KUSH JAIN

kdjain@andrew.cmu.edu • (650) 965 1750 • https://www.linkedin.com/in/kush-jain/• https://www.kushjain.com

RESEARCH INTERESTS: My research focuses on developing new machine learning techniques for code and test generation. I am also interested in machine learning for software engineering and classical software testing.

EDUCATION

Carnegie Mellon University – School of Computer Science August 2021 – May 2025 (expected)
Doctor of Philosophy, Computer Science, advisor: Claire Le Goues
Research Areas: artificial intelligence for code, LLMs for code, fuzzing, mutation testing

University of Texas at Austin August 2018 – May 2021
Batchelor of Science, Computer Science, advisor: Milos Gligoric

AI FOR SOFTWARE ENGINEERING PUBLICATIONS

[1] Example Generation for OpenAPI Specifications using Large Language Models
Kush Jain, Kiran Kate, Jason Tsay, Claire Le Goues, Martin Hirzel
International Symposium on Software Testing and Analysis (submitted to ISSTA 2024)

[2] CAT-LM: Training Language Models on Aligned Code and Tests
Nikitha Rao*, **Kush Jain***, Uri Alon, Claire Le Goues, Vincent Hellendoorn (* = equal contribution)
Automated Software Engineering (ASE 2023)

[3] Contextual Predictive Mutation Testing
Kush Jain, Uri Alon, Alex Groce, and Claire Le Goues
Foundations of Software Engineering (FSE 2023)

AI FOR SOFTWARE ENGINEERING PROJECTS

Generating Readable High Coverage Test Suites using Large Language Models January 2023 – Present

- Generating high coverage test suites is both a challenging and costly endeavor for developers
- Our idea is to use mutants to guide test generation models, specifically using reinforcement learning to align models with tests that compile, pass, improve coverage and detect new mutants
- We plan to compare against EvoSuite, showing that our generated tests are more readable with similar coverage

Automatically Detecting ROS Misconfigurations using Language Models August 2023 – Present

- ROS misconfigurations lead to unexpected robot behavior, costing developers valuable time debugging
- No neural approach exists to detect these configurations, static analysis is very limited in the types of misconfigurations it can detect
- We plan to use language models to derive a new fault localization technique, currently have a dataset of 40k projects and misconfigurations

Example Generation for OpenAPI Specifications using Large Language Models May 2023 – Present

- OpenAPI specifications are widely used in industry in both small and large scale APIs
- We develop context diversity prompting, a novel prompting approach to generating OpenAPI parameters that maintains both correctness and diversity
- We evaluate our approach on numerous downstream tasks including fuzzing, dialog agents and human understanding, outperforming state of the art approaches

CAT-LM – Training Language Models on Aligned Code and Tests August 2022 – May 2023

- Testing is an integral part of software development, yet has received far less attention than areas such as code completion and bug repair
- We pretrain a 3B parameter large language model from scratch on the dual objective of autoregressive code completion and test generation, creating a dataset of over one million code test file pairs
- CAT-LM outperforms state of the art test generation models with training budgets orders of magnitude larger on both lexical and runtime metrics

Contextual Predictive Mutation Testing May 2022 – May 2023

- Mutation testing has been extensively researched in academia, yet has failed to achieve much industry traction due to its large compute requirements
- Our idea is to leverage language models over code to embed methods, operators, and test suites to predict whether a given mutant will be killed or not killed without running the test suite, saving significant compute
- Our novel approach outperforms state of the art techniques in this space, with major gains in time savings and model accuracy

WORK EXPERIENCE

IBM TJ Watson – *AI Research Intern;* Yorktown Heights, New York June 2023 – August 2023

- Developed a novel LLM prompting approach that produces correct and diverse OpenAPI parameter examples
- Improved state of the art in a wide range of domains including fuzzing, dialog systems and human API understanding

Amazon Lab126 – *Software Engineering Intern;* Sunnyvale, California June 2021 – August 2021

- Developed a webapp to manage the approval process for all prototype devices at Amazon
- Migrated data to DynamoDB and integrated unified authentication

Amazon Lab126 – *Software Engineering Intern;* Sunnyvale, California June 2020 – August 2020

- Developed a device search service for prototype devices using AWS lambda, ElasticSearch, API gateway and Database Migration Service to serve over 30 million requests a month, while dramatically improving existing search functionality in a schema change tolerant way, leveraging federated authentication

VISA Inc. – *Software Engineering Intern;* Austin, Texas June 2019 – August 2019

- Developed a dashboard to track health of core IT services using NodeJS, React and PowerShell. In the first two months of production, proactively detected five major outages, preventing over 250 support tickets
- Implemented a customized link shortener for VISA's internal network, using NodeJS, React and SQL.

