metric.

This work was featured on several online media.

[3] *Deep Program Structure Modeling Through Multi-Relational Graph-based Learning*

G. Ye, Z. Tang, **H. Wang**, J. Fang, S. Huang, Z. Wang

The 29th ACM International Conference on Parallel Architectures and Compilation Techniques (PACT), 2020.

Lead student author (first two authors are supervisors)

Premier ACM conference in parallel computing (CORE A)

[4] *Detecting Code Vulnerabilities by Learning from Large-Scale Open Source Repositories*

R. Xu, Z. Tang, G. Ye, **H. Wang**, X. Ke, D. Fang, L. Bian, Z. Wang

Journal of Information Security and Applications (JISA), 2022. *(Impact factor: 3.579)*

[5] *Invalidating Analysis Knowledge for Code Virtualization Protection Through Partition Diversity*

Wei W, M Li, Z Tang, **H Wang**, G Ye, F Wang, J Ren, X. Gong, D. Fang, Z. Wang

IEEE Access 7, 169160-169173, 2019. *(Impact factor: 3.745)*

[6] *Leveraging WebAssembly for Numerical JavaScript Code Virtualization*

S. Wang, G. Ye, M. Li, L. Yuan, Z. Tang, **H. Wang**, W. Wang, F. Wang, J. Ren, Z. Wang

IEEE Access 7, 169160-169173, 2019. *(Impact factor: 3.745)*

Patent

Handwritten letter recognition method based on short-time Fourier transform of high-frequency sound waves,

Z. Tang, Q. Li, S. Zhai, T. Fan, **H. Wang**, D. Fang, X. Gong, F. Chen.

Patent Issued. Patent number: CN109657739A.

A source code vulnerability detection method based on graph convolution network,

W. Kong, G. Ye, **H.Wang**, Z. Tang, D. Fang

Patent Issued. Patent number: CN111460450A

A fuzzy test method for JS engine based on standard document analysis,

G. Ye, W. Yi, **H. Wang**, Y. Tian, X. Qu, Y. Wang, Z. Tang, D. Fang

Under Review. Patent number: P1201778.

A vulnerability detection model based on a multi-graph network,

G. Ye, **H. Wang**, Z. Tang, D. Fang

Under Review. Patent number: P1201913.

A high-quality source code vulnerability data collection method based on integrated learning and conformal techniques,

G. Ye, **H. Wang**, Z. Tang, D. Fang

Under Review. Patent number: P1201914.

Awards

2021 to 2024 School of Computing **Full Scholarship** (2 places)

2018 to 2020 **First Class** Scholarship (top 5%)

Language Qualifications

Chinese and English

Research Experience

My research experience and interest in the area of **vulnerability detection** and **compiler optimization** through the use of **deep learning techniques**. My objection is towards automated, robust and trustworthy machine learning for program modelling.

I have participated in and contributed to the following projects during my MSc and Ph.D. study. My work has contributed to **six publications**, **five patents** and three papers submission that is currently under review.

1. Automated Machine Learning for Program Modeling

- **Hybrid Learning-based Software Vulnerability Prediction** **December 2021 to June 2023**

Deep Learning (DL) is increasingly utilized to detect software bugs and vulnerabilities. These learning based approaches extract program representations from static code sources like code texts, Abstract Syntax Trees, and Program Data and Control Flow Graphs. DL may encounter challenges related to intricate and ambiguous details stemming from redundant statements, complex data structures, and extensive execution paths present in the source code. This can lead to overly cautious determinations and a reduced true positive ratio.

We proposed use DL to learn program presentations by combining static source code information and dynamic program execution traces. We employs unsupervised active learning techniques to determine a subset of important paths to collect dynamic symbolic execution traces. By implementing a focused symbolic execution solution, we brings the benefits of static and dynamic code features while reducing the expensive symbolic execution overhead.

We integrate our approach with fuzzing techniques to detect function-level code vulnerabilities in C programs from 20 open-source projects. We has successfully uncovered vulnerabilities in all tested projects,

identifying 53 unique vulnerabilities and yielding 36 new, unique CVE IDs and also significantly outperforms 14 prior methods by providing higher accuracy and lower false positive rates.

This work led to one paper under review for which I am the first author.

