THE UNIVERSITY OF SYDNEY

FACULTY OF ENGINEERING

Engineering Vacation Research Internship Program

COMPUTER SCIENCE RESEARCH PROJECTS - SUMMER 2022-23

FACULTY OF ENGINEERING

COMPUTER SCIENCE PROJECTS

CS2022-23/1 Privacy preserving models for decision tree algorithm

Supervisor: Dr Nazanin Borhan

Eligibility: WAM>75 and Undergraduate candidates must have already completed at least 96 credit points towards their undergraduate degree at the time of application.

Project Description:
There are always privacy considerations when publishing the results of a data mining or machine learning models publicly. However, privacy preserving models can be a good solution for this problem, without losing too much value and utility from published data. There will be always a trade-off between how much you preserved the data, and how does that effect the accuracy of the published data. In this research, you are looking at decision tree learning algorithm as a popular example of data mining models and explore different privacy preserving techniques that are available on this algorithm. There are a wide range of practical, theoretical and statical models to preserve the result of decision trees in the literature, and this research can give you a good foundation on the existing models to identify the available techniques in the literature, assess and quantify the guarantees provided by each of them, and compare the trade-offs.

Requirement to be on campus: No

CS2022-23/2 Faithful Visualisation for Big Complex Data

Supervisors: Prof. Seokhee Hong; Dr. Amyra Meidiana

Eligibility: Skills Required: Data Structure and Algorithms and Programming (Java, C++, Python, Javascript)

Project Description:
Technological advances have increased data volumes in the last few years, and now we are experiencing a “data deluge” in which data is produced much faster than it can be understood by humans.

These big complex data sets have grown in importance due to factors such as international terrorism, the success of genomics, increasingly complex software systems, and widespread fraud on stock markets.

Visualisation is a powerful tool to compute good geometric representation of abstract data to support analysts to find insights and patterns in big complex data sets.

This project aims to design, implement, and evaluate new visualisation algorithms for faithful visualisation of big complex data, to enable humans to find ground truth structure in big complex data sets, such as social networks and biological networks.

These new visualisation methods are in high demand by industry for the next generation visual analytic tools.

Requirement to be on campus: No

CS2022-23/3 Neural Interfaces for Visually Impaired as Tactile Enabling Technology

Supervisor: Dr Anusha Withana

Eligibility Criteria: You will work with the supervisor and a PhD student, and we expect you are a fast learner. Excellent skills in programming, skills in embedded systems, machine learning, knowledge in design and fabrication, and human computer interaction are added benefits.

Project Description:
Modern computers frequently use visual and auditory interfaces. For example, we can see the visual information on our screens and hear the audio computers generate. However, one important modality is less explored, that is, what we feel through our skin. We feel vibrations, temperature, and pressure through sensory receptors in our skin. This modality is particularly important to create enabling interfaces, for instance computer interfaces used by visually impaired people. In this project, we will explore how we can create enabling technologies using novel tactile interfaces, particularly an interface directly communicates with our neural system.

Requirement to be on campus: Yes * *dependent on government's health advice*

CS2022-23/4 Predictive Gesture Classification in Virtual Reality (VR)

Supervisor: Dr Anusha Withana

Eligibility: You will work with the supervisor and a student, and we expect you are a fast learner. Excellent skills in programming, skills in embedded systems, machine learning, knowledge in design and fabrication, and human computer interaction are added benefits.

Project Description:
Can we predict a player's next move in a VR game? Gestural input, for example, using your hand movements as input has become one of the most popular input technologies for rapidly developing virtual reality (VR) and augmented reality (AR) applications (eg. GearVR, HTC Vive, etc.). Gestures, such as hand movements and poses in space, are an essential part of our daily communication (ie. body language) and thus create an intuitive modality for interacting with these immersive new computer applications. In this project we focus on predicting hand movements for the purpose of pre-recognising user activities in VR. The project will build our existing work on continuous hand movement recognition (See video in the link) and the data we have collected.
https://www.dropbox.com/s/jkgik9mj4bd5e83/uist21a-sub1729-cam-i27.mp4?dl=0

Requirement to be on campus: Yes * *dependent on government's health advice*

CS2022-23/5 Drawing graphs with smooth paths and cycles

Supervisor: Peter Eades

Eligibility
Essential:
- Good knowledge of graph theory and continuous curve mathematics
- Some knowledge of graph drawing algorithms

Desirable:
- Some experience in either JavaScript or python

Project Description:
Planar graphs and networks are typically drawn with polyline edges (that is, each edge is a connected sequence of straight-line segments). Much effort has been spent over the past two decades in designing algorithms to minimise the number of straight-line segments in an

edge. In the best case, each edge is a single straight-line segment; trivially, a straight-line segment is a smooth curve.