OpsHub Inc. – *Software Engineering Intern;* Palo Alto, California June 2018 – August 2018

- Prototyped an Angular dashboard to visualize multi-system KPI's using the company's integration platform
- Proof of concept was successful, and company is looking to fully develop the product

OpsHub Inc. – *Software Engineering Intern;* Palo Alto, California June 2017 – August 2017

- Developed a model to analyze the riskiness of a source code file and to predict the number of bugs expected
- Got 70% accuracy and had visibility to improve it further by bringing in data from additional systems

PRESENTATIONS

- Contextual Predictive Mutation Testing. Presented at Foundations of Software Engineering, 2023
- Analyzing the Difference Between Code Coverage and Mutation Score. Presented at International Symposium on Software Reliability Engineering, 2023
- Mutation Analysis for Coq Verification Projects. Presented at Amazon Lab 126, 2021

SERVICE

- Student volunteer at the International Conference on Software Engineering, 2022 (ICSE 2022)

- Sub-reviewer for the International Conference on Software Engineering, 2023 (ICSE 2023)

OTHER PUBLICATIONS

[1] Mind the Gap: The Difference Between Coverage and Mutation Score Can Guide Testing Efforts
Kush Jain, Goutamkumar Tulajappa Kalburgi, Claire Le Goues, Alex Groce
International Symposium on Software Reliability Engineering (ISSRE 2023)

[2] Looking for Lacunae in Bitcoin Core's Fuzzing Efforts
Alex Groce, **Kush Jain**, Rijnard van Tonder, Goutamkumar Tulajappa Kalburgi, and Claire Le Goues
International Conference on Software Engineering (ICSE 2022)

[3] Registered Report: First, Fuzz the Mutants
Alex Groce, Goutamkumar Tulajappa Kalburgi, Claire Le Goues, **Kush Jain**, and Rahul Gopinath
International Fuzzing Workshop (FUZZING 2022)

[4] Programming and Execution Models for Parallel Bounded Exhaustive Testing
Nader Al Awar, **Kush Jain**, Christopher J. Rossbach, and Milos Gligoric
Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2021)

[5] mCoq: Mutation Analysis for Coq Verification Projects
Kush Jain, Karl Palmskog, Ahmet Celik, Emilio Jesus Gallego Arias, and Milos Gligoric
International Conference on Software Engineering Tool Demonstrations Track (ICSE 2021)

OTHER PROJECTS

Analyzing the Difference Between Code Coverage and Mutation Score August 2021 – August 2022
- Most testing efforts in industry use coverage as the primary metric of test-adequacy, with an even smaller subset using mutation score, however neither metric is fully sufficient
- We analyze the difference between code coverage and mutation score, and examine how it can guide real world testing efforts
- We find that this difference can detect both induced examples of testing inadequacy and real-world examples corresponding to GitHub commits

Looking for Lacunae in Bitcoin Core's Fuzzing Efforts August 2021 – August 2022
- Conducted cross-cryptocurrency mutation testing, benchmarking Bitcoin against other cryptocurrencies to understand its relative strengths, which underlined Bitcoin Core's superior file and project coverage
- Manually examined cases where coverage and mutation score diverged, pinpointing gaps in cryptocurrency testing efforts

First, Fuzz the Mutants August 2021 – May 2022
- Fuzzing is a popular testing technique that has begun to achieve more widespread adoption, however fuzzers still struggle to reach certain code snippets due to number or branches or expensive compile optimizations
- Our approach is to first apply mutation analysis to the target program being fuzzed, and then fuzz these mutants for half of the fuzzing budget to generate seeds that cover these hard-to-reach branches
- Results indicate that this approach can both lead to higher coverage and more bugs found on multiple fuzzing benchmarks

Tempo – Bounded Exhaustive Testing August 2020 – August 2021
- We develop Tempo: a programming and execution model for parallel bounded exhaustive testing. It supports hybrid test generation programs using two strategies: fork-based and re-execution based
- Evaluation shows fork-based works well for simple programs on GPUs while re-execution based handles complex programs better

- We also study Clang and GCC finding multiple bugs in both compilers

Mutation Analysis for Coq Verification Projects August 2020 – August 2021
- We perform mutation analysis on Coq formal specifications to detect partial or incomplete specifications
- Implemented a wide range of mutation operators ranging from simple (replace and with or) to complex ones (reversing Coq inductive cases)

OPEN SOURCE CONTRIBUTIONS

FuzzBench: added our fuzzers that fuzz mutants first and then run normal fuzzing over the benchmarks – https://github.com/google/fuzzbench

mCoq: mutation analysis tool for Coq verification projects, used by around a dozen developers from around the world – https://github.com/EngineeringSoftware/mcoq

SMUM-Checkin: open source code of our digital recordkeeping system used by Santa Maria Urban Ministry – https://github.com/UnconditionedLife/smum