- **Automatic Reinforcement Learning Model Architecture Design** **July 2020 to Apr 2022**

While programmers apply reinforcement learning (RL) to their domain, the first and most important step is to design the RL architecture for their tasks. This process includes environment preparation, feature extraction, network design, and parameter tuning. Inappropriate features cannot correctly represent the program, and randomly combining neural networks cannot support model training. In other words, expertise creates a barrier between programmers and RL.

We have proposed an open-source framework to automate the RL architecture search and parameter tuning process, making it easier to integrate RL into compilers. The framework can automatically assemble an RL architecture for targeted optimization from an extensible set of built-in RL components with just a few lines of Python, using an easy-to-use API. A key feature of our framework is the utilization of deep reinforcement learning and multi-task learning techniques to develop a meta-optimizer that automatically identifies and fine-tunes the suitable RL architecture using training benchmarks.

We have applied our approach to four code optimization problems: optimizing image pipelines, neural network code generation, code size reduction, and superoptimization. Experimental results consistently demonstrate its superiority over hand-tuned methods, delivering enhanced overall performance and accelerating the deployment-stage search by an average of 1.75x (with increases of up to 100x).

*This work has led to one paper published in **ACM CC 2022 [1]** for which I am the first author and an open source framework in Github which has collected more than **100 stars**.*

- **Deep Program Structure Modeling using Graph Neural Networks** **June 2019 to December 2020**

Deep learning is emerging as a promising technique for building predictive models to support code-related tasks like compiler optimization. One of the critical aspects of building a successful predictive model is having the right representation to characterize the model input for the given task. Existing approaches in the area typically treat the program structure as a sequential sequence but fail to capitalize on the rich semantics of data and control flow information, for which graphs are a proven representation structure.

We proposed to use graph neural networks (GNNs) to automatically learn useful code representations from graph-based program structures. I designed a novel GNN for capturing the syntax and semantic information from the program abstract syntax tree and the control and data flow graph. As a departure from existing GNN-based code modeling techniques, our network simultaneously learns over multiple relations of a program graph. This capability enables the learning framework to distinguish and reason about the diverse code relationships, be it data or a control flow or any other relationships that may be important for the downstream processing task.

We apply our approach to four representative tasks that require a strong ability to reason about the program structure: heterogeneous device mapping, parallel thread coarsening, loop vectorization, and code vulnerability detection. We evaluate our approach on programs written in OpenCL, C, Java, Php, and Swift. Experimental results show that our approach consistently outperforms all competing methods across evaluation settings.

*This work led to one paper published at **IEEE TIFS [2]** for which I am the first author, and two papers published at and **ACM PACT 2020 [3] and JISA [4]** (Impact factor: 3.579) for which I am the co-author.*

2. Robust Machine Learning for Program Modeling **April 2022 to now**

In recent years, machine learning has emerged as a powerful tool for assisting code analysis and optimization tasks. Machine learning models can be vulnerable to changes in the deployment environment.

Even slight alterations in hardware or application workloads can severely impact their accuracy. One primary cause of this issue is data drift, which occurs when there's a discrepancy between training and test data distributions (past training data will be a good representation of future test data).

We applied our system to 11 representative machine learning models, covering optimization tasks such as heterogeneous device mapping, GPU thread coarsening, loop vectorization, and source-code level bug detection. We demonstrate that our system can successfully identify an average of 90% (and up to 100%) of mispredicted test samples. Additionally, using incremental learning, Prom substantially enhances prediction performance in the operational environment.

This work led to one paper under review for which I am the first author.

3. Zero Shot Learning **August 2021 - November 2021**

My work primarily revolves around the creation and implementation of machine learning models, tools, and infrastructure. I ensure the seamless transfer of these components to downstream tasks, ensuring end-to-end integration within Alibaba's extensive pre-trained model ecosystem. My performance and contributions have provided strong positive feedback to Alibaba DAMO Academy, encouraging the continuous advancement of this program.

4. Code Virtualization and Software Protection **September 2018 - March 2019**

During my first year of postgraduate study, I was involved in using code virtualization techniques for software protection. I helped running experiments and developing the software framework, from which I learn how to do research, run experiments, and present results.

*This work has led to two papers published in **IEEE Access** (Impact factor: 3.745) **[5]** and **[6]**, for which I am a co-author.*

Programming Skills

C++, Python

Machine/Deep/Reinforcement Learning; Software Security; Program Optimization;