However, drawing each edge as a straight-line segment ignores the *paths* and *cycles* in the graph, which can bend sharply.

We aim to draw planar graphs such that designated sets of paths and cycles are *smooth*, that is, have C^k continuity where $k > 1$. We will use cubic Bezier curves. The challenge is to maintain planarity as well as smoothness.

Specifically, we aim to show that smoothness is achievable for directed paths in upward planar graphs, and for cycles in 4-regular planar graphs.

Requirement to be on campus: Yes * *dependent on government's health advice*

CS2022-23/6 Dimensionality Reduction on the Simplex: Domain Reduction made practical

Supervisor: Dr Clement Canonne

Eligibility: Mathematical background in (basic) discrete probability/statistics, and familiarity with coding in Python or C++.

Project Description:
A recent line of (theoretical) work has proposed and relied on a new algorithmic technique, "domain reduction", to hash the (large) domain of randomly sampled data points to a much smaller domain, while preserving some statistical properties of the data distribution. This technique has led to several (theoretical) results on a variety of statistical questions in hypothesis testing under privacy or bandwidth constraints.

Despite its versatility, this technique (essentially a dimensionality reduction for probability distributions) suffers a major drawback: the theoretical analysis might be overly pessimistic, and as a result it is not in its current form clear it's efficient in practice. This project's aims are twofold:

- Understand and improve the analysis of this technique
- Implement it and empirically assess its performance

Requirement to be on campus: No

CS2022-23/7 Explanations and Formal Language Theory

Supervisor: Dr Sasha Rubin

Eligibility: Finished at least 3 years of undergraduate courses in computer science or mathematics, with an HD in COMP202022/292022 or similar course.

Project Description:
The goal is to define and study the computational problem of finding explanations of statements such as "string s is in the regular language L". This has applications to program repair and to giving explanations to actions (made by controllers, policies). The project will involve exploring different logical formalisms in which to describe explanations. A starting point would be first-order logic.

Requirement to be on campus: No

CS2022-23/8 Verification of Probabilistic Systems

Supervisor: Dr Sasha Rubin

Eligibility: Finished at least 3 years of undergraduate courses in computer science or mathematics, with an HD in COMP202022/292022 or similar course.

Project Description:
The purpose of this project is to establish precise complexity bounds on verification problems of probabilistic systems (such as Markov chains, Markov Decision Processes) from declarative specifications, especially over finite traces (such as LTLf, nondeterministic and alternating automata).

Requirement to be on campus: No

CS2022-23/9 Automatic machine learning with neural architecture search for AI applications

Supervisor: Dr Chang Xu

Eligibility: Essential deep learning knowledge and programming skills.

Project Description:
As an AutoML (Automatic machine learning) algorithm, neural network structure search can adaptively find the most suitable structure for any AI architecture such as CNN or transformer, and tasks such as classification, detection, etc. In this project, we will directly teach students AutoML knowledge and provide professional and meticulous academic guidance to help students accumulate rich AI research and application experience. This project will guide students to complete CV tasks such as image classification, image detection, etc. according to their own wishes.

Requirement to be on campus: No

CS2022-23/10 Design and Implementation of Federated Learning Frameworks for Human Activity Recognition

Supervisors: Dr Omid Tavallaie and Professor Albert Zomaya

Eligibility: Specific required knowledge, skills, and/or technology:
Python programming, Machine learning, Android programming, Amazon EC2 servers.

Project Description:
Personal mobile sensing is used for activity monitoring applications such as healthcare rehabilitation. In recent years, by applying deep learning on resource-limited devices, these applications have achieved significant attention from both industry and academia. Compared to cloud-based solutions, running deep learning algorithms on mobile devices provides several benefits including data privacy preservation and low-latency processes. On the other hand, personal mobile sensing applications are mostly user-specific and highly affected by environment. As a result, continuous local changes may seriously affect the performance of a global model generated by deep/federated learning models. To enhance running deep learning algorithms on resource limited devices, we design and implement new deep/federated learning models on android mobile devices.

Requirement to be on campus: No

CS2022-23/11 Delta Debugging for Parsers

Supervisor: Dr Rahul Gopinath

Eligibility: You will work with the supervisor directly for this project. You should be a fast learner. Excellent skills in programming (Basic Python & Java knowledge is necessary), problem solving, as well as the ability to work independently are required.

Project Description:
In cybersecurity, your fuzzers can often come up with a very large input that causes a crash. To debug the crash, one often needs a much smaller input that reproduces the crash. HDD is an algorithm used to reduce the input size if the input conforms to a grammar. However, it does not work well if the grammar is incomplete such as in many handwritten parsers and the input can't be parsed.

In this project, we will explore input-repair based techniques to repair and then parse the input and compare it with using original delta debugging (which does not require a grammar) and coverage-based strategies to identify hierarchies in the partially parsed input.

This project, if completed successfully, may be extended for a paper in one of the A/A* conferences in software engineering.

Requirement to be on campus: No

CS2022-23/12 Error Correction with Grammars (Earley)

Supervisor: Dr Rahul Gopinath

Eligibility: You will work with the supervisor directly for this project. You should be a fast learner. Excellent skills in programming (Basic Python & Java knowledge is necessary), understanding research papers, problem solving, as well as the ability to work independently are required.

Project Description:
Data aggregation and cleaning is one of the most important steps in data science. The data may come from multiple sources and hence, may not match the required format exactly. This is especially an issue if the required format is a rich format like JSON, XML, S-Expr etc. In such cases, we may have to rely on available error correction algorithms (at best) or manual labour (at worst). While numerous error correction algorithms exist, error correction for context-free grammars is still lagging with the best-known algorithm from 1972 from Aho et al. which extends Earley parsing.

In this project, we will explore how to implement faster error correction for context-free grammars by extending the Aho's algorithm and compare it with the original.

This project, if completed successfully, may be extended for a paper in one of the A/A* conferences in software engineering.

Requirement to be on campus: No

CS2022-23/13 Error Correction with Grammars (GLL)

Supervisor: Dr Rahul Gopinath

Eligibility: You will work with the supervisor directly for this project. You should be a fast learner. Excellent skills in programming (Basic Python & Java knowledge is necessary), understanding research papers, problem solving, as well as the ability to work independently are required.

Project Description:
Data aggregation and cleaning is one of the most important steps in data science. The data may come from multiple sources and hence, may not match the required format exactly. This is especially an issue if the required format is a rich format like JSON, XML, S-Expr etc. In such cases, we may have to rely on available error correction algorithms (at best) or manual labour (at worst). While numerous error correction algorithms exist, error correction for context-free grammars is still lagging with the best-known algorithm from 1972 from Aho et. al. which extends Earley parsing.

In this project, we will explore how to implement faster error correction for context-free grammars by extending the GLL parsing algorithm (a faster parsing technique for general context-free grammars) and compare it with the original algorithm from Aho et. al.

This project, if completed successfully, may be extended for a paper in one of the A/A* conferences in software engineering.

Requirement to be on campus: No

CS2022-23/14 Error Correction with Grammars (GLR)

Supervisor: Dr Rahul Gopinath

Eligibility: You will work with the supervisor directly for this project. You should be a fast learner. Excellent skills in programming (Basic Python & Java knowledge is necessary), understanding research papers, problem solving, as well as the ability to work independently are required.

Project Description:
Data aggregation and cleaning is one of the most important steps in data science. The data may come from multiple sources and hence, may not match the required format exactly. This is especially an issue if the required format is a rich format like JSON, XML, S-Expr etc. In such cases, we may have to rely on available error correction algorithms (at best) or manual labour (at worst). While numerous error correction algorithms exist, error correction for context-free grammars is still lagging with the best-known algorithm from 1972 from Aho et. al. which extends Earley parsing.

In this project, we will explore how to implement faster error correction for context-free grammars by extending the GLR parsing algorithm (a faster parsing technique for general context-free grammars) and compare it with the original algorithm from Aho et. al.

This project, if completed successfully, may be extended for a paper in one of the A/A* conferences in software engineering.

Requirement to be on campus: No

CS2022-23/15 Optimizing Grammar Fuzzers

Supervisor: Dr Rahul Gopinath

Eligibility: You will work with the supervisor directly for this project. You should be a fast learner. Excellent skills in programming (Good Python, Java, and C Skills. Knowledge of assembly is really good to have.), problem solving, as well as the ability to work independently are required.

Project Description:
Grammar based Fuzzers are one of the most important tools in cybersecurity. The effectiveness of fuzzing is in many cases determined by the speed at which inputs can be generated, and highly performant grammar fuzzers are extremely important. When making a grammar fuzzer, there are multiple tradeoffs that can be made to make it performant. These include fixing the depth of recursion, at which point it becomes automata

that can be easily implemented in code without subroutines, or supercompiling the grammar (i.e., eliminating redundant procedure calls), or superoptimizing the produced code (i.e., assembly level optimizations using solvers), or come up with other better optimization techniques.

This project will explore how to make grammar fuzzers much more performant and compare it with the automata-based approach.

Requirement to be on campus: No

CS2022-23/16 Inferring context-free grammars for Blackbox programs

Supervisor: Dr Rahul Gopinath

Eligibility: You will work with the supervisor directly for this project. You should be a fast learner. Excellent skills in programming (Basic Python & Java knowledge is necessary), problem solving, as well as the ability to work independently are required.

Project Description:
Fuzzing Blackbox programs that use an unknown but rich input structure (e.g. conforms to a context-free grammar) has been traditionally very difficult. The problem is that any input that does not conform to the expected input structure will get rejected almost immediately and will not penetrate the actual program logic. Hence, there has been numerous recent attempts to infer the grammar from such Blackbox programs, and to use such inferred grammars for fuzzing. This has been, however, rather difficult due to a theorem by Gold which says that inferring context-free grammars from Blackbox programs is as difficult as cracking RSA.

In this project, we will explore how to get around this requirement by relaxing the opacity constraint a little bit. We will assume that the Blackbox program will tell us when the input is completely incorrect verses whether the input is a valid prefix that can be fixed by adding some suffix. This can allow us to guess at the internal state of the program, and hence, infer the input grammar.

This project, if completed successfully, may be extended for a paper in one of the A/A* conferences in software engineering.

Requirement to be on campus: No

CS2022-23/17 Reproducible HTML Notebooks

Supervisor: Dr Rahul Gopinath

Eligibility: You will work with the supervisor and possibly a student, and we expect you are a fast learner. Excellent skills in programming (Excellent JavaScript knowledge, and basic Python skills is necessary), problem solving, as well as the ability to work independently are required.

Project Description:
Reproducibility is the core of science, and software engineering unfortunately doesn't have a great reputation when it comes to reproducibility. This project attempts to change that. The idea is to start with the WASM powered Jupyter notebooks, bundle them into a completely self-contained HTML file which can contain implementations of algorithms, and can be passed around and modified by other researchers and students. The modified HTML files should be savable using the TiddlyWiki (https://tiddlywiki.com/) technique.

This project, if completed successfully, may provide a starting foundation for a workshop on reproducible software engineering, and lead to being adopted by researchers worldwide.

Requirement to be on campus: No

CS2022-23/18 Mutation Analysis for Fuzzers

Supervisor: Dr Rahul Gopinath

Eligibility: You will work with the supervisor directly for this project. You should be a fast learner. Excellent skills in programming (Basic Python & Java knowledge is necessary), problem solving, as well as the ability to work independently are required.

Project Description:
Fuzzers are test generators that specialize on producing millions of inputs at a time. The next frontier of fuzzing is to incorporate better oracles, that is verifiers of behavior. However, for this to happen, we need evaluators that can verify behavior. Mutation analysis is the gold standard for evaluating behavior. It accomplishes this by generating thousands of possibly buggy variants of the given program and checking how many of these are detected. To use mutation analysis for fuzzing, its computational expenditure needs to be brought under control. One way to do that is to evaluate multiple bugs at the same time in each execution. However, to do that, we need to ensure that the inserted bugs do not interact with each other. This project will explore the best way to identify independent bugs so that they can be combined.

This project, if completed successfully, may be extended for a paper in one of the A/A* conferences in software engineering.

Requirement to be on campus: No

CS2022-23/19 Using side channels for feedback driven fuzzing

Supervisor: Dr Rahul Gopinath

Eligibility: You will work with the supervisor directly for this project. You should be a fast learner. Excellent skills in programming (expert C & UNIX knowledge is necessary), problem solving, as well as the ability to work independently are required.

Project Description:
Fuzzing is a premier technique in Cybersecurity and is used by software engineers to ensure that our systems are reliable, and by pen testers to identify possible avenues of attack. Naive fuzzing (blackbox) is usually ineffective in practice, and feedback from the program is necessary for effective fuzzing. Current fuzzing tools such as AFL rely on extensive program instrumentation for feedback, which unfortunately limits their use. Many systems can't be instrumented at all, and in some other systems, instrumentation can change the behaviour of the system.

One way we can avoid instrumentation is by looking at the memory contents at the end of execution, the system calls that were made, and other side channels. This project will explore such side channels especially memory and system calls to find if we can do better than AFL in detecting faults in programs.

This project, if completed successfully, may be extended for a paper in one of the A/A* conferences in software engineering.

Requirement to be on campus: No